Combinatorial Optimization and Theoretical Computer Science

Interfaces & Perspectives

Edited by Vangelis Th. Paschos



ISTE

WILEY

Combinatorial Optimization and Theoretical Computer Science

Combinatorial Optimization and Theoretical Computer Science

Interfaces and Perspectives

30th Anniversary of the LAMSADE

Edited by
Vangelis Th. Paschos

ISTE

WILEY

First published in Great Britain and the United States in 2008 by ISTE Ltd and John Wiley & Sons, Inc.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned address:

ISTE Ltd
6 Fitzroy Square
London W1T 5DX
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

© ISTE Ltd, 2008

The rights of Vangelis Th. Paschos to be identified as the author of this work have been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN: 978-1-84821-021-9

Printed and bound in Great Britain by Antony Rowe Ltd, Chippenham, Wiltshire.



Contents

Preface 15

Chapter 1. The Complexity of Single Machine Scheduling Problems under Scenario-based Uncertainty 23
Mohamed Ali ALOULOU and Federico DELLA CROCE

1.1. Introduction 23
1.2. Problem *MinMax*(1|*prec*|$f_{\max}$, θ) 25
1.2.1. Uncertainty on due dates 25
1.2.2. Uncertainty on processing times and due dates 27
1.3. Problem *MinMax*(1||$\sum w_j C_j, w_j$) 28
1.4. Problem *MinMax*(1||$\sum U_j, \theta$) 29
1.4.1. Uncertainty on due dates 29
1.4.2. Uncertainty on processing times 29
1.5. Bibliography 35

Chapter 2. Approximation of Multi-criteria Min and Max *TSP*(1, 2) 37
Eric ANGEL, Evripidis BAMPIS, Laurent GOURVÈS and Jérôme MONNOT

2.1. Introduction 37
2.1.1. The traveling salesman problem 37
2.1.2. Multi-criteria optimization 38
2.1.3. Organization of the chapter 39
2.2. Overview 39
2.3. The bicriteria *TSP*(1, 2) 40
2.3.1. Simple examples of the non-approximability 42
2.3.2. A local search heuristic for the bicriteria *TSP*(1, 2) 43
2.3.3. A nearest neighbor heuristic for the bicriteria *TSP*(1, 2) 48
2.3.4. On the bicriteria Max *TSP*(1, 2) 52

2.4. *k*-criteria *TSP*(1, 2) 55
2.4.1. Non-approximability related to the number of generated solutions 56
2.4.2. A nearest neighbor heuristic for the *k*-criteria *TSP*(1, 2) 60
2.5. Conclusion 67
2.6. Bibliography 68

Chapter 3. Online Models for Set-covering: The Flaw of Greediness 71
Giorgio AUSIELLO, Aristotelis GIANNAKOS and Vangelis Th. PASCHOS

3.1. Introduction 71
3.2. Description of the main results and related work 73
3.3. The price of ignorance 76
3.4. Competitiveness of `TAKE-ALL` and `TAKE-AT-RANDOM` 77
3.4.1. `TAKE-ALL` algorithm 77
3.4.2. `TAKE-AT-RANDOM` algorithm 78
3.5. The nasty flaw of greediness 79
3.6. The power of look-ahead 82
3.7. The maximum budget saving problem 88
3.8. Discussion 91
3.9. Bibliography 91

Chapter 4. Comparison of Expressiveness for Timed Automata and Time Petri Nets 93
Béatrice BÉRARD, Franck CASSEZ, Serge HADDAD, Didier LIME and Olivier-Henri ROUX

4.1. Introduction 93
4.2. Time Petri nets and timed automata 95
4.2.1. Timed transition systems and equivalence relations 96
4.2.2. Time Petri nets 98
4.2.3. Timed automata 101
4.2.4. Expressiveness and equivalence problems 103
4.3. Comparison of semantics *I*, *A* and *PA* 104
4.3.1. A first comparison between the different semantics of TPNs 104
4.3.2. A second comparison for standard bounded TPN 107
4.4. Strict ordering results 111
4.5. Equivalence with respect to timed language acceptance 113
4.5.1. Encoding atomic constraints 113
4.5.2. Resetting clocks 115
4.5.3. The complete construction 116
4.5.4. $\Delta(\mathcal{A})$ and $\mathcal{A}$ accept the same timed language 117
4.5.5. Consequences of the previous results 121

4.6. Bisimulation of TA by TPNs 122
4.6.1. Regions of a timed automaton 122
4.6.2. From bisimulation to uniform bisimulation 124
4.6.3. A characterization of bisimilarity 128
4.6.4. Proof of necessity 129
4.6.5. First construction 131
4.6.6. Second construction 137
4.6.7. Complexity results 141
4.7. Conclusion 142
4.8. Bibliography 143

Chapter 5. A "Maximum Node Clustering" Problem 145
Giuliana CARELLO, Federico DELLA CROCE, Andrea GROSSO and Marco LOCATELLI

5.1. Introduction 149
5.2. Approximation algorithm for the general problem 147
5.3. The tree case 150
5.3.1. Dynamic programming 150
5.3.2. A fully polynomial time approximation scheme 155
5.4. Exponential algorithms for special cases 156
5.5. Bibliography 159

Chapter 6. The Patrolling Problem: Theoretical and Experimental Results .. 161
Yann CHEVALEYRE

6.1. Introduction 161
6.2. The patrolling task 162
6.3. Previous work 164
6.4. The cyclic strategies 165
6.4.1. Patrolling with a single-agent 165
6.4.2. Extending to multi-agent case 167
6.4.3. Optimality of cyclic strategies 168
6.5. Partition-based strategies 169
6.6. Experiments 171
6.7. Conclusion 172
6.8. Bibliography 174

Chapter 7. Restricted Classes of Utility Functions for Simple Negotiation Schemes: Sufficiency, Necessity and Maximality 175
Yann CHEVALEYRE, Ulle ENDRISS and Nicolas MAUDET

7.1. Introduction 175
7.2. Myopic negotiation over indivisible resources 177
7.2.1. Negotiation problems and deals 178

7.2.2. Negotiating with money 179
7.2.3. Negotiating without money 180
7.3. Convergence for restricted classes of utility functions 180
7.4. Modular utility functions and variants 182
7.5. Sufficient classes of utility functions 184
7.5.1. Framework with money 184
7.5.2. Framework without money 185
7.6. Necessity issues 185
7.6.1. Modularity is not necessary 186
7.6.2. There is no sufficient and necessary class 186
7.6.3. Evaluating conditions on profiles of utility functions is intractable 187
7.7. Maximal classes of utility functions 191
7.7.1. Framework with money 192
7.7.2. Framework without money 196
7.8. Conclusion 198
7.9. Bibliography 199

Chapter 8. Worst-case Complexity of Exact Algorithms for NP-hard Problems 203
Federico DELLA CROCE, Bruno ESCOFFIER, Marcin KAMIŃSKI and Vangelis Th. PASCHOS

8.1. MAX-CUT 204
8.1.1. Extending a partial partition of vertices 206
8.1.2. An algorithm for graphs with bounded maximum degree 209
8.1.3. An algorithm for general graphs 210
8.2. Pruning the search tree by dominance conditions: the case of MAXCUT-3 211
8.2.1. Dominance conditions 212
8.2.2. The worst-case upper-time bound for MAX-CUT-3 214
8.3. A more careful analysis for pruning: the case of MIN 3-DOMINATING SET 226
8.3.1. Analysis 227
8.3.2. Counting 228
8.3.3. Case n°1 228
8.3.4. Case n°2 232
8.3.5. Case n°3 234
8.3.6. Complexity analysis 237
8.4. Bibliography 239

Chapter 9. The Online Track Assignment Problem 241
Marc DEMANGE, Gabriele DI STEFANO and Benjamin LEROY-BEAULIEU

9.1. Introduction 241
9.1.1. Related works 242
9.1.2. Results 243
9.2. Definitions and notations 243
9.3. Bounds for the permutation graph in a left-to-right model 245
9.4. Bounds for overlap graphs 249
9.5. Bounds for permutation graphs in a more general model 252
9.6. Conclusion 256
9.7. Bibliography 256

Chapter 10. Complexity and Approximation Results for the Min Weighted Node Coloring Problem 259
Marc DEMANGE, Bruno ESCOFFIER, Jérôme MONNOT, Vangelis Th. PASCHOS and Dominique DE WERRA

10.1. Introduction 259
10.2. General results 262
10.2.1. Structural properties 262
10.2.2. Approximation results 264
10.2.2.1. General graphs 264
10.2.2.2. k-colorable graphs 266
10.2.2.3. When list coloring is easy 268
10.3. Weighted node coloring in triangle-free planar graphs 270
10.4. Weighted node coloring in bipartite graphs 273
10.4.1. Hardness results 273
10.4.2. P_5-free bipartite graphs 278
10.5. Split graphs 280
10.5.1. Complexity result 280
10.5.2. Approximation result 281
10.6. Cographs 284
10.7. Interval graphs 285
10.8. Bibliography 286

Chapter 11. Weighted Edge Coloring 291
Marc DEMANGE, Bruno ESCOFFIER, Giorgio LUCARELLI, Ioannis MILIS, Jérôme MONNOT, Vangelis Th. PASCHOS and Dominique DE WERRA

11.1. Introduction 291
11.2. Related problems 293
11.3. Preliminaries and notation 294
11.4. Complexity and (in) approximability 295

11.5. Graphs of $\Delta = 2$ 296
11.6. A 2-approximation algorithm for general graphs 298
11.7. Bipartite graphs 299
11.7.1. A $\frac{2\Delta - 1}{3}$-approximation algorithm 299
11.7.2. A $\frac{7}{6}$-approximation algorithm for $\Delta = 3$ 301
11.7.3. An approximation algorithm for $\Delta \leq 7$ 307
11.8. Trees 310
11.8.1. Feasible *k*-Coloring and bounded degree trees 310
11.8.2. Stars of chains 312
11.9. Conclusions 315
11.10. Bibliography 315

Chapter 12. An Extensive Comparison of 0-1 Linear Programs for the Daily Satellite Mission Planning 319
Virginie GABREL

12.1. Introduction 319
12.2. Different formulations for the daily satellite mission planning problem 320
12.2.1. The daily satellite mission planning problem 320
12.2.2. The "natural" model 320
12.2.3. The flow formulation 321
12.3. Models comparison 324
12.3.1. About the stable set polytope 324
12.3.2. Stable set polytope and daily mission planning problem formulations 325
12.4. Experiments and results 326
12.5. Conclusion 327
12.6. Bibliography 328

Chapter 13. Dantzig-Wolfe Decomposition for Linearly Constrained Stable Set Problem 329
Virginie GABREL

13.1. Introduction 329
13.2. The Dantzig-Wolfe decomposition in 0-1 linear programming 330
13.3. The stable set problem with additional linear constraints 333
13.4. Dantzig-Wolfe decomposition on stable set constraints: strengthening the LP-relaxation 334
13.4.1. Gap between LP-relaxations of master and initial problems 334
13.4.2. The case of co-comparability graph 335
13.4.3. A decomposition scheme for general graphs 337

13.5. Conclusion 337
13.6. Bibliography 338

Chapter 14. Algorithmic Games 339
Aristotelis GIANNAKOS, Vangelis Th. PASCHOS and Olivier POTTIÉ

14.1. Preliminaries 340
14.1.1. Basic notions on games 340
14.1.2. Complexity classes mentioned throughout the chapter 343
14.2. Nash equilibria 345
14.3. Mixed extension of a game and Nash equilibria 347
14.4. Algorithmic problems 348
14.4.1. Games of succinct description 349
14.4.2. Results related to the computational complexity of a mixed equilibrium 349
14.4.3. Counting the number of equilibria in a mixed strategies game 355
14.5. Potential games 355
14.5.1. Definitions 356
14.5.2. Properties 356
14.6. Congestion games 360
14.6.1. Rosenthal's model 360
14.6.2. Complexity of congestion games (Rosenthal's model) 363
14.6.3. Other models 364
14.7. Final note 368
14.8. Bibliography 368

Chapter 15. Flows! 373
Michel KOSKAS and Cécile MURAT

15.1. Introduction 373
15.2. Definitions and notations 374
15.3. Presentation of radix trees 375
15.3.1. Database management 377
15.3.2. Image pattern recognition 379
15.3.3. Automatic translation 380
15.4. Shortest path problem 380
15.4.1. Finding paths in $G[t]$ 382
15.4.2. Complexity 385
15.5. The Flow problem 388
15.5.1. The Ford-Fulkerson algorithm 388
15.6. Conclusion 390
15.7. Bibliography 391

Chapter 16. The Complexity of the Exact Weighted Independent Set Problem 393
Martin MILANIČ and Jérôme MONNOT

16.1. Introduction 393
16.2. Preliminary observations 397
16.3. Hardness results 399
16.3.1. Bipartite graphs 400
16.3.2. A more general hardness result 403
16.4. Polynomial results 405
16.4.1. Dynamic programming solutions 406
16.4.1.1. mK_2-free graphs 406
16.4.1.2. Interval graphs 407
16.4.1.3. k-thin graphs 408
16.4.1.4. Circle graphs 409
16.4.1.5. Chordal graphs 410
16.4.1.6. AT-free graphs 413
16.4.1.7. Distance-hereditary graphs 415
16.4.1.8. Graphs of treewidth at most k 417
16.4.1.9. Graphs of clique-width at most k 422
16.4.2. Modular decomposition 424
16.5. Conclusion 428
16.6. Bibliography 429

Chapter 17. The Labeled Perfect Matching in Bipartite Graphs: Complexity and (in)Approximability 433
Jérôme MONNOT

17.1. Introduction 433
17.2. The 2-regular bipartite case 436
17.3. Some inapproximation results 438
17.4. The complete bipartite case 446
17.5. Bibliography 452

Chapter 18. Complexity and Approximation Results for Bounded-size Paths Packing Problems 455
Jérôme MONNOT and Sophie TOULOUSE

18.1. Introduction 455
18.1.1. Bounded-size paths packing problems 455
18.1.2. Complexity and approximability status 457
18.1.3. Theoretical framework, notations and organization 458
18.2. Complexity of $\mathbf{P}_k$PARTITION and related problems in bipartite graphs 460

18.2.1. Negative results from the *k*-dimensional matching problem 460
18.2.1.1. *k*-dimensional matching problem 460
18.2.1.2. Transforming an instance of *k*DM into an instance of $\mathbf{P}_k$PACKING 460
18.2.1.3. Analyzing the obtained instance of $\mathbf{P}_k$ PACKING 462
18.2.1.4. **NP**-completeness and **APX**-hardness 463
18.2.2. Positive results from the maximum independent set problem ... 466
18.3. Approximating MAXW$\mathbf{P}_3$PACKING and MIN3-PATHPARTITION 466
18.3.1. MAXW$\mathbf{P}_3$PACKING in graphs of maximum degree 3 467
18.3.2. MAXW$\mathbf{P}_3$PACKING in bipartite graphs of maximum degree 3 470
18.3.3. MIN3-PATHPARTITION in general graphs 473
18.4. Standard and differential approximation of $\mathbf{P}_k$P 475
18.4.1. Differential approximation of $\mathbf{P}_k$P from the traveling salesman problem 475
18.4.2. Approximating $\mathbf{P}_4$P by means of optimal matchings 478
18.4.2.1. Description of the algorithm 478
18.4.2.2. General $\mathbf{P}_4$P within the standard framework 479
18.4.2.3. General $\mathbf{P}_4$P within the differential framework 482
18.4.2.4. Bi-valued metric $\mathbf{P}_4$P with weights 1 and 2 within the standard framework 483
18.4.2.5. Bi-valued metric $\mathbf{P}_4$P with weights *a* and *b* in the differential framework 486
18.5. Conclusion 491
18.6. Bibliography 492

Chapter 19. An Upper Bound for the Integer Quadratic Multi-knapsack Problem 495
Dominique QUADRI, Eric SOUTIF, Pierre TOLLA

19.1. Introduction 495
19.2. The algorithm of Djerdjour *et al.* 497
19.3. Improving the upper bound 498
19.4. An efficient heuristic to calculate a feasible solution 500
19.5. Computational results 500
19.6. Conclusion 503
19.7. Bibliography 503

List of Authors 507

Index 511

Preface

The LAMSADE[1] was established in 1976 as a research laboratory jointly funded by Paris-Dauphine University and the CNRS (the French National Science Foundation) oriented to decision aiding, mainly in the areas of multiple criteria decision aiding and linear programming.

It very soon aggregated the research activities on computer science conducted within Paris-Dauphine University. In 30 years the LAMSADE gained a world-wide reputation in operations research and decision aiding, while developing and strengthening a specific vision of computer science, that is management and decision oriented computer science (from the French term “informatique décisionnelle”). Today the LAMSADE is one of the very few research laboratories showing such originality in its research orientation.

During these years new specific research subjects came to enrich those already existing: multi-agent systems, distributed computing and databases. In this effort, the LAMSADE had to put together different interdisciplinary competencies: decision theory, operations research, mathematics, social sciences and several fields of computer science. At the turning point of its 30 years the LAMSADE is organizing its research activities around four principal areas:

1) decision aiding;

2) optimization and its applications;

3) multi-agent and distributed systems;

4) database systems, information systems and knowledge management.

Under such perspective, the laboratory’s scientific project mainly aims to:

1. Laboratoire d’Analyse et Modélisation de Systèmes pour l’Aide à la DEcision (Laboratory of Analysis and Modeling of Decision Support Systems).

– consolidate and extend our international leadership in operations research and decision aiding;

– strengthen and promote our vision of management and decision-oriented computer science;

– create new large interfaces between operations research and theoretical computer science.

In particular, research in the intersection of combinatorial optimization and theoretical computer science always remains a central key-point of LAMSADE's research activity.

Combinatorial optimization and theoretical computer science have been, and still are, considered as two subjects different from each other. If the difference is quite evident for some areas of both subjects, it becomes much less so if we think of areas such as complexity theory, theory of algorithms, solving hard combinatorial problems, graph theory and, more generally, discrete mathematics, etc. All these matters form a very large interface between combinatorial optimization and theoretical computer science. Historically, researchers in the areas mentioned above have been members of two distinct major scientific communities, namely theoretical computer science and operations research. They have addressed almost the same problems, worked under the same paradigms, used parallel formalisms, tools and methods and developed either parallel or complementary approaches. The fruits of this "separate research" have impregnated the entire field of information technology and industry and almost the whole of what is considered today as management sciences. Moreover, they have been widespread over numerous scientific disciplines, initially orthogonal to both computer science and combinatorial optimization, giving rise to new areas of research. However, if from this "separate attack" we witnessed the emergence of practically all of the traditional concepts dealing with complexity theory, discrete mathematical modeling and polynomial approximation of discrete optimization problems, numerous problems and challenges remain open and without satisfying answers, thus the need for intensive research in the interface of combinatorial optimization and theoretical computer science becomes not only clear but also extremely challenging. This kind of research is one of the major directions in the scientific project of the LAMSADE.

With such studies, we expect to advance in the research for new paradigms, getting an insight mainly from the complex system sciences. I strongly believe that in the near future, the themes of our research will be central to operational research and will reshape the research landscape in combinatorial optimization. I also believe that they will influence all the active research for new calculating machine paradigms based upon properties of natural and human systems that are not exploited by conventional computers, by providing them with new problems to deal with and new solutions to try out. Our scientific project can thus be seen as an initiative to drastically renovate the research agenda in combinatorial optimization, by addressing open and novel problems

arising from complex human systems. In order to achieve this objective, we have first to support a research environment that overcomes traditional cluster barriers among communities historically defined as “operations research” and “theoretical computer science”. We have also to work over the common basis of established theories and expertise for studying decidability, complexity, structure and solutions of hard optimization problems, which will definitely serve as the framework for validation of any advances in new research topics.

As stated above, bringing together operations research and theoretical computer science can be the first step in developing close synergies between all the complex systems disciplines, mainly those based upon the study of human systems. Research in the interface of these subjects is the main attempt to build such a broad alliance and to give it a clear scientific status. Moreover, by handling novel problems issued by still unexploited models and working hypotheses, we aim to strongly contribute to the emergence of a new paradigm for both combinatorial optimization, and algorithmic and complexity theory aspects of theoretical computer science.

The main objective of the book is to bear witness to the quality and the depth of the work conducted in the laboratory along the epistemological lines just outlined. In the chapters, the reader will find all the ingredients of a successful matching between combinatorial optimization and theoretical computer science, with interesting results carrying over a large number of their common subjects and going from “pure” complexity theoretical approaches dealing with concepts like **NP**- and **PSPACE**-completeness to “oldies but goodies” and always essential and vital operational research subjects such as flows, scheduling, or linear and mathematical programming, passing from polynomial approximation, online calculation, multicriteria combinatorial optimization, game theory, design of algorithms for multi-agent systems, etc. All of the chapters make a valuable contribution to both the two main topics of the book and any of the areas dealt.

In Chapter 1, Aloulou and Della Croce deal with single machine scheduling. They consider scheduling environments where some job characteristics are uncertain, this uncertainty being modeled through a finite set of well-defined scenarios. They search for a solution that is acceptable for any considered scenario using the “absolute robustness” criterion and present algorithmic and computational complexity results for several single machine scheduling problems.

Although the approximability of multi-criteria combinatorial problems has been the inspiration for numerous articles, the non-approximability of these problems seems to have never been investigated until now. Angel *et al.* in Chapter 2 propose a way to get some results of this kind that work for several problems. Then, they apply their method on a multi-criteria version of the traveling salesman problem in graphs with edge-distances one and two. Furthermore, they extend existing approximation results

for the bi-criteria traveling salesman problem in graphs with edge-weights 1 or 2 to any number k of criteria.

In Chapter 3, Ausiello *et al.* study online models for minimum set cover problem and minimum dominating set problem. For the former problem, the basic model implies that the elements of a ground set of size n arrive one-by-one; we assume that with any such element, arrives also the name of some set containing it and covering most of the still uncovered ground set-elements. For this model they analyze a simple greedy algorithm and prove that its competitive ratio is $O(\sqrt{n})$ and that it is asymptotically optimal for the model dealt. They finally deal with a new way to tackle online problems by using what they call "budget models". For the case of the minimum set cover problem the model considered generates the so-called maximum budget saving problem, where an initial budget is allotted that is destined to cover the cost of an algorithm for solving set-covering and the objective is to maximize the savings on the initial budget.

In Chapter 4 by Bérard *et al.*, Merlin-like time Petri nets (TPN) and timed automata (TA) are considered. The authors investigate questions related to expressiveness for these models: they study the impact of slight variations of semantics on TPN and compare the expressive power of TA and TPN with respect to both time language acceptance and weak time bisimilarity. On the one hand, they prove that TA and bounded TPNs (enlarged with strict constraints) are equivalent w.r.t. timed language equivalence, by providing an efficient construction of a TPN equivalent to a TA. On the other hand, they exhibit a TA such that no TPN (even unbounded) is weakly bisimilar to it. Motivated from this latter result, they characterize the subclass TA^- of TA that is equivalent to the original model of Merlin-like TPN and show that both the associated membership problem and the reachability problem for TA^- are **PSPACE**-complete.

Carello *et al.*, in Chapter 5, introduce a graph problem which is called maximum node clustering. They prove that it is strongly **NP**-hard, but it can be approximated, in polynomial time, within a ratio arbitrarily close to 2. For the special case where the graph is a tree, they prove that the associated decision problem is weakly **NP**-complete as it generalizes the 0-1 knapsack problem and is solvable in pseudo-polynomial time by a dynamic programming approach. For this case they devise a fully polynomial time approximation schema for the original (optimization) problem.

In Chapter 6, Chevaleyre tackles the problem of multi-agent patrolling dealt with as a combinatorial optimization problem. More precisely, territory (one of the inputs of the problem) is modeled by means of a suitable edge-weighted graph $G(V, E)$ and then the exploration strategies for this graph are based upon particular solutions of the traveling salesman problem. With this method, when the graph is metric, he obtains, in polynomial time, an exploration strategy with value bounded above by $3\mathrm{opt}(G) + 4\max\{w(i,j) : (i,j) \in E\}$, where $\mathrm{opt}(G)$ is the value of the optimal exploration strategy and $w(i,j)$ is the weight of the edge $(i,j) \in E$. It is also proved that, using

another approach for the patrolling problem, based on a particular graph-partitioning problem, the multi-agent patrolling problem is approximable within approximation ratio 15, even in the case where the underlying graph is not metric.

In Chapter 7, Chevaleyre *et al.* investigate the properties of an abstract negotiation framework where, on the one hand, agents autonomously negotiate over allocations of discrete resources and, on the other hand, reaching an optimal allocation potentially requires very complex multilateral deals. Therefore, they are interested in identifying classes of utility functions such that, whenever all agents model their preferences using them, any negotiation conducted by means of deals involving only a single resource at a time is bound to converge to an optimal allocation. They show that the class of modular utility functions is not only sufficient (when side-payments are allowed) but is also maximal in this sense. A similar result is proved in the context of negotiation without money.

In Chapter 8, Della Croce *et al.* study two very well-known hard combinatorial problems, the maximum cut problem and the minimum dominating set restricted to graphs of maximum degree 3 (minimum 3-dominating set). For the former, they mainly focus on sparse graphs, i.e., on graphs having bounded maximum degree. They first use a technique based upon enumeration of cuts in a properly chosen subgraph of the input graph and then an extension of them in an optimal way to produce a cut for the whole instance. By means of this method they produce an exact algorithm for the weighted maximum cut problem with improved upper complexity bound in the case of sparse graphs. Next, they restrict themselves to the unweighted maximum cut problem in graphs of maximum degree 3 and devise a tree-search based exact algorithm. Exploiting some simple and intuitive dominance conditions that efficiently prune the search-tree, they provide a fairly competitive upper complexity bound for the case settled. Finally, they refine the search tree's pruning by introducing a counting procedure, based upon the introduction of weights for the fixed data, which allows them to measure in a more precise way the progress made by the algorithm when it fixes them. They apply this method to min 3-dominating set.

In Chapter 9, Demange *et al.* study the computational complexity of online shunting problems. They consider a depot consisting of a set of parallel tracks. Each track can be approached from one side only and the number of trains per track is limited. The departure times of the trains are fixed according to a given timetable. The problem is to assign a track to each train as soon as it arrives to the depot and such that it can leave the depot on time without being blocked by any other train. They show how to solve this problem as an online bounded coloring problem on special graph classes. They also study the competitiveness of the first fit algorithm and show that it matches the competitive ratio of the problem.

Chapter 10, by Demange *et al.*, surveys complexity and approximation results for the minimum weighted vertex coloring problem. This is a natural generalization of the

traditional minimum graph coloring problem obtained by assigning a strictly positive integer weight for any vertex of the input graph, and defining the weight of a color (independent set) as the maximum of the weights of its vertices. Then, the objective is to determine vertex coloring for the input graph minimizing the sum of the weights of the colors used. Complexity and approximation issues for this problem are presented for both general graphs and for graphs where the traditional minimum graph coloring problem is polynomial.

Chapter 11 is a complement of Chapter 10 where, along the same lines, complexity and approximation issues are addressed for the minimum weighted edge coloring problem where, instead of vertices, edges are now to be legally colored.

In Chapter 12, Gabrel considers the Dantzig-Wolfe decomposition for 0-1 linear programming when a subset of constraints defines a independent set polytope. She compares linear relaxations of both the initial and master program (obtained by decomposing on independent set constraints) with respect to various independent set polytope representations. For perfect graphs (in particular for co-comparability graphs), the linear relaxation of the master program is easy to solve while for general graphs its optimal value cannot be calculated in polynomial time. Consequently, she proposes to decompose only on a subset of the independent set constraints (those associated with "polynomial" independent set problems) in order to define another master program for which the LP-relaxation is easy to solve and remains stronger than the traditional LP-relaxation of the initial program.

In Chapter 13, Gabrel compares several 0-1 linear programs for solving the satellite mission planning problem. She considers two models and explains why one of them systematically calculates lower upper bounds. Her explanation is based upon independent set polytope formulations for perfect graphs. Then, she proposes new upper bounds for some large-size benchmark instances.

Chapter 14, by Giannakos *et al.*, is a survey on some of the main results dealing with the problem of finding a Nash equilibrium in a game. After reporting several questions concerning complexity of general games (how many equilibria exist?, what are the conditions of the existence of an equilibrium verifying some given property?), the authors focus on games having pure Nash equilibria, as potential games and congestion games, for which they present several models.

In Chapter 15, entitled "Flows!", Koskas and Murat give another novel interface between operational research and theoretical computer science by showing how tools from combinatorics of words can be very efficiently used in order to devise "divide and conquer" algorithms in a number of operational research and computer science fields, like database management, automatic translation, image pattern recognition, flow or shortest path problems, etc. The current contribution details one of them, dealing with maximum flow in a network.

Milanič and Monnot, in Chapter 16, introduce the exact weighted independent set problem, consisting of determining whether a weighted graph contains an independent set of a given weight. They determine the complexity of this problem as well as the complexity of its restricted version, where the independent set is required to be of maximum size, for several graph-classes. Furthermore, they show that these problems can be solved in pseudo-polynomial time for chordal graphs, AT-free graphs, distance-hereditary graphs, circle graphs, graphs of bounded clique-width, and several subclasses of P_5-free and fork-free graphs. Monnot, in Chapter 17, deals with complexity and approximability of the labeled perfect matching problem in bipartite graphs, as well as with minimum labeled matching and maximum labeled matching in 2-regular bipartite graphs, i.e., in collections of pairwise disjoint cycles of even length.

In Chapter 18, Monnot and Toulouse present several standard- and differential-approximation results for the P_4-partition problem for both minimization and maximization versions.

Finally, in Chapter 19, Quadri *et al.* present an improvement of a well-known method, based upon surrogate relaxation and linearization of the objective function, for calculating an upper bound of integer separable quadratic multi-knapsack and report computational experiments that seem to confirm the efficiency of their approach.

I think that all these contributions show the vitality and the originality of the research carried out by the LAMSADE. I do hope that the reader will really appreciate the depth and the richness of all the presented contributions.

To conclude, let me say once more that it is always a pleasure for me to work with Chantal, Sami and Raphael Menasce, Jon Lloyd and their colleagues at ISTE Ltd.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 1

The Complexity of Single Machine Scheduling Problems under Scenario-based Uncertainty

1.1. Introduction

This chapter deals with single machine scheduling problems where some job characteristics are uncertain. This uncertainty is described through a finite set S of well-defined scenarios. We denote by p_j^s, d_j^s and w_j^s, respectively, the processing time, the release date, the due date and the weight of job j under scenario $s \in S$.

Consider a scheduling problem, denoted by $\alpha|\beta|\gamma$ according to Graham *et al.* notation [GRA 79]. Let Π be the set of feasible schedules with respect to the problem constraints. For each scenario $s \in S$, we denote by OPT$(\alpha|\beta|\gamma, s)$ the problem of finding an optimal schedule π_s^* satisfying:

$$F(\pi_s^*, s) = \min_{\pi \in \Pi} F(\pi, s). \qquad [1.1]$$

When problem parameters are uncertain, it is appropriate to search for a solution that is acceptable for any considered scenario. For this purpose, several criteria can be applied in order to select among solutions. In [KOU 97], Kouvelis and Yu proposed three different robustness criteria: the absolute robustness or maximal cost, the maximal regret or robust deviation and the relative robustness. In this chapter, we focus on the *absolute robustness* criterion.

Chapter written by Mohamed Ali ALOULOU and Federico DELLA CROCE.

To the best of our knowledge, the absolute robustness in single machine scheduling problems has only been considered in [DAN 95] and [YAN 02] where, for the $1||\sum C_j$ problem with uncertain processing times, two distinct proofs of the **NP**-hardness even for $|S| = 2$ were provided (notice that the corresponding deterministic version is well known to be polynomially solvable [SMI 56]). The maximal regret criterion was instead much more studied (see, for instance, [YAN 02] but also [AVE 00, KAS 05, KOU 00]).

The absolute robustness of schedule π over all scenarios $s \in S$ is denoted by $\bar{F}(\pi)$. We have:

$$\bar{F}(\pi) = \max_{s \in S} F(\pi, s) \qquad [1.2]$$

We denote by $MinMax(\alpha|\beta|\gamma, \theta)$ the problem of finding a schedule π^A minimizing the absolute robustness $\bar{F}(\pi)$ among all schedules $\pi \in \Pi$. Field θ indicates the set of uncertain problem parameters. For the problems considered here, $\theta \subseteq \{p_j, d_j, w_j\}$. Sequence π^A is called in [KOU 97] *absolute robust sequence*. Its cost $\bar{F}(\pi^A)$ satisfies:

$$\bar{F}(\pi^A) = \min_{\pi \in \Pi} \bar{F}(\pi) = \min_{\pi \in \Pi} \max_{s \in S} F(\pi, s) \qquad [1.3]$$

Notice that if problem $\alpha|\beta|\gamma$ is **NP**-hard, then, in the presence of uncertainty, the corresponding problems $MinMax(\alpha|\beta|\gamma, \theta)$ are also **NP**-hard. However, if problem $\alpha|\beta|\gamma$ is polynomially solvable, then, the corresponding problems $MinMax(\alpha|\beta|\gamma, \theta)$ are not necessarily polynomially solvable. In this chapter we establish the complexity status for the absolute robustness versions of the most well known non-preemptive polynomial-time single machine scheduling problems, namely problems $1|prec|f_{\max}$ (with $f_{\max} \in \{C_{\max}, L_{\max}, T_{\max}\}$) , $1||\sum w_j C_j$ and $1||\sum U_j$. Notice that all these problems present regular cost functions which are non-decreasing in the job completion times. In this context any schedule $\pi \in \Pi$ is completely characterized by the corresponding job sequence. Given a schedule $\pi \in \Pi$, the completion time of job j under scenario s, denoted by $C_j(\pi, s)$, $j = 1, \ldots, n, s \in S$, can easily be determined and the quality of the schedule $\pi \in \Pi$ under scenario s is then evaluated using the regular cost function $F(\pi, s)$. We consider the following cost functions:

– the general maximum cost function $f_{\max} = \max_j\{f_j(C_j)\}$ with $f_{\max} \in \{C_{\max}, L_{\max}, T_{\max}\}$: hence we deal with the maximum completion time (makespan) $C_{\max} = \max_j\{C_j\}$, the maximum lateness $L_{\max} = \max_j\{L_j\}$ with $L_j = C_j - d_j$ and the maximum tardiness $T_{\max} = \max_j\{T_j\}$ with $T_j = \max\{0, L_j\}$;

– the total weighted completion time $\sum_j w_j C_j$;

– the number of late jobs $\sum_j U_j$ with $U_j = 0$, if job j is *on-time* ($C_j \leqslant d_j$) and $U_j = 1$ if j is *late* ($C_j > d_j$)

Using the set S of scenarios, we construct a scenario s^w in which parameters k_j take their worst case value, denoted by k_j^w. In our case, we have $p_j^w = \max_{s\in S} p_j^s$, $d_j^w = \min_{s\in S} d_j^s$ and $w_j^w = \max_{s\in S} w_j^s$. Notice that in the context of a discrete set of scenarios, the constructed scenario is not necessarily feasible, i.e. we can have $s^w \notin S$: s^w is called *worst-case artificial scenario*.

REMARK 1.1.– When parameters are interval-uncertain, s^w is a feasible scenario. In this case, an absolute robust solution π^A of problem $MinMax(1|\beta|\gamma, \theta)$ is such that:

$$\bar{F}(\pi^A) = \min_{\pi\in\Pi} \bar{F}(\pi) = \min_{\pi\in\Pi} \max_{s\in S} F(\pi, s) \quad [1.4]$$

$$= \min_{\pi\in\Pi} F(\pi, s^w) \quad [1.5]$$

Hence, π^A is also optimal for problem OPT$(1|\beta|\gamma, s^w)$. This means that the problem of finding an absolute robust sequence can be solved straightforwardly by the algorithm solving the problem without uncertainty applied to the worst-case artificial scenario.

When uncertainty is scenario-based, we cannot apply the same reasoning because scenario s^w is not necessarily feasible. Nevertheless, we show in this chapter that problem $MinMax(1|prec|L_{max}, d_j)$ can be solved by Lawler's algorithm applied to the worst case artificial scenario. We also prove that an extension of Lawler's algorithm, called here *MinMax-Lawler*, solves problem $MinMax(1|prec|f_{max}, p_j, d_j)$ in polynomial time. On the other hand, problems $MinMax(1||\sum w_j C_j, w_j)$ and $MinMax(1||\sum U_j, p_j)$ are proved to be **NP**-hard even when $|S| = 2$. However, problem $MinMax(1||\sum U_j, d_j)$ is still open.

Table 1.1 summarizes the above results presenting the complexity status for the absolute robustness versions of the most well known non-preemptive polynomial-time single machine scheduling problems, where an entry "-" indicates that the considered case does not apply (for instance, problem $1||\sum w_j C_j$ cannot have uncertainty on due dates as due dates are not present in the problem).

1.2. Problem $MinMax(1|prec|f_{\max}, \theta)$

1.2.1. *Uncertainty on due dates*

We consider problem $MinMax(1|prec|f_{\max}, d_j)$ where processing times are deterministic and due dates are uncertain (here $f_{\max} \in \{L_{\max}, T_{\max}\}$ as for $C_{\max}$ no

Uncertain parameter	Deterministic problem		
	$1\|\|\sum w_jC_j$	$1\|prec\|f_{\max}$	$1\|\|\sum U_j$
d_j	–	$O(n^2+n\|S\|)$ (Theor. 1.1)	open
p_j	**NP**-hard (see [DAN 95, YAN 02])	$O(n^2\|S\|)$ (Corol. 1.2)	**NP**-hard (Theor. 1.4)
w_j	**NP**-hard (Theor. 1.3)	–	–
p_j and d_j	–	$O(n^2\|S\|)$ (Theor. 1.2)	**NP**-hard (Corol. 1.3)
p_j and w_j	**NP**-hard (see [DAN 95, YAN 02])	–	–

Table 1.1. *Summary of the obtained results*

uncertainty holds). In this case the worst-case artificial scenario s^w is such that, for all $j \in N$, $d_j^w = \min_{s\in S} d_j^s$.

We recall that problem $1|prec|f_{\max}$ can be solved in $O(n^2)$ time by Lawler's algorithm [LAW 73]. This algorithm constructs an optimal schedule backwards. At the points in time where the unscheduled jobs should be completed, starting with point $t = P = \sum_{j\in N} p_j$, Lawler's algorithm chooses among the unscheduled jobs having no successors a job with minimum cost to be completed at t. Notice that as processing times are deterministic, we have:

$$\forall \pi \in \Pi, \forall s \in S, \forall j \in N, C_j(\pi, s) = C_j(\pi) \quad [1.6]$$

The following theorem holds.

THEOREM 1.1.– *Problem $MinMax(1|prec|f_{\max}, d_j)$ can be optimally solved in $O(n^2+n|S|)$ time by means of Lawler's algorithm applied to the worst-case artificial scenario s^w.*

Proof. For the sake of clarity we consider that $f_{\max} = L_{\max}$, but the same analysis holds for $f_{\max} = T_{\max}$. An absolute robust solution π^A of problem $MinMax(1|prec|L_{\max}, d_j)$ is such that:

$$\bar{L}(\pi^A) = \min_{\pi\in\Pi} \max_{s\in S} L_{\max}(\pi, s) \quad [1.7]$$

$$= \min_{\pi\in\Pi} \max_{s\in S} \max_{j\in N} (C_j(\pi, s) - d_j^s) \quad [1.8]$$

$$= \min_{\pi\in\Pi} \max_{s\in S} \max_{j\in N} (C_j(\pi) - d_j^s) \quad [1.9]$$

$$= \min_{\pi\in\Pi} \max_{j\in N} \max_{s\in S} (C_j(\pi) - d_j^s) \quad [1.10]$$

$$= \min_{\pi\in\Pi} \max_{j\in N} (C_j(\pi) - d_j^w) \quad [1.11]$$

$$\bar{L}(\pi^A) = \min_{\pi\in\Pi} L_{\max}(\pi, s^w) \quad [1.12]$$

Hence, π^A is also an optimal solution for problem OPT$(1|prec|L_{\max}, s^w)$. For the complexity, the construction of the worst-case scenario requires $O(n|S|)$ time and the application of Lawler's algorithm requires $O(n^2)$ time, hence the overall complexity is $O(n^2 + n|S|)$. ∎

We observe that the proof of Theorem 1.1 can be applied, as it is, to any scheduling problem $\alpha|\beta|f_{\max}$. Hence, we have the following result.

COROLLARY 1.1.– Any algorithm optimally solving problem $\alpha|\beta|f_{\max}$ provides an absolute robust solution for problem $MinMax(\alpha|\beta|f_{\max}, d_j)$, when applied to the worst-case artificial scenario s^w.

1.2.2. *Uncertainty on processing times and due dates*

We consider problem $MinMax(1|prec|f_{\max}, p_j, d_j)$ where we suppose now that both processing times and due dates are uncertain. A robust solution π^A is such that:

$$\bar{F}(\pi^A) = \min_{\pi\in\Pi} \max_{s\in S} F_{\max}(\pi, s) \quad [1.13]$$

$$= \min_{\pi\in\Pi} \max_{s\in S} \max_{j\in N} f_j(C_j(\pi, s)) \quad [1.14]$$

We propose an algorithm, called *MinMax-Lawler*, which is an extension of Lawler's algorithm. This algorithm constructs a sequence π in reverse order. Let U be the set of unscheduled jobs. Define $p^s(U) = \sum_{j\in U} p_j^s$ for all $s \in S$. The rule is the following: schedule last the job $j \in U$, which has no successor in U and such that $\max_{s\in S} f_j^s(p^s(U))$ is minimal. It is immediately clear that the complexity of *MinMax-Lawler* is $O(n^2|S|)$.

We have the following result.

THEOREM 1.2.– *Problem $MinMax(1|prec|f_{\max}, p_j, d_j)$ is optimally solved by algorithm MinMax-Lawler.*

Proof. The proof is very similar to the proof of Lawler's algorithm optimality for problem $1|prec|f_{\max}$.

Enumerate the jobs in such a way that $(1, 2, \ldots, n)$ is the sequence constructed by the proposed algorithm. Let π^A be an absolute robust sequence for problem $MinMax(1|prec|f_{\max}, p_j, d_j)$ with $\pi^A(i) = i$ for $i = n, n-1, \ldots, r$ and $\pi^A(r) = j < r$.

Notice that it is possible to schedule $r-1$ immediately before r. Hence, we can construct a sequence π' in which we shift to the left the block between jobs $r-1$ and r and process $r-1$ immediately before r. Clearly:

$$\forall i \in N - \{r-1\}, \forall s \in S, C_i(\pi', s) \leqslant C_i(\pi^A, s) \quad [1.15]$$

Hence:

$$\begin{aligned}
\bar{F}(\pi') = \max_{s \in S} F(\pi', s) &= \max\{\bar{F}(\pi^A), \max_{s \in S} f^s_{r-1}(C_{r-1}(\pi', s))\} && [1.16]\\
&= \max\{\bar{F}(\pi^A), \max_{s \in S} f^s_{r-1}(C_j(\pi^A, s))\} && [1.17]\\
&\leqslant \max\{\bar{F}(\pi^A), \max_{s \in S} f^s_j(C_j(\pi^A, s))\} && [1.18]\\
&\leqslant \bar{F}(\pi^A) && [1.19]
\end{aligned}$$

Consequently, π' is also an absolute robust sequence.

We can reiterate the same reasoning and transform sequence π^A into sequence $(1, 2, ..., n)$ without increasing the objective function value. ∎

Correspondingly, the following corollary also holds.

COROLLARY 1.2.– Problem $MinMax(1|prec|f_{\max}, p_j)$ is optimally solved by algorithm *MinMax-Lawler*.

1.3. Problem $MinMax(1||\sum w_j C_j, w_j)$

We consider problem $MinMax(1||\sum w_j C_j, w_j)$ where processing times are deterministic and weights are uncertain. We prove that this problem is **NP**-hard even when $|S| = 2$ and $p_j = 1 \ \ \forall j$. To this extent we need to prove the following instrumental lemma.

LEMMA 1.1.– *The* $1||\sum C_j$ *problem and the* $1|p_j = 1|\sum w_j C_j$ *problem are equivalent.*

Proof. Given any instance of the $1||\sum C_j$ problem where each job j has processing time p'_j, generate an instance of the $1|p_j = 1|\sum w_j C_j$ problem where each job j has weight $w''_j = p'_{n-j+1}$. Consider a generic sequence $(1, 2, ..., n-1, n)$. For the

$1||\sum C_j$ problem the corresponding cost function value is $Z_1 = \sum_{j=1}^{n}(n-j+1)p'_j$. For the $1|p_j = 1|\sum w_j C_j$ problem the corresponding cost function value is $Z_2 = \sum_{j=1}^{n} jw''_j$. We show that $Z_2 = Z_1$. Indeed, $Z_2 = \sum_{j=1}^{n} jw''_j = \sum_{j=1}^{n} jp'_{n-j+1} = \sum_{j=1}^{n}(n-j+1)p'_j = Z_1$. ∎

THEOREM 1.3.– *Problem $MinMax(1||\sum w_j C_j, w_j)$ is **NP**-hard even when $|S| = 2$ and $p_j = 1 \ \forall j$.*

Proof. Due to Lemma 1.1 and the **NP**-hardness of problem $MinMax(1||\sum C_j, p_j)$ from [DAN 95, YAN 02], the proof immediately holds. ∎

1.4. Problem $MinMax(1||\sum U_j, \theta)$

We consider problem $MinMax(1||\sum U_j, \theta)$ with $\theta \subseteq \{p_j, d_j\}$. For problem $MinMax(1||\sum U_j, d_j)$, we did not find any result. We only show that it cannot be solved by the well-known Moore's algorithm [MOO 68] applied to the worst-case artificial scenario. However, we prove that problem $MinMax(1||\sum U_j, p_j)$ is **NP**-hard.

1.4.1. *Uncertainty on due dates*

Consider the following example: two jobs 1 and 2 such that $p_1 = 2$, $p_2 = 3$, $d_1^1 = 1$, $d_1^2 = 10$, $d_2^1 = 4$ and $d_2^2 = 2$.

Consider sequence $(1, 2)$. Both jobs are late for scenario 1 and job 1 is on-time for scenario 2. The maximum cost is 2. For sequence $(2, 1)$, there is one on-time job for each scenario: job 1 is on-time in scenario 2 and job 2 is on-time for scenario 1. The maximum cost is equal to 1. Hence, $(2, 1)$ is the optimal solution.

In the worst case scenario, the jobs are both late in any solution. Hence, Moore's algorithm applied to the worst case scenario would sequence 1 and 2 in any order, which is not always optimal. Hence, it does not solve the considered problem.

1.4.2. *Uncertainty on processing times*

We prove that problem $MinMax(1||\sum U_j, p_j)$ is **NP**-hard even when $|S| = 2$ by constructing a reduction from the **NP**-complete even-odd partition problem [GAR 79].

Even-odd partition problem

Given $2n$ positive integers $a_1 < a_2 < \ldots < a_{2n}$ where $\sum_{j=1}^{2n} a_j = 2B$, is there a partition of the integers into two subsets A_1 and A_2 such that:

$$\sum_{j \in A_1} a_j = \sum_{j \in A_2} a_j = B \qquad [1.20]$$

and A_1 and A_2 each contains exactly one element of each pair (a_{2i-1}, a_{2i}), $i = 1, \ldots, n$?

Given an instance of the even-odd partition problem, we construct the following instance of problem $MinMax(1||\sum U_j, p_j)$.

Instance of problem $MinMax(1||\sum U_j, p_j)$

We have $2n$ jobs and two scenarios 1 and 2. Job processing times and due dates are such that:

$$\begin{cases} p_1^1 = p_2^2 = a_1 \\ p_1^2 = p_2^1 = a_2 \\ d_1 = d_2 = a_2 \end{cases} \qquad [1.21]$$

$$\forall i = 2, ..., n \begin{cases} p_{2i-1}^1 = p_{2i}^2 = a_{2i-1} + a_{2i-2} + \ldots + a_2 \\ p_{2i-1}^2 = p_{2i}^1 = a_{2i} + a_{2i-2} + \ldots + a_2 \\ d_{2i-1} = d_{2i} = a_{2i} + 2a_{2i-2} + \ldots + ia_2, i < n \\ d_{2n} = d_{2n-1} = B + d_{2n-2} \end{cases} \qquad [1.22]$$

Jobs $2i - 1$ and $2i$ are called $a_{[i]} - jobs$. Notice that with this construction, the following lemmata hold.

LEMMA 1.2.– *Any sequence in which each position $i, i = 1, \ldots, n$, is occupied by an $a_{[i]} - job$ admits at least $n - 1$ on-time jobs.*

Proof. We can prove straightforwardly that in such a sequence the first $n - 1$ jobs are on-time. ∎

LEMMA 1.3.– *If two jobs $2i$ and $2i-1$ are both on-time, then there exists $j, 1 \leqslant j < i$, such that jobs $2j$ and $2j - 1$ are both late.*

Proof. Suppose by contradiction that there exists a sequence in which for all $j, 0 \leqslant j < i$, at least one job $2j$ or $2j - 1$, denoted $a_{[j]}$-job, is on time and that jobs $2i$ and $2i - 1$ are both on-time.

It is clear that resequencing these on-time jobs in a non-decreasing order of their due dates does not increase the number of late jobs.

In this new sequence, jobs $2i$ and $2i-1$ are sequenced last with respect to the considered on-time jobs. The completion time of the last job ($2i$ or $2i-1$) is greater than or equal to the sum of:

1) the sum of the minimal processing times of $a_{[j]}$-jobs scheduled before jobs $2i$ and $2i-1$, $0 \leqslant j < i$, denoted by A;

2) the processing times of jobs $2i$ and $2i-1$, denoted by B.

We have:

$$\begin{aligned} A &= \begin{cases} a_1+ \\ a_3+a_2+ \\ a_5+a_4+a_2+ \\ \ldots \\ a_{2i-3}+a_{2i-4}+\cdots+a_4+a_2 \end{cases} \\ &= a_1+a_3+\cdots+a_{2i-3}+a_{2i-4}+\cdots+(i-2)*a_2 \end{aligned} \quad [1.23]$$

and

$$B = p^1_{2i}+p^1_{2i-1} = p^2_{2i}+p^2_{2i} \quad [1.24]$$

$$= a_{2i}+a_{2i-1}+2*a_{2i-2}+\cdots+2*a_4+2*a_2 \quad [1.25]$$

The completion time of the last job is greater than or equal to:

$$A+B = a_1+a_3+\cdots+a_{2i-3}+a_{2i}+2a_{2i-2}+\ldots+ia_2 \quad [1.26]$$

$$= a_1+a_3+\cdots+a_{2i-3}+d_{2i} \quad [1.27]$$

$$> d_{2i} \quad [1.28]$$

Hence at least one job among jobs $2i$ and $2i-1$ is late. This contradicts the considered hypothesis. ∎

LEMMA 1.4.– *Every schedule admits at most n on-time jobs.*

Proof. Suppose by contradiction that there exists a sequence such that the number of on-time jobs is greater than or equal to $n+1$. Resequencing the on-time jobs in a

non-decreasing order of their due dates and sequencing late jobs last does not increase the number of late jobs. Let π be such a sequence.

Since we have $n+1$ on-time jobs, then there exists at least one integer i such that jobs $2i$ and $2i-1$ are both on-time. Let i^* be the smallest i such that $2i$ and $2i-1$ are both on-time and π_{i^*} be the subsequence of on-time jobs among jobs $k, 1 \leqslant k \leqslant 2i^*$. We have $\pi = (\pi_{i^*}, \pi_R)$. Notice that in subsequence π_{i^*} there are at most i^* jobs, which are all on-time.

Due to Lemma 1.3, there exists an integer $j, 1 \leqslant j < i^*$, such that neither job $2j$ nor job $2j-1$ is on-time. We construct a new subsequence sequence π'_{i^*} in which we exchange one job among $2i^*$ and $2i^*-1$ (which are on-time in π) with job $2j$ or $2j-1$ (which are late in π). For example, we exchange jobs $2i^*$ and $2j-1$. Then, we reorder the jobs of π'_{i^*} in a non-decreasing order of their due dates. Due to Lemma 1.2, all jobs of π'_{i^*} are on-time. Since $\forall s \in S, p^s_{2j-1} < p^s_{2i^*}$, then sequence (π'_{i^*}, π_R) has at least the same number of jobs on-time as π.

We can reiterate the same reasoning until we get a sequence in which we have at most $n-1$ jobs on-time among jobs $1, \ldots, 2n-2$. Since we have at least $n+1$ jobs on-time, then jobs $2n$ and $2n-1$ are both on-time but this cannot occur due to Lemma 1.3. ∎

The following theorem holds.

THEOREM 1.4.– *Problem MinMax($1||\sum U_j, p_j$) is **NP**-hard even when $|S| = 2$.*

Proof. We first prove that, if the even-odd partition problem has a solution, then the constructed instance admits a solution with n on-time jobs.

Let A_1 be a solution of the even-odd partition problem. Consider a sequence $\pi = (\pi_1, \pi_2)$ such that:

– for all $j \leqslant 2n$, if $a_j \in A_1$ then job $j \in \pi_1$;

– the jobs of π_1 are ordered in non-decreasing order of their due dates.

Suppose that $\pi = ([1], [2], \ldots, [n], [n+1], [n+2], \ldots [2n])$ and $\pi_1 = ([1], [2], \ldots, [n])$

Due to Lemma 1.2, the first $n-1$ jobs in π_1 are on-time. We prove that the last job in π_1 is also on-time.

The completion time of the last job $[n]$ in scenario s_1 is:

$$\begin{aligned} C_{[n]}(\pi, s_1) &= p^1_{[1]} + p^1_{[2]} + \cdots p^1_{[n]} \\ &= \begin{cases} a_{[1]}+ \\ a_{[2]} + a_2+ \\ \ldots \\ a_{[n]} + a_{2n-2} + \cdots + a_4 + a_2 \end{cases} \\ &= a_{[1]} + a_{[2]} + \cdots + a_{[n]} + a_{2n-2} \\ &\quad + \cdots + (n-1) * a_2 \\ &= B + d_{2n-2} \\ &= d_{2n} = d_{2n-1} = d_{[n]} \end{aligned} \qquad [1.29]$$

The completion time of the last job $[n]$ in scenario s_2 is:

$$\begin{aligned} C_{[n]}(\pi, s_2) &= p^2_{[1]} + p^2_{[2]} + \cdots p^2_{[n]} \\ &= \begin{cases} a_{[n+1]}+ \\ a_{[n+2]} + a_2+ \\ \ldots \\ a_{[2n]} + a_{2n-2} + \cdots + a_4 + a_2 \end{cases} \\ &= a_{[n+1]} + a_{[n+2]} + \cdots + a_{[2n]} + a_{2n-2} \\ &\quad + \cdots + (n-1) * a_2 \\ &= B + d_{2n-2} \\ &= d_{2n} = d_{2n-1} = d_{[n]} \end{aligned} \qquad [1.30]$$

Consequently, job $[n]$ is on-time for the two scenarios and the number of on-time jobs in π is at least n and cannot be greater than n due to Lemma 1.4.

We prove now that if the constructed instance admits a solution with n on-time jobs, then the even-odd partition problem admits a solution.

Let π be a sequence in which the number of on-time jobs is equal to n. Suppose that in π, there exist two on-time jobs $2i$ and $2i-1$. Then, due to Lemma 1.3, there exists an index j, $1 \leqslant j < i$, such that jobs $2j$ and $2j-1$ are both late. Using the same technique as in Lemma 1.4, we transform sequence π in a new sequence in which job $2j$ or $2j-1$ is on-time and job $2i$ or $2i-1$ is late. We reiterate the same reasoning until we get a sequence π' in which, for all $k, 1 \leqslant k \leqslant n$, only one job among jobs $2k$ and $2k-1$ is on-time.

Resequencing the on-time jobs in a non-decreasing order of their due dates and sequencing the late jobs last we do not increase the number of late jobs. We have $\pi' = ([1], [2], \ldots, [n], [n+1], [n+2], \ldots [2n])$ and jobs $[i], 1 \leqslant i \leqslant n$ are on-time.

The completion time of job $[n]$ in scenario s_1 is:

$$\begin{aligned} C_{[n]}(\pi', s_1) &= p^1_{[1]} + p^1_{[2]} + \cdots p^1_{[n]} && [1.31] \\ &= a_{[1]} + a_{[2]} + \cdots + a_{[n]} + a_{2n-2} \\ &\quad + \cdots + (n-1) * a_2 && [1.32] \\ &= a_{[1]} + a_{[2]} + \cdots + a_{[n]} + d_{2n-2} && [1.33] \end{aligned}$$

The completion time of job $[n]$ in scenario s_2 is:

$$\begin{aligned} C_{[n]}(\pi', s_2) &= p^2_{[1]} + p^2_{[2]} + \cdots p^2_{[n]} && [1.34] \\ &= a_{[n+1]} + a_{[n+2]} + \cdots + a_{[2n]} + a_{2n-2} && [1.35] \\ &\quad + \cdots + (n-1) * a_2 && [1.36] \\ &= a_{[n+1]} + a_{[n+2]} + \cdots + a_{[2n]} + d_{2n-2} && [1.37] \end{aligned}$$

Since job $[n]$ is on-time then:

$$\forall s \in \{s_1, s_2\}, C_{[n]}(\pi', s) \leqslant d_{[n]} = d_{2n} = d_{2n-1} = B + d_{2n-2} \quad [1.38]$$

Consequently, we have:

$$\left\{ \begin{array}{rcl} a_{[1]} + a_{[2]} + \cdots + a_{[n]} & \leqslant & B \\ a_{[n+1]} + a_{[n+2]} + \cdots + a_{[2n]} & \leqslant & B \\ a_{[1]} + a_{[2]} + \cdots + a_{[2n]} & = & B \end{array} \right. \quad [1.39]$$

This proves that:

$$a_{[1]} + a_{[2]} + \cdots + a_{[n]} = a_{[n+1]} + a_{[n+2]} + \cdots + a_{[2n]} = B \quad [1.40]$$

which means that the even-odd partition problem has a solution. ∎

Correspondingly, the following corollary also holds.

COROLLARY 1.3.– MinMax(1$||\sum U_j, p_j, d_j$) is **NP**-hard even when $|S| = 2$.

1.5. Bibliography

[AVE 00] AVERBAKH I., "Minmax regret solutions for minimax optimization problems with uncertainty", *Operations Research Letters*, vol. 27, p. 57–65, 2000.

[DAN 95] DANIELS R., KOUVELIS P., "Robust scheduling to hedge against processing time uncertainty in single stage production", *Management Science*, vol. 41, p. 363–376, 1995.

[GAR 79] GAREY M., JOHNSON D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, 1979.

[GRA 79] GRAHAM R., LAWLER E., LENSTRA J., KAN A. R., "Optimization and approximation in deterministic machine scheduling: a survey", *Annals of Discrete Mathematics*, vol. 5, p. 287–326, 1979.

[KAS 05] KASPERSKI A., "Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion", *Operations Research Letters*, vol. 33, p. 431–436, 2005.

[KOU 97] KOUVELIS P., YU G., *Robust Discrete Optimisation and its Applications*, Kluwer Academic Publisher, 1997.

[KOU 00] KOUVELIS P., DANIELS R., VAIRAKTARAKIS G., "Robust scheduling of a two-machine flow shop with uncertain processing times", *IIE Transactions*, vol. 32, p. 421–432, 2000.

[LAW 73] LAWLER E., "Optimal sequencing of a single machine subject to precedence constraints", *Management Science*, vol. 19, p. 544–546, 1973.

[MOO 68] MOORE J., "A n job, one machine sequencing algorithm for minimizing the number of late jobs", *Management Science*, vol. 15, p. 102–109, 1968.

[SMI 56] SMITH W., "Various optimizers for single-stage production", *Naval Research Logistics Quarterly*, vol. 3, p. 59–66, 1956.

[YAN 02] YANG J., YU G., "On the Robust Single Machine Scheduling Problem", *Journal of Combinatorial Optimization*, vol. 6, p. 17-33, 2002.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 2

Approximation of Multi-criteria Min and Max $TSP(1,2)$

2.1. Introduction

This chapter presents some recent results obtained by the authors ([ANG 04, ANG 05, ANG 05b]) about the inapproximation and the approximation properties of the multi-criteria traveling salesman problem with distances one and two. These results concern the minimization version of the problem. We also present some new results for the maximization variant.

2.1.1. *The traveling salesman problem*

The traveling salesman problem (TSP) is one of the most studied problems in the operations research community (see for instance [JOH 85]). Given a complete graph where the edges are associated with a positive distance, we search for a cycle visiting each vertex of the graph exactly once and minimizing the total distance. It is well known that the TSP problem is **NP**-hard, and that it cannot be approximated within a constant approximation ratio, unless ***P*=*NP***. However, for the metric TSP (i.e., when the distances satisfy the triangle inequality), Christofides proposed an algorithm with performance ratio 3/2 [CHR 76]. For more than 25 years, many researchers attempted to improve this bound but with no success. In [PAP 93], the authors studied a more restrictive version of the metric TSP, in which all distances are either one or two (denoted by $TSP(1,2)$), and they proposed a 7/6 approximation algorithm. Recently,

Chapter written by Eric ANGEL, Evripidis BAMPIS, Laurent GOURVÈS and Jérôme MONNOT.

Berman and Karpinski proposed a $8/7$-approximation algorithm for the $TSP(1,2)$ problem [BER 06] and some inapproximability results are presented in [ENG 01]. The $TSP(1,2)$ problem is a generalization of the Hamiltonian cycle problem since we are asking for the tour of the graph that contains the fewest possible non-edges (edges of distance 2). In [MON 02, MON 03], some results for the $TSP(1,2)$ with respect to the *differential* approximation ratio are obtained. Finally, there is the maximization version of the problem, denoted by Max TSP, where the goal is to find a tour maximizing the total distance. Max TSP is also known to be **APX**-complete, even if the edge distances are either 1 or 2 (denoted by Max $TSP(1,2)$), [PAP 93] and it is approximable with expected performance ratio $25/33 - \varepsilon$ for all $\varepsilon > 0$ in the general case [HAS 00], and $7/8$ in the metric case [HAS 02].

In this chapter, we deal with a multi-criteria version of the problem: the k-criteria $TSP(1,2)$. The distance between any pair of vertices is a vector of length k instead of a scalar.

2.1.2. *Multi-criteria optimization*

Multi-criteria optimization refers to problems with two or more objective functions which are in conflict. Vilfredo Pareto introduced in 1896 a concept –known today as *Pareto optimality*– that constitutes the origin of the research in this area. According to this concept, the goal in a multi-criteria optimization problem is normally not a single solution, but instead a set of *non-dominated* solutions, the so-called *Pareto curve*. From a calculational point of view, the notion of Pareto curve is problematic. Two of the main reasons are:

– the size of a Pareto curve which is often exponential with respect to the size of the corresponding instance, [PAP 00, VAS 05];

– calculating one Pareto optimal solution of a multi-criteria optimization problem is often an **NP**-hard problem, [EHR 00].

Approximating it with a performance guarantee, *i.e.* designing polynomial-time algorithms which return *ε-approximate Pareto curves*, is a motivating challenge. Inapproximability results are also needed. However, inapproximability results are not numerous in other works on multi-criteria optimization.

In this chapter, we provide approximation and inapproximation results for several versions of the multi-criteria traveling salesman with distances one and two. In particular, we propose a way to get some negative results which works for several multi-criteria problems and we put it into practice on the multi-criteria $TSP(1,2)$. To our knowledge, existing multi-criteria inapproximation results have been investigated only from the point of view of **NP**-hardness [DEN 02, PAP 00, VAS 05]. Our method is based on the following observation: in multi-criteria optimization, we try

to approximate a set of solutions (the Pareto curve) with another set of solutions (the ε-approximate Pareto curve) and the more the ε-approximate Pareto curve contains solutions, the more accurate the approximation can be. Then, the best approximation ratio that could be achieved can be related to the size of the approximate Pareto curve.

2.1.3. *Organization of the chapter*

The chapter is organized as follows: in section 2.2, we give some definitions concerning exact and approximate Pareto curves. Section 2.3 is devoted to a method used to derive some negative results (section 2.3.1) for the bicriteria $TSP(1,2)$ and two polynomial time algorithms providing constant approximation of the Pareto curve. More precisely, in section 2.3.2, we propose a *local search* algorithm called BLS which, with only two solutions generated in $\mathcal{O}(n^3)$, returns a $1/2$-approximate Pareto curve and in section 2.3.3, we propose a *greedy* algorithm inspired by the traditional *nearest neighbor* heuristic. This algorithm, called 2NN, returns two solutions generated in $\mathcal{O}(n^2)$ which constitute a $1/2$-approximation of the Pareto curve. We also give in section 2.3.4, some approximation results for a restriction of the bicriteria Max $TSP(1,2)$ problem. In section 2.4, we study the k-criteria $TSP(1,2)$ problem. More precisely, in section 2.4.1, we propose several negative results for k-criteria $TSP(1,2)$. Our method is applied in the case of k-criteria $TSP(1,2)$ but it works for several other problems. In section 2.4.2, we study the approximability of the k-criteria $TSP(1,2)$ by giving a generalized version of 2NN. This multi-criteria nearest neighbor heuristic, called KNN, works for any $k \geqslant 3$ and produces a $(k-1)/(k+1)$-approximate Pareto curve. Finally, some concluding remarks are given in section 2.5.

2.2. Overview

The traveling salesman problem consists of finding in a weighted complete graph $G = (V, E)$ on n vertices, a Hamiltonian cycle whose total distance is minimal. For the k-criteria TSP , each edge e has a *distance* $\vec{d}(e) = (\vec{d}_1(e), \ldots, \vec{d}_k(e))$ which is a vector of length k (instead of a scalar). The *total distance* of a tour T is also a vector $\vec{D}(T)$ where $\vec{D}_j(T) = \sum_{e \in T} \vec{d}_j(e)$ and $j = 1, \ldots, k$. In fact, a tour is evaluated with k objective functions. Given this, the goal of the optimization problem could be the following: generating a feasible solution which simultaneously minimizes (or maximizes) each co-ordinate. Unfortunately, such an ideal solution rarely exists since objective functions are, in general, in conflict. However, a set of solutions representing all best possible trade-offs always exists: the so-called *Pareto curve*. Formally, a Pareto curve is a set of feasible solutions, each of them optimal in the sense of Pareto, which *dominates* all the other solutions outside this set. If the problem is a minimization problem, a tour T dominates another tour T' iff $\vec{D}_j(T) \leqslant \vec{D}_j(T')$ for all $j = 1, \ldots, k$

and, for at least one co-ordinate j', $\vec{D}_{j'}(T) < \vec{D}_{j'}(T')$. When the problem is a maximization problem, we just reverse the previous inequalities. A solution is optimal in the sense of Pareto if no solution dominates it.

Since calculating Pareto curves is problematic, getting an approximation of it is more reasonable and often sufficient. For a minimization (resp. maximization) problem, an ε-approximate Pareto curve P_ε is a set of solutions such that, for every solution s of the instance, there is an s' in P_ε which satisfies $\vec{D}_j(s') \leqslant (1+\varepsilon)\vec{D}_j(s)$ (resp. $\vec{D}_j(s') \geqslant (1-\varepsilon)\vec{D}_j(s)$) for all $j = 1, \ldots, k$[1].

Interestingly, Papadimitriou and Yannakakis [PAP 00] prove that every multicriteria problem has an ε-approximate Pareto curve that is polynomial in the size of the input and $1/\varepsilon$, but exponential in the number k of criteria. However, calculating such an approximate Pareto curve cannot always be done within polynomial-time. The design of polynomial time algorithms which generate approximate Pareto curves with performance guarantee motivated a lot of recent papers.

2.3. The bicriteria $TSP(1,2)$

We consider the bicriteria $TSP(1,2)$ with n cities. In this case, we recall that for an edge e, $\vec{d}(e) = (\vec{d}_1(e), \vec{d}_2(e)) \in \{(1,1),(1,2),(2,1),(2,2)\}$ and the objective for the minimization version, called bicriteria Min $TSP(1,2)$ (resp., for the maximization version, called bicriteria Max $TSP(1,2)$) is to find a tour T minimizing (resp., maximizing) $\vec{D}_1(T) = \sum_{e\in T} \vec{d}_1(e)$ and $\vec{D}_2(T) = \sum_{e\in T} \vec{d}_2(e)$. We mainly propose two different algorithms leading to the same approximation of the Pareto curve for the bicriteria $Min\, TSP(1,2)$: a *local search* procedure using the *2-opt neighborhood* and a *nearest neighbor* heuristic which calculates a $1/2$-approximate Pareto curve. Although these two algorithms provide the same performance guarantee, it is interesting to present them since they are adaptations of well known heuristics previously used for the mono-criterion TSP [JOH 85, ROS 77].

For the bicriteria $TSP(1,2)$ (minimization and maximization versions), it is easy to observe that the Pareto curve is composed of at most $n+1$ solutions where we recall that n is the number of vertices; the worst case appears when the $n+1$ tours T_i with vector distance $\vec{D}(T_i) = (\vec{D}_1(T_i), \vec{D}_2(T_i)) = (n+i, 2n-i)$ for $i = 0, \cdots, n$, belong to the instance. Moreover, notice that a 1-approximate Pareto curve can be trivially constructed, just pick any tour.

1. Here, ε is the error while $1+\varepsilon$ (resp. $1-\varepsilon$) is the approximation ratio for a minimization (resp., maximization) problem. In the chapter, we equivalently use the error and its associated approximation ratio.

We now prove that the approximability of the Pareto curve for bicriteria Min $TSP(1,2)$ and the approximability of the Pareto curve for bicriteria Max $TSP(1,2)$ are linked by an approximation preserving reduction. The idea is to modify the instance by replacing each edge (2,2) by an edge (1,1), each edge (1,1) by an edge (2,2), and each edge (1,2) by an edge (2,1) and vice versa. It can be shown that obtaining a ε-approximate Pareto curve for the bicriteria Min $TSP(1,2)$ on this modified instance yields a $2\varepsilon/(3\varepsilon+1)$-approximate Pareto curve for the bicriteria Max $TSP(1,2)$ problem on the original instance.

THEOREM 2.1.– *Any ε-approximate Pareto curve for the bicriteria Min $TSP(1,2)$ problem can be polynomially transformed into a $\frac{2\varepsilon}{3\varepsilon+1}$-approximate Pareto curve for the bicriteria Max $TSP(1,2)$ problem.*

Proof. We only prove the result for the mono-criterion problems, i.e., any ε-approximation for Min $TSP(1,2)$ can be polynomially transformed into a $\frac{2\varepsilon}{3\varepsilon+1}$-approximation for Max $TSP(1,2)$. Let $I=(G,d)$ be an instance of Max $TSP(1,2)$ where $G=(V,E)$ is a complete graph on n vertices and consider the instance $I'=(G,d')$ of Min $TSP(1,2)$ with $d'(e)=3-d(e)$ for all $e\in E$. Finally, let T^* be an optimal solution of I for Max $TSP(1,2)$ and assume that T is an ε-approximation for Min $TSP(1,2)$ on I'. Obviously, T is also a solution on I and consider the two following cases:

- if $d(T^*)\leqslant\frac{3\varepsilon+1}{\varepsilon+1}n$, then since $d(T)\geqslant n$, we have:

$$\frac{d(T)}{d(T^*)}\geqslant\frac{\varepsilon+1}{3\varepsilon+1}=1-\frac{2\varepsilon}{3\varepsilon+1}$$

thus, T is a $\frac{2\varepsilon}{3\varepsilon+1}$-approximation for Max $TSP(1,2)$.

- Otherwise, $d(T^*)\geqslant\frac{3\varepsilon+1}{\varepsilon+1}n$. By construction, any tour T' verifies $d'(T')=3n-d(T')$. In particular, this is true for T^* and T. Hence, we deduce that T^* is also an optimal solution of I' for Min $TSP(1,2)$ and we have:

$$d'(T^*)\leqslant\frac{2}{\varepsilon+1}n \qquad [2.1]$$

Since by hypothesis $d'(T)\leqslant(1+\varepsilon)d'(T^*)$, we obtain:

$$\frac{d(T)}{d(T^*)}=\frac{3n-d'(T)}{3n-d'(T^*)}\geqslant 1-\varepsilon\frac{d'(T^*)}{3n-d'(T^*)} \qquad [2.2]$$

Using inequality [2.1] and since the right side of inequality [2.2] is increasing with $d'(T^*)$, we deduce $d(T)\geqslant(1-\frac{2\varepsilon}{3\varepsilon+1})d(T^*)$.

This completes the proof. ∎

2.3.1. *Simple examples of the non-approximability*

Usually, non-approximability results for mono-criteria problems bring thresholds of performance guarantee under which no polynomial time algorithm is likely to exist. Given a result of that kind for a mono-criterion problem Π, we directly get a negative result for a multi-criteria version of Π. Indeed, the multi-criteria version of Π generalizes Π. The non-approximability of the mono-criterion $TSP(1,2)$ has been studied in [ENG 01, PAP 93] and the best known lower bound is $1 + 1/740 - \delta$ (for all $\delta > 0$). Consequently, for every $\delta > 0$, no polynomial time algorithm can generate a $(1/740 - \delta)$-approximate Pareto curve unless **P** = **NP**.

As indicated previously, in multi-criteria optimization, we try to approximate a set of solutions (the Pareto curve) with another set of solutions (the ε-approximate Pareto curve) and the more the ε-approximate Pareto curve contains solutions, the more accurate the approximation can be. As a consequence, the best approximation ratio that could be achieved can be related to the size of the approximate Pareto curve. Formally, ε is a function of $|P_\varepsilon|$. If we consider instances for which the whole (or a large part of the) Pareto curve P is known and if we suppose that we approximate it with a set $P' \subset P$ such that $|P'| = x$, then the best approximation ratio ε such that P' is an ε-approximate Pareto curve is related to x. Indeed, there must be a solution in P' which approximates at least two (or more) solutions in P.

The question asked here is: *"what is the best approximation ratio an algorithm $\mathcal{A}$ can achieve if it outputs r solutions?"* (where r is supposed to be between 1 and $n + 1$).

In order to obtain an upper bound of this approximation ratio, we consider a particular class of instances for which the distances on the edges are in $\{(1,2),(2,1)\}$. This subclass is very interesting since each tour is Pareto optimal. Actually, let G be a n nodes graph of this class; each feasible tour has a total distance which is $(n+q, 2n-q)$ where q is the number of $(2,1)$ edges contained in the tour. Thus, no feasible tour can be dominated by another one.

THEOREM 2.2.– *Any ε-approximate Pareto curve P_ε for the bicriteria Min $TSP(1,2)$ problem composed of at most r tours is such that $\varepsilon \geqslant \frac{1}{3r-1}$.*

Proof. In what follows, we consider an instance of the subclass described above and assume that n is a multiple of r and all solutions T_i^* with distance $\vec{D}(T_i^*) = \left(n + \frac{i}{r}n, 2n - \frac{i}{r}n\right)$ are feasible for $i = 0, \cdots, r$. A formal proof of this claim will be given in section 2.4.1 when we generalize this result to any number of criteria k; actually, the goal of this proof is just to give an intuition of the method used. Now, assume that $\mathcal{A}$ outputs r tours T_i with $i = 1, \cdots, r$, which approximates the Pareto curve within a ratio $(1 + \alpha)$ on both criteria. Since there are $r + 1$ Pareto optimal tours T_i^*, at least

one tour $T_i \in P_\varepsilon$ must approximate two tours T_i^* for $i = x$ and $i = y$ with $x < y$. In particular, we have $y \geqslant x + 1$, and thus, if $\vec{D}(T_i) = \big(n + q, 2n - q\big)$, we deduce:

$$\begin{aligned} n + q &\leqslant (1+\alpha)\frac{r+x}{r}n \\ 2n - q &\leqslant (1+\alpha)\frac{2r-x-1}{r}n \end{aligned}$$

From the first inequality we derive that:

$$q \leqslant \frac{x}{r}n + \frac{r+x}{r}\alpha n$$

and from the second we get that:

$$q \geqslant \frac{x+1}{r}n - \frac{2r-x-1}{r}\alpha n$$

In fact

$$\frac{x+1}{r}n - \frac{2r-x-1}{r}\alpha n \leqslant q \leqslant \frac{x}{r}n + \frac{r+x}{r}\alpha n$$

can only be possible if $\alpha \geqslant \frac{1}{3r-1}$. ∎

When the goal is the maximization of both criteria (i.e., the bicriteria Max $TSP(1,2)$ problem), we obtain a threshold of inapproximation of $2/(3r-2)$ using the approximation curve preserving the reduction given in Theorem 2.1. However, applying the same proof of Theorem 2.2 in the context of bicriteria Max $TSP(1,2)$, we can improve this ratio to $\frac{1}{3r+1}$.

COROLLARY 2.1.– Any ε-approximate Pareto curve P_ε of the bicriteria Max $TSP(1,2)$ composed of at most r tours is such that $\varepsilon \geqslant \frac{1}{3r+1}$.

2.3.2. *A local search heuristic for the bicriteria* $TSP(1,2)$

In [KHA 98] and [MON 02b], the authors have shown separately that a simple local search algorithm using the well known 2-opt neighborhood [CRO 58] returns a $3/2$-approximate tour for the mono-criterion Min $TSP(1,2)$. To the best of our knowledge, no result was known about the ability of local search algorithms to provide solutions with performance guarantee in multi-criteria optimization. Here, we present an algorithm of this type which returns a $1/2$-approximate Pareto curve for the bicriteria $TSP(1,2)$ problem using the same neighborhood.

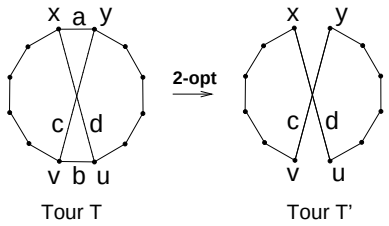


Figure 2.1. *The 2-opt move depicted here consists of replacing edges a and b by c and d. We have $T' \in \mathcal{N}(T)$*

Given a tour T, its 2-opt neighborhood $\mathcal{N}(T)$ is the set of all Hamiltonian cycles which can be obtained by removing two non-adjacent edges and inserting two new edges (see Figure 2.1).

If only one criterion is considered, a simple local search algorithm consists of starting with a feasible solution and iteratively replacing it with a neighboring solution which is better with respect to the criterion. The algorithm stops when the current solution, a *local optimum*, has no better neighbor.

If we say that a tour T is a local optimum tour with respect to the 2-opt neighborhood when no tour $T' \in \mathcal{N}(T)$ dominates T, then there exist instances for which a locally optimal tour gives a poor performance guarantee for at least one criterion. Then, we introduce two symmetric preference relations defining potentially two different local optima. These preference relations, denoted by $\prec_1$ and $\prec_2$, are depicted in Figures 2.2 and 2.3. Note that we define them considering the fact that we deal with 2-opt moves, which are exchanges of couples of edges. The set of the 10 possible couples of distance-vectors of the edges has been partitioned into three sets S_1, S_2 and S_3, and for any $c_1 \in S_1$, $c_2 \in S_2$, $c_3 \in S_3$, we have $c_1 \prec_1 c_2 \prec_1 c_3$ (resp. $c_1 \prec_2 c_2 \prec_2 c_3$). Intuitively, the preference relation $\prec_1$ (resp. $\prec_2$) leads to solutions which are good for the first (resp. second) criterion.

DEFINITION 2.1.– *We say that the tour T is a local optimum tour with respect to the 2-opt neighborhood and the preference relation $\prec_1$ (resp. $\prec_2$) if there does not exist a tour $T' \in \mathcal{N}(T)$, obtained from T by removing edges a, b and inserting edges c, d, such that $(\vec{d}(c), \vec{d}(d)) \prec_1 (\vec{d}(a), \vec{d}(b))$ (resp. $(\vec{d}(c), \vec{d}(d)) \prec_2 (\vec{d}(a), \vec{d}(b))$).*

We consider the following algorithm:

BICRITERIA LOCAL SEARCH (BLS):

1) Let T_1 be a 2-opt local optimum tour with the preference relation $\prec_1$.

2) Let T_2 be a 2-opt local optimum tour with the preference relation $\prec_2$.

3) Return $\{T_1, T_2\}$.

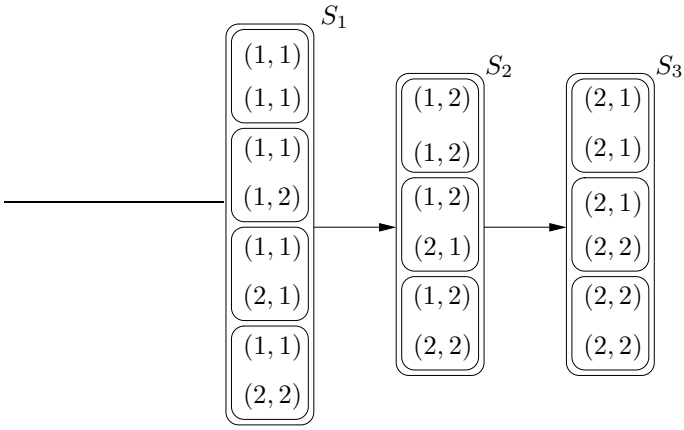


Figure 2.2. *The preference relation* $\prec_1$

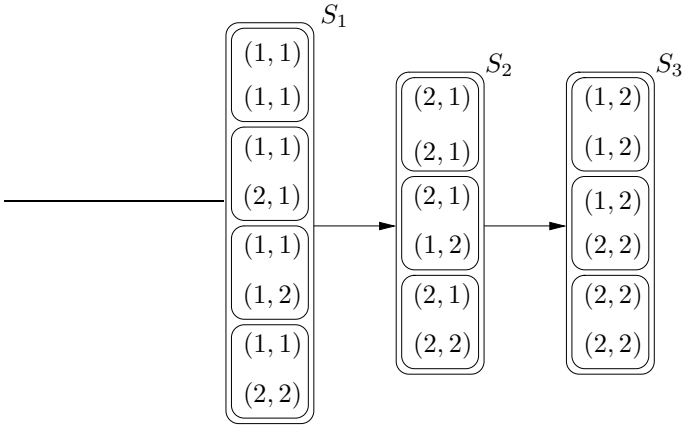


Figure 2.3. *The preference relation* $\prec_2$

Analysis of BLS. In what follows, we assume that T is any 2-opt local optimal tour with respect to the preference relation $\prec_1$. The tour O is any fixed tour. We denote by x (resp. y,z and t) the number of edges with distance vector (1,1) (resp. (1,2), (2,1) and (2,2)) in T. We denote by x' (resp. y',z' and t') the number of edges with distance vector (1,1) (resp. (1,2), (2,1) and (2,2)) in O.

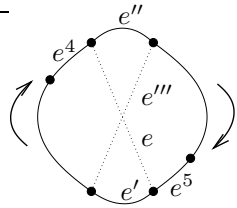


Figure 2.4. *The local optimal tour T (arbitrarily oriented)*

LEMMA 2.1.– *With the preference relation $\prec_1$ we have $x \geqslant x'/2$.*

Proof. Let U_O (resp. U_T) be the set of $(1,1)$ edges in O (resp. in T). We define a function $f : U_O \to U_T$ as follows: for $e \in U_O$, $f(e) = e$ if $e \in U_T$. Otherwise, let e' and e'' be the two edges adjacent to e in T as depicted in Figure 2.4 (we assume an arbitrary orientation of T and consider that the edges adjacent to e are e' and e'' but not e^4 and e^5). Let e''' be the edge forming a cycle of length 4 with e, e' and e'' (see Figure 2.4). We claim that there is at least one edge among e' and e'' with a distance $(1,1)$ and define $f(e)$ to be one of those edges (possibly chosen arbitrarily). Otherwise, we have $\{e, e'''\} \in S_1$ and $\{e', e''\} \in S_2 \cup S_3$ (see Figures 2.2 and 2.3), contradicting the fact that T is a local optimum with respect to the preference relation $\prec_1$. Now observe that for a given edge $e'' \in U_T$, there can be at most two edges e and e' in U_O such that $f(e) = f(e') = e''$ since O is a tour. Therefore, we have $|U_T| \geqslant |U_O|/2$. ∎

LEMMA 2.2.– *With the preference relation $\prec_1$ we have $x + y \geqslant (x' + y')/2$.*

Proof. The proof is similar to that of Lemma 2.1. Here, U_O (resp. U_T) is the set of $(1,1)$ and $(1,2)$ edges of O (resp. T). ∎

LEMMA 2.3.– *If $\vec{D}_1(O) \leqslant \vec{D}_2(O)$ then $\vec{D}_1(T) \leqslant \frac{3}{2}\vec{D}_1(O)$ and $\vec{D}_2(T) \leqslant \frac{3}{2}\vec{D}_2(O)$.*

Proof. We have $\vec{D}_1(T) = 2n - x - y$, $\vec{D}_1(O) = 2n - x' - y'$ and $\vec{D}_2(T) = 2n - x - z$, $\vec{D}_2(O) = 2n - x' - z'$. Let us consider the first co-ordinate. We want to show that $\frac{\vec{D}_1(T)}{\vec{D}_1(O)} = \frac{2n-x-y}{2n-x'-y'} \leqslant \frac{3}{2}$. Using Lemma 2.2 we get $\frac{2n-x-y}{2n-x'-y'} \leqslant \frac{2n-\frac{x'}{2}-\frac{y'}{2}}{2n-x'-y'}$. Now we have

$$\frac{2n - \frac{x'}{2} - \frac{y'}{2}}{2n - x' - y'} \leqslant \frac{3}{2} \iff 4n - x' - y' \leqslant 6n - 3x' - 3y'$$
$$\iff x' + y' \leqslant n$$

which is true since $x' + y' + z' + t' = n$ and $z', t' \geqslant 0$. Now, we consider the second co-ordinate and use the fact that $\vec{D}_1(O) \leqslant \vec{D}_2(O) \iff z' \leqslant y'$.

$$\begin{aligned}
\frac{2n - x - z}{2n - x' - z'} \leqslant \frac{3}{2} \quad &\iff \quad 4n - 2x - 2z \leqslant 6n - 3x' - 3z' \\
&\iff \quad 3x' - 2x + 3z' - 2z \leqslant 2n \\
&\iff \quad 3x' - 2x + 3z' - 2z \leqslant 2(x' + y' + z' + t') \\
&\iff \quad x' - 2x + z' - 2z \leqslant 2y' + 2t'
\end{aligned}$$

which is true since $x' - 2x \leqslant 0$ according to Lemma 2.1, $z' \leqslant y'$ and $-z \leqslant t'$. ∎

Now, we suppose that T is a 2-opt local optimal tour with respect to the preference relation $\prec_2$. The tour O is any fixed tour. In a similar way to the case of $\prec_1$, we can prove:

LEMMA 2.4.– *With the preference relation $\prec_2$ we have $x \geqslant \frac{x'}{2}$ and $x + z \geqslant (x' + z')/2$.*

LEMMA 2.5.– *If $\vec{D}_1(O) \geqslant \vec{D}_2(O)$, then $\vec{D}_1(T) \leqslant \frac{3}{2}\vec{D}_1(O)$ and $\vec{D}_2(T) \leqslant \frac{3}{2}\vec{D}_2(O)$.*

Now, we are ready to prove the main result.

THEOREM 2.3.– *The set of tours returned by BLS is a 1/2-approximate Pareto curve for the bicriteria Min $TSP(1,2)$ problem. Moreover, this bound is asymptotically sharp.*

Proof. Using Lemma 2.3 and Lemma 2.5, we know that given a Pareto optimal tour O, if $\vec{D}_1(O) \leqslant \vec{D}_2(O)$, then $\vec{D}(T_1) \leqslant \frac{3}{2}\vec{D}(O)$, otherwise $\vec{D}(T_2) \leqslant \frac{3}{2}\vec{D}(O)$.

In order to see that this bound is asymptotically sharp, consider the instance depicted in Figure 2.5. The tour $s_1 s_2 \ldots s_{2n} s_1$ is a local optimum with respect to $\prec_1$ and $\prec_2$, and it has a total distance vector $(3n, 3n)$, whereas the optimal tour:

$$s_1 s_3 s_{2n} s_4 s_{2n-1} \ldots s_{n-1} s_{n+4} s_n s_{n+3} s_{n+1} s_{n+2} s_2 s_1$$

has a total distance vector $(2n + 1, 2n + 1)$. ∎

Concerning the time complexity, we can show that BLS runs in time $\mathcal{O}(n^3)$ since searching the 2-opt neighborhood of a tour is done in $\mathcal{O}(n^2)$ and at most $\mathcal{O}(n)$ 2-opt moves are done to reach a local optimum.

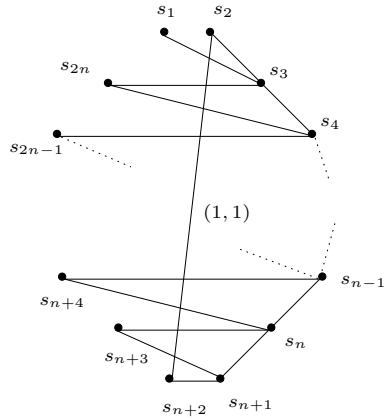


Figure 2.5. *The edges represented have a distance vector* $(1,1)$*, whereas non-represented edges have a distance vector* $(2,2)$

The result given in Theorem 2.3 can also be applied to the bicriteria version of the Max $TSP(1,2)$ problem. We recall that in this problem, the objective is the maximization of the length of the tour. For the mono-criterion case, the best approximation algorithm known has a performance ratio of 7/8 [MON 02, MON 03]. Using Theorems 2.1 and 2.3, we obtain a 2/5-approximate Pareto curve for the bicriteria Max $TSP(1,2)$ problem. However, if we call BMAXLS the algorithm which consists of using modified preference relations $\prec_1'$ and $\prec_2'$ obtained from $\prec_1$ and $\prec_2$ by replacing each edge (a,b) by an edge $(3-a,3-b)$, we can show that the inequalities obtained in Theorem 2.3 allow us to obtain a 1/3-approximate Pareto curve.

COROLLARY 2.2.– The set of solutions returned by BMAXLS is a 1/3-approximate Pareto curve for the bicriteria Max $TSP(1,2)$ problem. Moreover, this bound is asymptotically sharp.

2.3.3. *A nearest neighbor heuristic for the bicriteria* $TSP(1,2)$

We now propose a *nearest neighbor* heuristic which calculates in $\mathcal{O}(n^2)$ a $1/2$-approximate Pareto curve for the bicriteria $Min\ TSP(1,2)$. The idea of this traditional heuristic, applied to a mono-criterion TSP instance, consists of starting from a randomly chosen node and greedily inserting non-visited vertices, which are chosen as those closest to the last inserted vertex [ROS 77]. We adapt this algorithm to the bicriteria Min $TSP(1,2)$. As done before with BLS, we build two solutions using two symmetric preference relations denoted by $\prec_1$ and $\prec_2$.

2NN: Bicriteria Nearest Neighbor
Input: $I = (G, \vec{d})$ instance of the bicriteria $TSP(1,2)$;
Output: Two tours p_1 and p_2 of I;

Take arbitrarily $v^\star \in V$;
Set $v_s = v^\star$
Set $S = \{v_s\}$ and $u = v_s$;
Until $S \neq V$ Do
  Take $r \in V - S$ such that $\nexists t \in V - S$ s.t. $\vec{d}([u,t]) \prec_1 \vec{d}([u,r])$;
  Set $p_1(u) = r$ and $u = r$;
End Until;
Set $p_1(r) = v_s$;

Take arbitrarily $v^{\star\star} \in V$;
Set $v_s = v^{\star\star}$
Set $S = \{v_s\}$ and $u = v_s$;
Until $S \neq V$ Do
  Take $r \in V - S$ such that $\nexists t \in V - S$ s.t. $\vec{d}([u,t]) \prec_2 \vec{d}([u,r])$;
  Set $p_2(u) = r$ and $u = r$;
End Until;
Set $p_2(r) = v_s$;

Return $\{p_1, p_2\}$;

Table 2.1. 2NN *algorithm*

DEFINITION 2.2.– *For two edges e and e', $\vec{d}(e) \prec_1 \vec{d}(e')$ if $\vec{d}_1(e) < \vec{d}_1(e')$ or, $\vec{d}_1(e) = \vec{d}_1(e')$ and $\vec{d}_2(e) < \vec{d}_2(e')$. Symmetrically, $\vec{d}(e) \prec_2 \vec{d}(e')$ if $\vec{d}_2(e) < \vec{d}_2(e')$ or, $\vec{d}_2(e) = \vec{d}_2(e')$ and $\vec{d}_1(e) < \vec{d}_1(e')$. When $\vec{d}(e) \not\prec_1 \vec{d}(e')$ and $\vec{d}(e) \not\prec_2 \vec{d}(e')$, we say that $\vec{d}(e)$ are $\vec{d}(e')$ incomparable.*

The algorithm proposed is called 2NN (*Bicriteria Nearest Neighbor*) and is given in Table 2.1. This algorithm returns two tours p_1 and p_2. We assume that for each node $v \in V$, $p_1(v)$ (resp. $p_2(v)$) represents the node which immediately follows v in p_1 (resp. p_2). Here, p^* denotes a Pareto optimal tour. Before proving that the two tours returned by 2NN constitute a $1/2$-approximate Pareto curve for the bicriteria $TSP(1,2)$, we introduce some notations. Let x_1 (resp. y_1, z_1, t_1) be the number of edges with distance $(1,1)$ (resp. $(1,2)$, $(2,1)$, $(2,2)$) of p_1. Let x_2 (resp. y_2, z_2, t_2) be the number of edges with distance $(1,1)$ (resp. $(1,2)$, $(2,1)$, $(2,2)$) of p_2. Let x^* (resp. y^*, z^*, t^*) be the number of edges with distance $(1,1)$ (resp. $(1,2)$, $(2,1)$, $(2,2)$) of p^*.

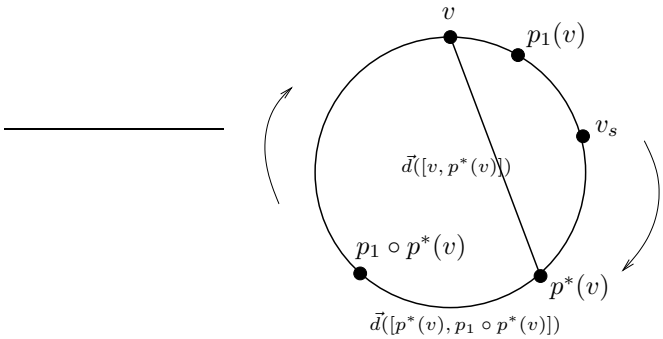


Figure 2.6. *Case 1*

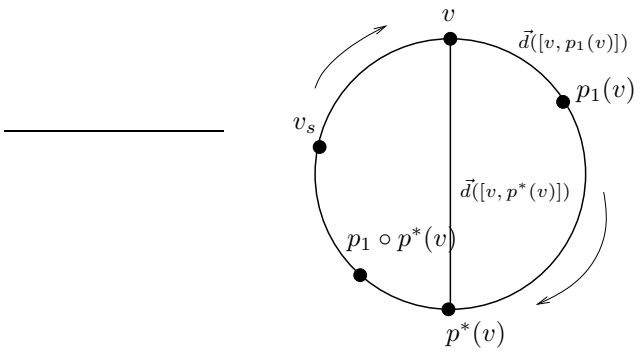


Figure 2.7. *Case 2*

Since a tour has exactly n edges, we know that:

$$n = x_1 + y_1 + z_1 + t_1 = x_2 + y_2 + z_2 + t_2 = x^* + y^* + z^* + t^*$$

LEMMA 2.6.– *For p_1 and p^*, we have $x_1 \geqslant x^*/2$.*

Proof. Let U_{p_1} (resp. U_{p^*}) be the set of $(1,1)$ edges of p_1 (resp. p^*). We define a function $f : U_{p^*} \to U_{p_1}$ in the following way. Let $[v, v']$ be an edge in U_{p^*}. If $[v, v'] \in U_{p_1}$ then $f([v, v']) = [v, v']$. Otherwise, we claim that among $[v, p_1(v)]$ and $[v', p_1(v')]$, there is at least one edge with distance $(1,1)$. Thus, $f([v, v']) = [v, p_1(v)]$ if the distance vector of $[v, p_1(v)]$ is $(1,1)$, otherwise $f([v, v']) = [v', p_1(v')]$. To see that, we consider two possibilities:

– $p_1(v) = p^*(v)$: if $\vec{d}([v,p^*(v)]) = (1,1)$, then $\vec{d}([v,p_1(v)]) = (1,1)$ and $f([v,p^*(v)]) = [v,p_1(v)]$.

– $p_1(v) \neq p^*(v)$: in case 1 (see Figure 2.6), if $\vec{d}([v,p^*(v)]) = (1,1)$, then $\vec{d}([p^*(v),p_1(p^*(v))]) = (1,1)$ and $f([v,p^*(v)]) = [p^*(v),p_1(p^*(v))]$. In case 2 (see Figure 2.7), if $\vec{d}([v,p^*(v)]) = (1,1)$, then $\vec{d}([v,p_1(v)]) = (1,1)$ and $f([v,p^*(v)]) = [v,p_1(v)]$.

Thus, each $(1,1)$ edge of p_1 has at most two antecedents of distance $(1,1)$. ∎

LEMMA 2.7.– *For p_2 and p^*, we have $x_2 \geqslant x^*/2$.*

Proof. See the proof of Lemma 2.6 and replace p_1 by p_2. ∎

LEMMA 2.8.– *For p_1 and p^*, we have $x_1 + y_1 \geqslant (x^* + y^*)/2$.*

Proof. Let U_{p_1} (resp. U_{p^*}) be the set of $(1,1)$ and $(1,2)$ edges of p_1 (resp. p^*). We define a function $f : U_{p^*} \to U_{p_1}$ in the following way. Let $[v,v']$ be an edge in U_{p^*}. If $[v,v'] \in U_{p_1}$ then $f([v,v']) = [v,v']$. Otherwise, we claim that among $[v,p_1(v)]$ and $[v',p_1(v')]$, there is at least one edge with distance $(1,1)$ or $(1,2)$. Thus, $f([v,v']) = [v,p_1(v)]$ if the distance of $[v,p_1(v)]$ is $(1,1)$, otherwise $f([v,v']) = [v',p_1(v')]$. To see that, we consider two possibilities:

– $p_1(v) = p^*(v)$: if $\vec{d}([v,p^*(v)]) = (1,1)$, then $\vec{d}([v,p_1(v)]) = (1,1)$ or $\vec{d}([v,p_1(v)]) = (1,2)$ and $f([v,p^*(v)]) = [v,p_1(v)]$.

– $p_1(v) \neq p^*(v)$: in case 1 (see Figure 2.6), if $\vec{d}([v,p^*(v)]) = (1,1)$ or $\vec{d}([v,p^*(v)]) = (1,2)$, then $\vec{d}([p^*(v),p_1(p^*(v))]) = (1,1)$ or $\vec{d}([p^*(v),p_1(p^*(v))]) = (1,2)$ and $f([v,p^*(v)]) = [p^*(v),p_1(p^*(v))]$. In case 2 (see Figure 2.7), if $\vec{d}([v,p^*(v)]) = (1,1)$ or $\vec{d}([v,p^*(v)]) = (1,2)$, then $\vec{d}([v,p_1(v)]) = (1,1)$ or $\vec{d}([v,p_1(v)]) = (1,2)$ and $f([v,p^*(v)]) = [v,p_1(v)]$.

Thus, each edge of distance in $\{(1,1),(1,2)\}$ of p_1 has at most two antecedents of distance $(1,1)$ or $(1,2)$. ∎

LEMMA 2.9.– *For p_2 and p^*, we have $x_2 + z_2 \geqslant (x^* + z^*)/2$.*

Proof. See the proof of Lemma 2.8. ∎

THEOREM 2.4.– *The two tours returned by 2NN constitute a $1/2$-approximate Pareto curve for the bicriteria $TSP(1,2)$. Moreover, this bound is asymptotically sharp.*

Proof. The proof is completely similar to that given in Theorem 2.3. Moreover, this bound is asymptotically attained for the particular instance given in Figure 2.5. Indeed, 2NN can produce two identical tours $s_1\, s_2\, \ldots\, s_{2n-1}\, s_{2n}$ of total distance $(3n,3n)$

while tour $s_1\, s_3\, s_{2n}\, s_4 \ldots s_{n+3}\, s_{n+1}\, s_{n+2}\, s_2$ has a total distance of $(2n+1, 2n+1)$. ∎

In the next section, we deal with a subcase of the bicriteria Max $TSP(1,2)$ problem.

2.3.4. *On the bicriteria Max* $TSP(1,2)$

In Corollary 2.2, we saw that BMAXLS gives a 1/3-approximate Pareto curve for the bicriteria Max $TSP(1,2)$ problem. In this section we improve this performance ratio, but only when we restrict the distances vector to edges with distance vectors $(1,2)$ and $(2,1)$. Actually, using Corollary 2.1, we observe that this restriction is already hard to approximate with $\varepsilon < \frac{1}{3r+1}$ if we use at most r tours. In particular, a single tour never provides a $(1/4+\varepsilon)$-approximation of this restriction, for any $\varepsilon > 0$. We now prove how we can reach this performance ratio using only 3 tours. The algorithm presented below, starts with the two solutions produced by a maximization version of 2NN together with a third solution based on matchings.

AL1:
Input: $I = (G, d)$ instance of the bicriteria Max $TSP(1,2)$ where $d(e) \in \{(1,2),(2,1)\}$;
Output: Three tours T_1, T_2 and T_3 of I ;

1. Produce a tour T_1 using the nearest neighbor algorithm where edges $(1,2)$ are preferred to edges $(2,1)$;
2. Produce a tour T_2 using the nearest neighbor algorithm where edges $(2,1)$ are preferred to edges $(1,2)$;
3. Find a third tour T_3 in the following way
   - 3.1 Find a maximum matching M_1 in the subgraph induced by the $(1,2)$ edges;
   - 3.2 Find a maximum matching M_2 in the subgraph induced by the $(2,1)$ edges;
   - 3.3 $C_1, \ldots, C_q$ and $P_1, \ldots, P_r$, respectively, denote the cycles and the paths in the graph induced by $M_1 \cup M_2$;
   - 3.4 For each C_{2i} with $1 \leqslant 2i \leqslant q$, delete one edge with distance vector $(1,2)$ and let P'_{2i} denote this path ;
   - 3.5 For each C_{2i+1} with $1 \leqslant 2i+1 \leqslant q$, delete one edge with distance vector $(2,1)$ and let P'_{2i+1} denote this path ;
   - 3.6 Build T_3 by adding arbitrarily chosen edges to $\left(\bigcup_{i=1}^{r} P_i\right) \cup \left(\bigcup_{i=1}^{q} P'_i\right)$;
4. Return T_1, T_2 and T_3;

Table 2.2. AL1 *algorithm*

THEOREM 2.5.– AL1 *algorithm (see Table 2.2) returns a* $\frac{1}{4}$*-approximate Pareto curve for the bicriteria Max* $TSP(1,2)$ *when we restrict the distance vectors to* $(1,2)$ *and* $(2,1)$.

Proof. We start with the notations introduced in section 2.3.3. The following equalities can be easily proved:

$$\begin{aligned} y_1 + z_1 &= y_2 + z_2 = y^* + z^* = n \\ \vec{D}(T_1) &= (n + z_1, n + y_1) \\ \vec{D}(T_2) &= (n + z_2, n + y_2) \\ \vec{D}(T^*) &= (n + z^*, n + y^*) \end{aligned} \quad [2.3]$$

According to Lemma 2.8, we have the following property for T_1:

$$2y_1 \geqslant y^* \quad [2.4]$$

Thus, using [2.3] and [2.4], we get:

$$\begin{aligned} \vec{D}_2(T_1) = n + y_1 &\geqslant n + y^*/2 \\ &= 3n/4 + 3y^*/4 + n/4 - y^*/4 \\ &= \frac{3}{4}(n + y^*) + z^*/4 \\ &\geqslant \frac{3}{4}\vec{D}_2(T^*) \end{aligned}$$

According to Lemma 2.9, we have the following property for T_2:

$$2z_2 \geqslant z^* \quad [2.5]$$

Thus, using [2.3] and [2.5], we get:

$$\begin{aligned} \vec{D}_1(T_2) = n + z_2 &\geqslant n + z^*/2 \\ &= 3n/4 + 3z^*/4 + n/4 - z^*/4 \\ &= \frac{3}{4}(n + z^*) + y^*/4 \\ &\geqslant \frac{3}{4}\vec{D}_1(T^*) \end{aligned}$$

(i) Assume that $z_1 \geqslant n/4$. If $2n \geqslant 3z^*$, then $\vec{D}_1(T_1) \geqslant \frac{3}{4}\vec{D}_1(T^*)$ and T_1 approximates the whole Pareto curve. Indeed, we have:

$$\begin{aligned} z_1 &\geqslant n/4 \\ n + z_1 &\geqslant 3n/4 + n/2 \\ n + z_1 &\geqslant 3n/4 + 3z^*/4 \\ \vec{D}_1(T_1) &\geqslant \frac{3}{4}\vec{D}_1(T^*) \end{aligned}$$

In the case where $2n \leqslant 3z^*$, we have $2y^* \leqslant z^*$ and $n \geqslant 3y^*$ since $n = y^* + z^*$. Moreover, $3n + 4y_2 \geqslant 3n$ is always true since $y_2 \geqslant 0$. If we add $n \geqslant 3y^*$ to $3n + 4y_2 \geqslant 3n$, we get:

$$\begin{aligned} 4n + 4y_2 &\geqslant 3n + 3y^* \\ \vec{D}_2(T_2) &\geqslant \frac{3}{4}\vec{D}_2(T^*) \end{aligned}$$

As a consequence, T_2 approximates the whole Pareto curve.

(ii) Assume $y_2 \geqslant n/4$. If $2n \geqslant 3y^*$, then $\vec{D}_2(T_2) \geqslant \frac{3}{4}\vec{D}_2(T^*)$ and T_2 approximates the whole Pareto curve. Indeed, we have:

$$\begin{aligned} y_2 &\geqslant n/4 \\ n + y_2 &\geqslant 3n/4 + n/2 \\ n + y_2 &\geqslant 3n/4 + 3y^*/4 \\ \vec{D}_2(T_2) &\geqslant \frac{3}{4}\vec{D}_2(T^*) \end{aligned}$$

In the case where $2n \leqslant 3y^*$, we have $2z^* \leqslant y^*$ and $n \geqslant 3z^*$ since $n = y^* + z^*$. Moreover, $3n+4z_1 \geqslant 3n$ is always true since $z_1 \geqslant 0$. If we add $n \geqslant 3z^*$ to $3n+4z_1 \geqslant 3n$, we get:

$$\begin{aligned} 4n + 4z_1 &\geqslant 3n + 3z^* \\ \vec{D}_1(T_1) &\geqslant \frac{3}{4}\vec{D}_1(T^*) \end{aligned}$$

As a consequence, T_1 approximates the whole Pareto curve.

(iii) Now assume that we simultaneously have $z_1 \leqslant n/4$ and $y_2 \leqslant n/4$. In this case, T_1 has at least $3n/4$ $(1,2)$ edges and T_2 has at least $3n/4$ $(2,1)$ edges. We deduce that M_1 (resp. M_2) has at least $3n/8$ edges of distance $(1,2)$ (resp. $(2,1)$). On the other hand, when we add M_1 to M_2, there is at most $n/4$ cycles and then, we delete at most $n/8$ edges of distance vector $(1,2)$ and $n/8$ edges of distance vector $(2,1)$. In conclusion, if z_3 (resp., y_3) denotes the set of $(2,1)$-edges (resp., $(1,2)$-edges) of T_3, we deduce that $z_3 \geqslant 3n/8 - n/8 = n/4$ and $y_3 \geqslant 3n/8 - n/8 = n/4$. Hence, we get:

$$\vec{D}(T_3) \geqslant (n + z_3, n + y_3) \geqslant (5n/4, 5n/4) \qquad [2.6]$$

We also assume that $z^* \leqslant 2y^*$ and $y^* \leqslant 2z^*$. Indeed, when $z^* \geqslant 2y^*$, we deduce that $y^* \leqslant n/3$ since $z^* + y^* = n$. Thus $\vec{D}_2(T^*) \leqslant 4n/3$. On the other hand, trivially $\vec{D}_2(T_2) \geqslant n$ and then $\vec{D}_2(T_2) \geqslant \frac{3}{4}\vec{D}_2(T^*)$. In conclusion, since previously we proved that $\vec{D}_1(T_2) \geqslant \frac{3}{4}\vec{D}_1(T^*)$ always holds, we get $\vec{D}(T_2) \geqslant \frac{3}{4}\vec{D}(T^*)$. When $y^* \geqslant 2z^*$, the same inequality holds for T_1. Hence, in conclusion we can assume that $z^* \leqslant 2y^*$ and $y^* \leqslant 2z^*$. Then, $2n \geqslant 3z^*$ and $2n \geqslant 3y^*$. We derive from these two inequalities that:

$$\begin{aligned} n/3 \leqslant \quad & y^* \quad \leqslant 2n/3 \\ n/3 \leqslant \quad & z^* \quad \leqslant 2n/3 \end{aligned}$$

So, we obtain:

$$\vec{D}(T^*) \leqslant (5n/3, 5n/3) \qquad [2.7]$$

Thus, in the worst case, T_3 approximates the whole Pareto curve within a ratio $3/4$. ∎

2.4. k-criteria $TSP(1,2)$

In this section, we present a generalization of the previous results when k, i.e., the number of criteria, is larger than 2. We first give some non-approximability results related to the number of generated solutions for the k-criteria $TSP(1,2)$. Afterwards, a generalization of 2NN, which has a better complexity than BLS, is proposed. This

generalization, called KNN, calculates in $\mathcal{O}(n^2k!)$ time a $\frac{k-1}{k+1}$-approximate Pareto curve for the k-criteria $TSP(1,2)$ when $k \geqslant 3$.

Let us observe here that the dependence of the time complexity on $k!$ is not surprising since the size of the approximate ε-Pareto curve is not necessarily polynomial on the number of the optimization criteria [PAP 00].

2.4.1. *Non-approximability related to the number of generated solutions*

We give in this section a non-trivial generalization of the results given in section 2.3.1. Thus, we propose a way to get some negative results which works for several multi-criteria problems and we put it into practice on the k-criteria $TSP(1,2)$.

In what follows, we explicitly give a family of instances (denoted by $I_{n,r}$) of the k-criteria $TSP(1,2)$ for which we know a lot of different Pareto optimal tours covering a large spectrum of the possible values.

We first consider an instance I_n with $n \geqslant 2k+1$ vertices where distances belong to $\{(1,2,\ldots,2),\ (2,1,2,\ldots,2),\ldots,(2,\ldots,2,1)\}$. We suppose that for any $i = 1,\ldots,k$, the subgraph of I_n induced by the edges whose distance is 1 only on co-ordinate i is Hamiltonian (T_i denotes this tour). Using an old result [LUC 92], we know that K_n is Hamiltonian cycles decomposable into k disjoint tours if $n \geqslant 2k+1$ and then, I_n exists.

We duplicate the instance I_n r times to get $I_{n,r}$. We denote by v_a^c the vertex v_a of the c-th copy of I_n. Between two copies with $1 \leqslant c_1 < c_2 \leqslant r$, we set $\vec{d}([v_a^{c_1}, v_b^{c_2}]) = \vec{d}([v_a, v_b])$ if $a \neq b$ and $\vec{d}([v_a^{c_1}, v_a^{c_2}]) = (1,2,\ldots,2)$.

LEMMA 2.10.– *There are* $\binom{r+k-1}{r}$ *Pareto optimal tours in* $I_{n,r}$ *(denoted by* $T_{c_1,\ldots,c_{k-1}}$ *where* c_i *for* $1 \leqslant i \leqslant k-1$ *are* $k-1$ *indices in* $\{0,\ldots,r\}$*) satisfying:*

(i) $\forall i = 1,\ldots,k-1$, $c_i \in \{0,\ldots,r\}$ *and* $\sum_{i=1}^{k-1} c_i \leqslant r$.

(ii) $\forall i = 1,\ldots,k-1$, $\vec{D}_i(T_{c_1,\ldots,c_{k-1}}) = 2rn - c_i n$ *and* $\vec{D}_k(T_{c_1,\ldots,c_{k-1}}) = rn + n(\sum_{i=1}^{k-1} c_i)$.

Proof. Let $c_1,\ldots,c_{k-1}$ be integers satisfying (i). We build the tour $T_{c_1,\ldots,c_{k-1}}$ by applying the following process: on the c_1 first copies, we take the tour T_1, on the c_2 second copies, we take the tour T_2 and so on. Finally, for the $r - \sum_{i=1}^{k-1} c_i$ last copies, we take T_k. For any $1 \leqslant l_1 < l_2 \leqslant r$, and any tours T and T', we patch T on copy l_1 with T' on copy l_2 by replacing the edges $[v_i^{l_1}, v_j^{l_1}] \in T$ and $[v_j^{l_2}, v_m^{l_2}] \in T'$ by the edges $[v_i^{l_1}, v_j^{l_2}]$ and $[v_m^{l_2}, v_j^{l_1}]$. Observe that the resulting tour has a total distance $\vec{D}(T') + \vec{D}(T)$. So, by applying r times this process, we can obtain a tour $T_{c_1,\ldots,c_{k-1}}$

satisfying (ii). Moreover, the number of tours is equal to the number of choices of $k-1$ elements among $r+(k-1)$. ∎

THEOREM 2.6.– *For any $k \geqslant 2$, any ε-approximate Pareto curve for the k-criteria $TSP(1,2)$ containing at most x solutions satisfies:*

$$\begin{aligned} \varepsilon &\geqslant max_{i=2,\ldots,k}\{\frac{1}{(2i-1)r(i,x)-1}\} \\ \textit{where } r(i,x) &= min\{r|\ x \leqslant \tbinom{r+i-1}{r} - 1\} \end{aligned}$$

Proof. Let $r(k,x) = r$ be the smallest integer such that $x \leqslant \binom{r+k-1}{r} - 1$ and consider the instance $I_{n,r}$. Since $x \leqslant \binom{r+k-1}{r} - 1$, there exists two distinct tours $T_{c_1,\ldots,c_{k-1}}$ and $T_{c'_1,\ldots,c'_{k-1}}$ and a tour T in the approximate Pareto curve such that:

$$\vec{D}(T) \leqslant (1+\varepsilon)\vec{D}(T_{c_1,\ldots,c_{k-1}}) \text{ and } \vec{D}(T) \leqslant (1+\varepsilon)\vec{D}(T_{c'_1,\ldots,c'_{k-1}}) \qquad [2.8]$$

Let $l_i = \max\{c_i, c'_i\}$ for $i = 1,\ldots,k-1$ and $l_k = \min\{\sum_{i=1}^{k-1} c_i, \sum_{i=1}^{k-1} c'_i\}$. By construction, we have $l_k \leqslant \sum_{i=1}^{k-1} l_i - 1$. Moreover, the total distance of T can be written $\vec{D}_i(T) = 2rn - q_i$ for $i = 1,\ldots,k-1$ and $\vec{D}_k(T) = rn + \sum_{i=1}^{k-1} q_i$ for any value of q_i (q_i is the number of edges of T where the distance has a 1 on co-ordinate i and 2 on the others). Thus, using inequalities [2.8], we deduce that for $i = 1,\ldots,k-1$, we have $2nr - q_i \leqslant (1+\varepsilon)(2rn - l_i n)$, which is equivalent to:

$$q_i \geqslant l_i n(1+\varepsilon) - 2rn\varepsilon \qquad [2.9]$$

Using inequalities [2.8], we also have $rn + \sum_{i=1}^{k-1} q_i \leqslant (1+\varepsilon)(rn + l_k n)$, which is equivalent to:

$$\sum_{i=1}^{k-1} q_i \leqslant \varepsilon rn + l_k n(1+\varepsilon) \qquad [2.10]$$

Adding inequalities [2.9] for $i = 1,\ldots,k-1$ and by using inequality [2.10] and $l_k \leqslant \sum_{i=1}^{k-1} l_i - 1$, we deduce:

$$\varepsilon \geqslant \frac{1}{(2k-1)r(k,x)-1} \qquad [2.11]$$

k \ x	1	2	3	4	5	6	7	8	9
2	0.500	0.200	0.125	0.090	0.071	0.058	0.050	0.043	0.038
3	0.500	0.250	0.125	0.111	0.111	0.071	0.071	0.071	0.071
4	0.500	0.250	0.166	0.111	0.111	0.076	0.076	0.076	0.076

Table 2.3. *Numerical values of ε according to Theorem 2.6*

Finally, since an ε-approximation for the k-criteria $TSP(1,2)$ is also an ε-approximation for the i-criteria $TSP(1,2)$ with $i = 2, \ldots, k-1$, we can apply $k-1$ times the inequality [2.11] and the result follows. ∎

Table 2.3 illustrates Theorem 2.6 for some values of k and x. From Theorem 2.6, we are able to give a more explicit but less powerful result.

COROLLARY 2.3.– For any $k \geqslant 2$, any ε-approximate Pareto curve for the k-criteria $TSP(1,2)$ containing at most x solutions satisfies:

$$\varepsilon \geqslant \frac{1}{(2k-1)\left(x(k-1)!\right)^{1/(k-1)} - 1}$$

Proof. By construction of $r(k,x) = r$, we have $x \geqslant \binom{r(k,x)-1+k-1}{k-1}$. Since:

$$\binom{r-1+k-1}{k-1} \geqslant \frac{r^{k-1}}{(k-1)!}$$

we deduce:

$$r \leqslant \left(x(k-1)!\right)^{1/(k-1)}$$

Thus, using the inequality [2.11], we obtain the expected result. ∎

More generally, if we write $R_k(x) = \left((2k-1)(x(k-1)!)^{1/(k-1)} - 1\right)^{-1}$, then we observe that the following property holds : $\forall k \geqslant 2, \exists x_0, \forall x \geqslant x_0$ we have $R_{k+1}(x) \geqslant R_k(x)$. In other words, between two different versions of the k-criteria $TSP(1,2)$, the negative bound increases with k. So, these bounds are interesting when k is a fixed constant and x is an arbitrarily large integer (indeed, when $k = o(x)$).

On the other hand, we can also obtain other bounds when x is fixed and k grows to infinity ($x = o(k)$). In particular, when the ε-approximate Pareto curve contains just x solutions, we obtain $\varepsilon \geqslant 1/(2x-1) - \delta$ for the k-criteria $TSP(1,2)$ with k arbitrarily large.

THEOREM 2.7.– *For any $k \geqslant 2$, any ε-approximate Pareto curve for the k-criteria $TSP(1,2)$ containing at most $x \leqslant k$ solutions satisfies:*

$$\varepsilon \geqslant 1 - \frac{1}{\lceil k/x \rceil}$$

Proof. Let x be an integer smaller than or equal to k and consider the instance $I_{n,r}$ used in Lemma 2.10 with $n \geqslant 2k+1$ and $r = 1$. The instance $I_{n,r}$ admits (at least) k Pareto optimal tours denoted by T_j (the tour T_j only uses edges with a 1 on co-ordinate j and a 2 on the others). By the construction of $I_{n,r}$, we know that $\vec{D}_i(T_j) = 2n$ when $i \neq j$ and $\vec{D}_i(T_j) = n$ otherwise.

Now, consider an ε-approximate Pareto curve that contains at most x solutions. One of these x solutions, denoted by T', approximates at least $p = \lceil \frac{k}{x} \rceil$ Pareto optimal tours. Without loss of generality, we suppose that T' ε-approximates the tours T_1, T_2, $\ldots T_p$:

$$1 + \varepsilon \geqslant \max_{i=1,\ldots,k \text{ and } j=1,\ldots p} \frac{\vec{D}_i(T')}{\vec{D}_i(T_j)}$$

Since $\vec{D}_i(T_j) = 2n$ when $j \in \{1, \ldots, p\}$ and $p < i \leqslant k$, we get:

$$\max_{i=1,\ldots,k \text{ and } j=1,\ldots p} \frac{\vec{D}_i(T')}{\vec{D}_i(T_j)} = \max_{i=1,\ldots,p \text{ and } j=1,\ldots p} \frac{\vec{D}_i(T')}{\vec{D}_i(T_j)}$$

Since $\vec{D}_j(T_j) = n$ when $j \in \{1, \ldots, p\}$, we get:

$$\max_{i=1,\ldots,p \text{ and } j=1,\ldots p} \frac{\vec{D}_i(T')}{\vec{D}_i(T_j)} = \frac{1}{n} \left(\max_{i=1,\ldots,p} \vec{D}_i(T') \right)$$

Then, we have:

$$1 + \varepsilon \geqslant \frac{1}{n} \left(\max_{i=1,\ldots,p} \vec{D}_i(T') \right) \qquad [2.12]$$

We know that any feasible tour T of $I_{n,r}$ satisfies:

$$\sum_{i=1}^{k} \vec{D}_i(T) = n(2k-1)$$

Indeed, the distance of each edge of the instance has exactly one co-ordinate equal to 1. Thus, any tour T satisfies:

$$\sum_{i=1}^{p} \vec{D}_i(T) \geqslant \sum_{i=1}^{k} \vec{D}_i(T) - 2n(k-p) = n(2p-1)$$

In particular, we can observe that:

$$\max_{i=1,\ldots,p} \vec{D}_i(T') \geqslant \frac{\sum_{i=1}^{p} \vec{D}_i(T')}{p} \geqslant n(2 - \frac{1}{p}) \qquad [2.13]$$

Thus, inequalities [2.12] and [2.13] give $1 + \varepsilon \geqslant 2 - 1/p$ which means that $\varepsilon \geqslant 1 - (\lceil k/x \rceil)^{-1}$. ∎

The method presented in this section can be applied to several other multi-criteria problems. For instance, it works with problems where all feasible solutions have the same size ($|V|$ for a Hamiltonian cycle, $|V| - 1$ for a spanning tree, etc).

2.4.2. *A nearest neighbor heuristic for the* k*-criteria* $TSP(1,2)$

Adapting the nearest neighbor heuristic to the k-criteria $TSP(1,2)$ gives rise to two questions: how can we translate the notion of closeness when multiple objectives are considered? How many solutions must be generated to get an approximation of the Pareto curve? In what follows, we propose a way which simultaneously brings an answer to both questions. Given the problem, the total distance of a Pareto optimal tour T^* is enclosed in a k-dimensional distance space. The way to generate a tour T which approximates T^* and also the notion of closeness depends on where $\vec{D}(T^*)$ is located in the distance space. The idea is to partition the distance space into a fixed number of parts. Then, with each part we associate an appropriate notion of closeness. Given a part and its proper notion of closeness, we can generate with the nearest neighbor rule a tour which approximates any Pareto optimal solution whose total distance is in this part. For any instance of the k-criteria $TSP(1,2)$, we propose to divide the distance space into $k!$ parts as follows: each part is identified by a permutation of $\{1,\ldots,k\}$. Given a permutation L of $\{1,\ldots,k\}$, a tour T is in the part identified by L if $\vec{D}_{L(1)}(T) \leqslant \ldots \leqslant \vec{D}_{L(k)}(T)$. For the notion of closeness, we introduce a preference relation over all possible distance vectors, which looks like a lexicographic order. This preference relation which depends on L (denoted by $\prec_L$) is defined by using $k+1$ sets $S_1,\ldots,S_{k+1}$:

$$\begin{aligned} S_q &= \{\vec{a} \in \{1,2\}^k \mid \forall j \leqslant k+1-q \quad \vec{a}_{L(j)} = 1\}, \text{ for } 1 \leqslant q \leqslant k \\ S_{k+1} &= \{1,2\}^k \end{aligned}$$

KNN: k-criteria Nearest Neighbor
$P := \emptyset$;
For each permutation L of $\{1, 2, \ldots, k\}$ do
  Take arbitrarily $v \in V$;
  $W := \{v\}$; $u := v$;
  While $W \neq V$ do
    Take $r \in V \backslash W$ s.t. r is the closest vertex to u by $\preccurlyeq_L$;
    $W := W \cup \{r\}$;
    $p(u) := r$; $u := r$;
  End While ;
  $p(r) := v$;
  $P := P \cup \{p\}$;
End do ;
Return P ;

Table 2.4. *For $v \in V$ and p a tour, $p(v)$ denotes the node which immediately follows v in p*

COROLLARY 2.4.– For any edge e, we say that e is S_q*-preferred* (for $\prec_L$) if $\vec{d}(e) \in S_q \backslash S_{q-1}$ (where $S_0 = \emptyset$). For two edges e and e' such that e is S_q-preferred and e' is $S_{q'}$-preferred, we say that $\vec{d}(e)$ is preferred (resp., weakly preferred) to $\vec{d}(e')$ and we note $\vec{d}(e) \prec_L \vec{d}(e')$ (resp., $\vec{d}(e) \preccurlyeq_L \vec{d}(e')$) if and only if $q < q'$ (resp., $q \leqslant q'$).

An example where $k = 3$ and L is the identity permutation is given in Figure 2.8.

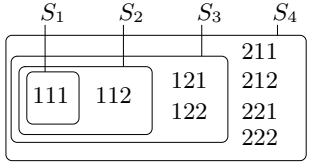


Figure 2.8. *An example with $k = 3$*

The algorithm that we propose for the k-criteria $TSP(1,2)$ is given in Table 2.4. Called KNN for k-criteria Nearest Neighbor, it is composed of $k!$ steps. A permutation L of $\{1, 2, \ldots, k\}$ is determined at each step. With a permutation L, we build a preference relation $\prec_L$ and finally, a solution is greedily generated with the nearest neighbor rule.

Analysis of KNN**.** We prove that KNN returns a $(k-1)/(k+1)$-approximate Pareto curve for the k-criteria $TSP(1,2)$ when $k \geqslant 3$. The proof of this result requires some notations and intermediate lemmata.

In what follows, we consider two particular tours p and p^*. We assume that p is the tour generated by KNN with the preference relation $\prec_L$ and that p^* is a Pareto optimal tour satisfying:

$$\vec{D}_{L(1)}(p^*) \leqslant \vec{D}_{L(2)}(p^*) \leqslant \ldots \leqslant \vec{D}_{L(k)}(p^*) \qquad [2.14]$$

The set of all possible distance vectors $\{1,2\}^k$ is denoted by Ω. For every $j \leqslant k$, we introduce $U_j = \{\vec{a} \in \Omega \mid \vec{a}_j = 1\}$ and $\overline{U}_j = \{\vec{a} \in \Omega \mid \vec{a}_j = 2\}$. For $\vec{a} \in \Omega$, we note $X_{\vec{a}} = \{v \in V \mid \vec{d}([v, p(v)]) = \vec{a}\}$ and $X^*_{\vec{a}} = \{v \in V \mid \vec{d}([v, p^*(v)]) = \vec{a}\}$. Finally, $x_{\vec{a}}$ (resp. $x^*_{\vec{a}}$) denotes the cardinality of $X_{\vec{a}}$ (resp. $X^*_{\vec{a}}$).

If n is the number of vertices, then by construction we have $\sum_{\vec{a}\in\Omega} x_{\vec{a}} = \sum_{\vec{a}\in\Omega} x^*_{\vec{a}} = n$, $\vec{D}_j(p) = 2n - \sum_{\vec{a}\in U_j} x_{\vec{a}}$ and $\vec{D}_j(p^*) = 2n - \sum_{\vec{a}\in U_j} x^*_{\vec{a}}$.

LEMMA 2.11.– *The following holds for any $q \leqslant k$:*

$$2 \sum_{\vec{a}\in\bigcap_{j=1}^{k+1-q} U_{L(j)}} x_{\vec{a}} \geqslant \sum_{\vec{a}\in\bigcap_{j=1}^{k+1-q} U_{L(j)}} x^*_{\vec{a}}$$

Proof. We define $F_q = \{v \in V \mid \vec{d}([v, p(v)]) \in S_q\}$ and $F^*_q = \{v \in V \mid \vec{d}([v, p^*(v)]) \in S_q\}$. Then, we have to prove that $2|F_q| \geqslant |F^*_q|$. The key result is to see that $p^*[F^*_q \backslash F_q] \subseteq F_q$ where $p^*[W] = \bigcup_{v\in W}\{p^*(v)\}$. Take a vertex v in $F^*_q \backslash F_q$ (see Figure 2.9). Then, $\vec{d}([v, p^*(v)]) \in S_q$, $\vec{d}([v, p(v)]) \in S_{q'}$ and $q' > q$. During the calculation of p, suppose that v is the current node and that $p^*(v)$ is not already visited. We get a contradiction (the nearest neighbor rule is violated) since $p(v)$ immediately follows v in p and $\vec{d}([v, p^*(v)]) \prec_L \vec{d}([v, p(v)])$. Now, suppose that $p^*(v)$ was already visited. It directly precedes $p(p^*(v))$ in p and then $\vec{d}([p*(v), p(p^*(v))]) \preccurlyeq_L \vec{d}([v, p^*(v)])$. As a consequence, $\vec{d}([p^*(v), p(p^*(v))]) \in S_{q''}$ such that $q'' \leqslant q$ and $p^*(v) \in F_q$ since $S_{q''} \subseteq S_q$.

Since $|p^*[F^*_q \backslash F_q]| = |F^*_q \backslash F_q|$, $|F^*_q| = |F^*_q \backslash F_q| + |F^*_q \cap F_q|$ and $|F_q| \geqslant |F^*_q \cap F_q|$, we deduce $|F^*_q| = |p^*[F^*_q \backslash F_q]| + |F^*_q \cap F_q| \leqslant 2|F_q|$. Finally, since $\bigcap_{j=1}^{k+1-q} U_{L(j)} = S_q$, $|F_q| = \sum_{\vec{a}\in S_q} x_{\vec{a}}$ and $|F^*_q| = \sum_{\vec{a}\in S_q} x^*_{\vec{a}}$, the result follows. ∎

The following inequality is equivalent to [2.14]:

$$\sum_{\vec{a}\in U_{L(1)}} x^*_{\vec{a}} \geqslant \sum_{\vec{a}\in U_{L(2)}} x^*_{\vec{a}} \geqslant \ldots \geqslant \sum_{\vec{a}\in U_{L(k)}} x^*_{\vec{a}}$$

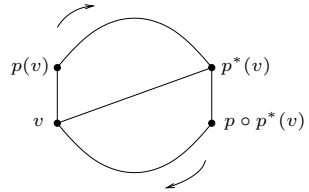


Figure 2.9. *The tour p generated by* KNN. *The edge $[v, p^*(v)]$ belongs to p^**

We easily deduce that for any couple j_1, j_2 such that $j_1 < j_2$ we have:

$$\sum_{\vec{a} \in U_{L(j_2)} \setminus U_{L(j_1)}} x^*_{\vec{a}} \leqslant \sum_{\vec{a} \in U_{L(j_1)} \setminus U_{L(j_2)}} x^*_{\vec{a}} \qquad [2.15]$$

Let b_1, b_2, j and m be such that $b_1 \in \{1,2\}$, $b_2 \in \{1,2\}$, $1 \leqslant j \leqslant k$ and $1 \leqslant m < j$. Let $R(b_1, j, m, b_2)$ be the set of all $\vec{a} \in \Omega$ such that $\vec{a}_{L(j)} = b_1$ and there exists exactly m distinct co-ordinates of $\vec{a}$ among $\{\vec{a}_{L(1)}, \vec{a}_{L(2)}, \ldots, \vec{a}_{L(j-1)}\}$ which are equal to b_2. Note that $R(b_1, j, m, b_2) = R(b_1, j, j-1-m, \overline{b_2})$ where $\overline{b_2} = 3 - b_2$.

LEMMA 2.12.– *For any $j \leqslant k$, we have:*

$$\sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(1,j,q,2) \cup R(2,j,q,2)} x^*_{\vec{a}} \right) \leqslant (j-1) \sum_{q=0}^{j-1} \left(\sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right)$$

Proof. We sum up inequality [2.15] with $j_1 \in \{1, \ldots, j-1\}$ and $j_2 = j$. We get the following inequality:

$$\sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in U_{L(j)} \setminus U_{L(q)}} x^*_{\vec{a}} \right) \leqslant \sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in U_{L(q)} \setminus U_{L(j)}} x^*_{\vec{a}} \right) \qquad [2.16]$$

We also have the following equality:

$$\forall j \leqslant k, \sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in U_{L(j)} \setminus U_{L(q)}} x^*_{\vec{a}} \right) = \sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(1,j,q,2)} x^*_{\vec{a}} \right) \qquad [2.17]$$

Let $\vec{a}$ be a distance vector in $R(1,j,q,2)$. By definition, $\vec{a}_{L(j)} = 1$ and there exists a set $\{i_1,\ldots,i_q\}$ with $1 \leqslant i_1 < i_2 < \ldots < i_q < j$ such that $\vec{a}_{L(i_1)} = \vec{a}_{L(i_2)} = \ldots = \vec{a}_{L(i_q)} = 2$. Moreover, for all $j' \leqslant j-1$ such that $j' \notin \{i_1,\ldots,i_q\}$, we have $\vec{a}_{L(j')} = 1$. Thus, $\vec{a} \in U_{L(j)} \backslash U_{L(g)}$ iff $g \in \{i_1, i_2, \ldots, i_q\}$.

Using a similar argument, we obtain:

$$\forall j \leqslant k,\ \sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in U_{L(q)} \backslash U_{L(j)}} x^*_{\vec{a}} \right) = \sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right) \qquad [2.18]$$

Then, using [2.16], [2.17] and [2.18] we get:

$$\sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(1,j,q,2)} x^*_{\vec{a}} \right) \leqslant \sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right) \qquad [2.19]$$

Since $R(2,j,q,2) = R(2,j,j-1-q,1)$, the following equality holds:

$$\begin{aligned} \sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right) &= (j-1) \sum_{q=0}^{j-1} \left(\sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right) \\ &\quad - \sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(2,j,q,2)} x^*_{\vec{a}} \right) \end{aligned} \qquad [2.20]$$

So, Lemma 2.12 follows from [2.19] and [2.20]. ∎

THEOREM 2.8.– KNN *returns a* $(k-1)/(k+1)$*-approximate Pareto curve for the* k*-criteria* $TSP(1,2)$ *when* $k \geqslant 3$.

Proof. In what follows, we consider that L is any permutation of $\{1,\ldots,k\}$, p^* is a Pareto optimal tour satisfying [2.14] and p is built with the nearest neighbor rule and the preference relation $\prec_L$. Then, we have to show that if $j \geqslant 3$, then $\vec{D}_{L(j)}(p) \leqslant (1 + \frac{j-1}{j+1})\vec{D}_{L(j)}(p^*)$.

The previous inequality holds if we have the following inequality:

$$-(j+1) \sum_{\vec{a} \in U_{L(j)}} x_{\vec{a}} \leqslant 2(j-1) \sum_{\vec{a} \in \overline{U}_{L(j)}} x^*_{\vec{a}} - 2 \sum_{\vec{a} \in U_{L(j)}} x^*_{\vec{a}} \qquad [2.21]$$

$$\begin{aligned}
\vec{D}_{L(j)}(p) \leqslant \frac{2j}{j+1}\vec{D}_{L(j)}(p^*) \quad &\Leftrightarrow \quad (j+1)\left(2n - \sum_{\vec{a}\in U_{L(j)}} x_{\vec{a}}\right) \\
&\leqslant 2j\left(2n - \sum_{\vec{a}\in U_{L(j)}} x^*_{\vec{a}}\right) \\
&\Leftrightarrow \quad -(j+1)\sum_{\vec{a}\in U_{L(j)}} x_{\vec{a}} \\
&\leqslant 2(j-1)n - 2j\sum_{\vec{a}\in U_{L(j)}} x^*_{\vec{a}} \\
&\Leftrightarrow \quad -(j+1)\sum_{\vec{a}\in U_{L(j)}} x_{\vec{a}} \\
&\leqslant 2(j-1)\sum_{\vec{a}\in \overline{U}_{L(j)}} x^*_{\vec{a}} - 2\sum_{\vec{a}\in U_{L(j)}} x^*_{\vec{a}}
\end{aligned}$$

using $n = \sum_{\vec{a}\in U_{L(j)}} x^*_{\vec{a}} + \sum_{\vec{a}\in \overline{U}_{L(j)}} x^*_{\vec{a}}$.

Let us denote by $\mathcal{A}$ and $\mathcal{B}$ the following quantities:

$$\begin{aligned}
\sum_{\vec{a}\in U_{L(j)}} x_{\vec{a}} &= \sum_{\vec{a}\in U_{L(j)}\setminus(\bigcap_{m\leqslant j-1} U_{L(m)})} x_{\vec{a}} + \sum_{\vec{a}\in\bigcap_{m\leqslant j} U_{L(m)}} x_{\vec{a}} = \mathcal{A} \\
\sum_{\vec{a}\in U_{L(j)}} x^*_{\vec{a}} &= \sum_{\vec{a}\in U_{L(j)}\setminus(\bigcap_{m\leqslant j-1} U_{L(m)})} x^*_{\vec{a}} + \sum_{\vec{a}\in\bigcap_{m\leqslant j} U_{L(m)}} x^*_{\vec{a}} = \mathcal{B}
\end{aligned}$$

Then, inequality [2.21] becomes:

$$-(j+1)\mathcal{A} \leqslant 2(j-1)\sum_{\vec{a}\in\overline{U}_{L(j)}} x^*_{\vec{a}} - 2\mathcal{B} \qquad [2.22]$$

In order to prove [2.22], we propose the following decomposition:

$$\begin{aligned}
\mathcal{C} \quad = \quad & 2(j-1)\sum_{\vec{a}\in\overline{U}_{L(j)}} x^*_{\vec{a}} - 2\sum_{\vec{a}\in U_{L(j)}\setminus\bigcap_{m\leqslant j-1} U_{L(m)}} x^*_{\vec{a}} \\
& - 4\sum_{\vec{a}\in\bigcap_{m\leqslant j} U_{L(m)}} x_{\vec{a}} \qquad [2.23]
\end{aligned}$$

$$-(j+1)\mathcal{A} \leqslant \mathcal{C} \qquad [2.24]$$

$$\mathcal{C} \leqslant 2(j-1) \sum_{\vec{a} \in \overline{U}_{L(j)}} x^*_{\vec{a}} - 2\mathcal{B} \qquad [2.25]$$

thus, [2.24] becomes:

$$-(j+1) \sum_{\vec{a} \in U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)}} x_{\vec{a}} - (j-3) \sum_{\vec{a} \in \bigcap_{m \leqslant j} U_{L(m)}} x_{\vec{a}} \leqslant$$
$$\leqslant 2(j-1) \sum_{\vec{a} \in \overline{U}_{L(j)}} x^*_{\vec{a}} - 2 \sum_{\vec{a} \in U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)}} x^*_{\vec{a}}$$

Since the left part of this inequality is negative, we want to prove that the right part is positive:

$$0 \leqslant 2(j-1) \sum_{\vec{a} \in \overline{U}_{L(j)}} x^*_{\vec{a}} - 2 \sum_{\vec{a} \in U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)}} x^*_{\vec{a}} \qquad [2.26]$$

$$\sum_{\vec{a} \in U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)}} x^*_{\vec{a}} \leqslant (j-1) \sum_{\vec{a} \in \overline{U}_{L(j)}} x^*_{\vec{a}} \qquad [2.27]$$

We also have:

$$\sum_{\vec{a} \in U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)}} x^*_{\vec{a}} = \sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in R(1,j,q,2)} x^*_{\vec{a}} \right) \text{ and}$$

$$(j-1) \sum_{\vec{a} \in \overline{U}_{L(j)}} x^*_{\vec{a}} = (j-1) \sum_{q=0}^{j-1} \left(\sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right)$$

The first equality follows from $U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)} = \bigcup_{q=1}^{j-1} R(1,j,q,2)$ since $\vec{a} \in U_{L(j)} \setminus \bigcap_{m \leqslant j-1} U_{L(m)}$ if and only if $\vec{a}_{L(j)} = 1$ and there are exactly q indices

$\{i_1, \ldots i_q\}$ such that $1 \leqslant q \leqslant j-1$ and $\vec{a}_{L(i_1)} = \vec{a}_{L(i_2)} = \ldots = \vec{a}_{L(i_q)} = 2$, which is equivalent to $\vec{a} \in R(1,j,q,2)$. The second equality follows from $\overline{U}_{L(j)} = \bigcup_{q=0}^{j-1} R(2,j,q,1)$ because $\vec{a} \in \overline{U}_{L(j)}$ means $\vec{a}_{L(j)} = 2$.

As a consequence, [2.27] becomes:

$$\sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in R(1,j,q,2)} x^*_{\vec{a}} \right) \leqslant (j-1) \sum_{q=0}^{j-1} \left(\sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right)$$

With Lemma 2.12, we have:

$$\sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(1,j,q,2) \cup R(2,j,q,2)} x^*_{\vec{a}} \right) \leqslant (j-1) \sum_{q=0}^{j-1} \left(\sum_{\vec{a} \in R(2,j,q,1)} x^*_{\vec{a}} \right)$$

and [2.27] follows from

$$\sum_{q=1}^{j-1} \left(q \sum_{\vec{a} \in R(1,j,q,2) \cup R(2,j,q,2)} x^*_{\vec{a}} \right) \geqslant \sum_{q=1}^{j-1} \left(\sum_{\vec{a} \in R(1,j,q,2)} x^*_{\vec{a}} \right)$$

By Lemma 2.11 with $q = k+1-j$ we have:

$$2 \sum_{\vec{a} \in \bigcap_{m \leqslant j} U_{L(m)}} x_{\vec{a}} \geqslant \sum_{\vec{a} \in \bigcap_{m \leqslant j} U_{L(m)}} x^*_{\vec{a}}$$

which is exactly [2.25]. ∎

2.5. Conclusion

Negative results for multi-criteria optimization problems were not extensively investigated though their approximability motivated a lot of articles. By connecting the size of the approximate Pareto curve and the best approximation ratio which can be achieved, we present a way to get negative results which do not rely on **NP**-hardness. We applied the method to the k-criteria $TSP(1,2)$ but it also works with problems where all feasible solutions have the same size.

The approximability of the k-criteria $TSP(1,2)$ is also investigated with multi-criteria versions of the traditional local search and the nearest neighbor heuristics. However, as the number of criteria grows, and even though the number of solutions is large ($k!$), the approximation ratio of KNN tends to 2. Then, it would be interesting to reduce the gap between positive and negative results. Following this direction, Manthey *et al.* [MAN 06] recently considered randomized algorithms for several particular cases of the multi-criteria traveling salesman problem.

2.6. Bibliography

[ANG 04] ANGEL E., BAMPIS E., GOURVÈS L., "Approximating the Pareto curve with local search for the bi-criteria TSP(1,2) problem", *Theoretical Computer Science*, vol. 310, p. 135-146, 2004.

[ANG 05] ANGEL E., BAMPIS E., GOURVÈS L., MONNOT J., "(Non)-Approximability for the Multi-criteria TSP(1, 2)", *Proc. of FCT'05, LNCS*, vol. 3623, p. 329-340, 2005.

[ANG 05b] ANGEL E., BAMPIS E., GOURVÈS L., MONNOT J., "A Note on the Bicriteria $TSP(1,2)$", *Technical Report*, 2005.

[BER 06] BERMAN P., KARPINSKI M., "$8/7$-Approximation Algorithm for (1,2)-TSP", *Proc. of SODA'2006*, p. 641-648, 2006.

[CHR 76] CHRISTOFIDES N., "Worst-Case analysis of a new heuristic for the traveling salesman problem", *Technical Report, GSIA*, Carnegie Mellon University, 1976.

[CRO 58] CROES A., "A method for solving traveling-salesman problems", *Operations Research*, vol. 5, p. 791-812, 1958.

[DEN 02] DENG X., PAPADIMITRIOU C. H., SAFRA S., "On the Complexity of Equilibria" *in Proc. of STOC'02*, p. 67-71, 2002.

[EHR 00] EHRGOTT M., "Multicriteria optimization", Springer-Verlag, 2000.

[ENG 01] ENGEBRETSEN L., KARPINSKI M., "Approximation Hardness of TSP with Bounded Metrics" *Proc. of ICALP'2001*, p. 201-212, 2001.

[HAS 00] HASSIN R., RUBINSTEIN S., "Better approximations for Max TSP", *Information Processing Letters*, vol. 75, p. 181-186, 2000.

[HAS 02] HASSIN R., RUBINSTEIN S., "A 7/8-approximation algorithm for metric Max TSP", *Information Processing Letters*, vol. 81, p. 247-251, 2002.

[JOH 85] JOHNSON D.S., PAPADIMITRIOU C. H., "Performance guarantees for heuristics", in *The Traveling Salesman Problem: a guided tour of Combinatorial Optimization*, LAWLER E.L, LENSTRA J.K, RINNOOY KAN A.H.G. and SHMOYS D.B (eds.), Wiley Chichester, p. 145-180, 1985.

[KHA 98] KHANNA S., MOTWANI R., SUDAN M., VAZIRANI V., "On syntactic versus computational views of approximability", *SIAM Journal on Computing*, vol. 28, p. 164-191, 1998.

[LUC 92] LUCAS D.E., "Récréations mathématiques Vol. II", Gauthier Villars, Paris, 1892.

[MAN 06] MANTHEY B., SHANKAR RAM L., "Approximation Algorithms for Multi-criteria Traveling Salesman Problems", *Proc. of WAOA'06, LNCS*, vol. 4368, p. 304-317, 2006.

[MON 02] MONNOT J., "Differential approximation results for the traveling salesman and related problems", *Information Processing Letters*, vol. 82, p. 229-235, 2002.

[MON 02b] MONNOT J., PASCHOS V. TH., TOULOUSE S., "Approximation algorithms for the traveling salesman problem", *Mathematical Models of Operations Research*, vol. 56, p. 387-405, 2002.

[MON 03] MONNOT J., PASCHOS V. TH., TOULOUSE S., "Differential approximation results for the traveling salesman problem with distances 1 and 2", *European Journal of Operational Research*, vol. 145, p. 557-568, 2003.

[PAP 93] PAPADIMITRIOU C. H., YANNAKAKIS M., "The traveling salesman problem with distances one and two", *Mathematics of Operations Research*, vol. 18(1), p. 1-11, 1993.

[PAP 00] PAPADIMITRIOU C. H., YANNAKAKIS M., "On the approximability of trade-offs and optimal access of web sources", *Proc. of FOCS'2000*, p. 86-92, 2000.

[ROS 77] ROSENKRANTZ D. J., STEARNS R. E , LEWIS II P. M, "An analysis of several heuristics for the traveling salesman problem", *SIAM J. Comp.*, vol. 6, p. 563-581, 1977.

[VAS 05] VASSILVITSKII S., YANNAKAKIS M., "Efficiently computing succinct trade-off curves", *Theoretical Computer Science*, vol. 348, p. 334-356, 2005.

[WAR 87] WARBURTON A., "Approximation of Pareto optima in multiple-objective shortest path problems", *Operations Research*, vol. 35(1), p. 70-79, 1987.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 3

Online Models for Set-covering: the Flaw of Greediness

3.1. Introduction

Let C be a ground set of n elements and $\mathcal{S}$ a family of m subsets of C such that $\cup_{S\in\mathcal{S}} S = C$. The set-covering problem consists of finding a family $\mathcal{S}' \subseteq \mathcal{S}$, of minimum cardinality, such that $\cup_{S\in\mathcal{S}'} S = C$. In what follows, for an element $\sigma_i \in C$, we set $F_i = \{S_i^j \in \mathcal{S} : \sigma_i \in S_i^j\}$ and $f_i = |F_i|$; also, we set $f = \max\{f_i : i = 1, \ldots, n\}$.

The set-covering problem has been extensively studied over the past decades. It has been shown to be **NP**-hard in Karp's seminal paper ([KAR 72]) and $O(\log n)$-approximable for both weighted and unweighted cases (see [CHV 79] for the former and [JOH 74, LOV 75, SLA 96] for the latter; see also [PAS 97] for a comprehensive survey on the subject). This approximation ratio is the best achievable, unless **P** = **NP** ([RAZ 97]).

In *online* calculation, we can assume that the instance is not known in advance but it is revealed step-by-step. Upon arrival of new data, we have to decide irrevocably which of these data are to be included in the solution under construction. The fact that the instance is not known in advance, gives rise to several online models specified by the ways in which the final instance is revealed, or by the amount of information that is achieved by the online algorithm at each step. In any of these models, we have to devise algorithms, called online algorithms, constructing feasible solutions whose

Chapter written by Giorgio AUSIELLO, Aristotelis GIANNAKOS and Vangelis Th. PASCHOS.

values are as close as possible to optimal offline values, i.e., to values of optimal solutions assuming that the final instance is completely known in advance. The closeness of an online solution to an optimal offline one is measured by the so-called competitive ratio $m(x, y)/\operatorname{opt}(x)$, where x is an instance of the problem dealt, y the solution calculated by the online algorithm dealt, $m(x, y)$ its value and $\operatorname{opt}(x)$ the value of an optimal offline solution. This measure for online calculation has been introduced in [SLE 85].

Informally, the basic online set-covering model adopted here is the following: elements of a ground set of size n arrive one-by-one and, with any such element σ_i (arriving during step i), arrives also something about some of the sets containing σ_i. This "something" can be either the names of sets containing σ_i or more information related to their cardinalities, or their covering potential, etc. At each step, σ_i has to be processed immediately, i.e., it has either to be covered by some of the sets whose names have arrived with it, or to be left uncovered thus risking infeasibility of the final solution.

We first assume that, together with an element, the names of the sets containing it in the final instance are revealed. We show that if no further information is given, then the competitive ratio of any algorithm is $\Omega(n)$.

Next, we handle several algorithms dealing with the online model just sketched. The first one, called `TAKE-ALL`, includes at each step in the solution all the sets containing the element just revealed, if it is still uncovered. We show that this algorithm has tight competitive ratio $O(f)$, where f is the maximum number of sets in $\mathcal{S}$ that contain a ground element. The second algorithm, called `TAKE-AT-RANDOM`, is a randomized algorithm that at each step picks a set at random among those whose names are revealed and includes it in the solution, if it is still uncovered. For this algorithm we provide an upper bound of $O(n)$ for its competitiveness, as well as an asymptotically matching expected lower bound.

We then assume that, together with the names of the sets covering a revealed element, information concerning their covering potential is also communicated. We show that, in this case, the competitive ratio of any algorithm that, for any arriving uncovered element, includes in the cover at least one set containing it is bounded below by $\Omega(\sqrt{n})$. Assuming that the covering potential information given with any element is the name of the larger set containing it (in the final instance), we show that the competitive ratio of the greedy algorithm, called `TAKE-LARGEST`, consisting of including this set in the solution is $O(n)$.

We next address the following question: "what must an online algorithm know about sets in order to guarantee an upper competitive bound of $O(\sqrt{n})$?" We show that such a bound can be attained at least by an algorithm, called `TAKE-LARGEST-ON-FUTURE-ITEMS`, which at any step i includes some set containing σ_i covering most of the

ground set-elements that have not been yet covered (clearly, any uncovered element is yet unrevealed). This assumption can be seen as the online counterpart of the natural greedy (offline) set-covering algorithm, called GREEDY in what follows. We recall that this algorithm includes in the solution a set covering most of the still uncovered elements.

Finally, we address a budget variant of set-covering. We assume that two algorithms collaborate to solve the problem. The application cost of the former is just the cardinality of the solution it finally calculates, while, for the latter, its application cost is the cardinality of its solutions augmented by an overhead due, for example, to the fact that it is allowed to wait before making its decisions. We can see the application cost of the former algorithm as a kind of budget allotted to them that is not allowed to be exceeded. The objective is to perform the maximum possible saving upon the initial budget. We show that there exists a natural algorithm-cost model such that GREEDY is asymptotically optimal for maximum budget saving when the budget allotted is the application cost of TAKE-LARGEST-ON-FUTURE-ITEMS.

3.2. Description of the main results and related work

In [ALO 03], the following online set-covering model has been studied. We suppose that we are given an instance $(\mathcal{S}, C)$ that is known in advance, but it is possible that only a part of it, i.e., a subinstance $(\mathcal{S}_p, C_p)$ of $(\mathcal{S}, C)$ will finally arrive; this subinstance is not known in advance. A picturesque way of understanding the model is to think of the elements of C as lights initially switched off. Elements switch on (get activated) one-by-one. Any time an element c gets activated, the algorithm has to decide which among the sets of $\mathcal{S}$ containing c has to be included in the solution under construction (since we assume that $(\mathcal{S}, C)$ is known in advance, all these sets are also known). In other words, the algorithm has to keep an online cover for the activated elements. The algorithm proposed for this model achieves competitive ratio $O(\log n \log m)$ (even if less than n elements of C will be finally switched on and less than m subsets of $\mathcal{S}$ include these elements).

The online model dealt with here is inspired (yet quite different since it does not allow an element to disappear once arrived) from that in [ALO 03]. Moreover, and most importantly, the instance $(\mathcal{S}, C)$ is not known in advance. In our model, we are given an arrival sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of the elements of C (i.e., elements of C are switched on following the order $\sigma_1, \ldots, \sigma_n$), the objective is to find, for any $i \in \{1, \ldots, n\}$, a family $\mathcal{S}'_i \subseteq \mathcal{S}$ such that $\{\sigma_1, \ldots, \sigma_i\} \subseteq \cup_{S \in \mathcal{S}'_i} S$. Recall that for any σ_i, $i = 1, \ldots, n$, we denote by $F_i = \{S_i^j : S_i^j \in \mathcal{S}, \sigma_i \in S_i^j\}$ the sets of $\mathcal{S}$ containing σ_i, by f_i the cardinality of F_i, usually called frequency, and we set $f = \max_{\sigma_i \in \Sigma}\{f_i\}$. We denote by $\bar{S}_i^j$ the subset of the elements of $S_i^j \in F_i$ still remaining uncovered and by δ_i^j the cardinality of $\bar{S}_i^j$.

When σ_i switches on, something about the sets in F_i, $j = 1, \ldots, f_i$ is revealed. We first assume that this "something" is the names of the sets covering σ_i, i.e., the names in F_i. We show that, if no further information is supplied, then no online algorithm can achieve competitive ratio better than $O(n)$.

We then study the competitiveness of two algorithms dealing with this model, namely `TAKE-ALL` and `TAKE-AT-RANDOM`.

The former, whenever some σ_i arrives, if it is still uncovered (i.e., if none of the sets included in the solution up-to-date does belong to F_i), then includes all the sets in F_i in the solution under construction. We show that the competitive ratio of `TAKE-ALL` is bounded above by f, i.e., the maximum frequency of the final instance, and that this ratio is tight and can even be exponential with n.

In the case of `TAKE-AT-RANDOM`, any time some σ_i arrives, if it is still uncovered, the algorithm picks at random a set in F_i and puts it in the solution. We show that its competitive ratio is bounded above by $O(n)$. We also prove that its expected competitive ratio cannot be better than $O(n^{1-\epsilon})$, for any $\epsilon > 0$. We so have a lower bound that asymptotically matches its competitive ratio.

We next assume that together with F_i, some more information is revealed about the covering potential of some of the sets in F_i. We show that under this assumption, no online algorithm that, upon the arrival of an uncovered element, processes it by adding to the cover at least one set containing it can guarantee competitive ratio better than $\sqrt{n/2}$.

Furthermore, we show that if together with F_i the name of some set:

$$\hat{S}_i \in \operatorname{argmax}\left\{\left|S_i^j\right|, j = 1, \ldots, f_i\right\}$$

is revealed, then the greedy rule (called `TAKE-LARGEST` later on) that if σ_i is still uncovered, includes $\hat{S}_i$ in the solution, achieves competitive ratio $O(n)$.

The two results mentioned above show that there is an important gap between the general lower bound of $\sqrt{n/2}$ and the ratio achieved by `TAKE-LARGEST`. Hence, a question arises naturally: "what is sufficient for an online algorithm to know in order to achieve a ratio closer to this bound, for example, $O(\sqrt{n})$, respecting always the given online model?". We show that there exists at least one implementation doing this, if it possesses some look-ahead information.

We show that if together with σ_i and F_i, the name of some set:

$$\tilde{S}_i \in \operatorname{argmax}\left\{\delta_i^j, j = 1, \ldots, f_i\right\}$$

i.e., the name of a set in F_i covering most of the still uncovered elements is revealed, then the greedy rule (TAKE-LARGEST-ON-FUTURE-ITEMS) that adds $\tilde{S}_i$ in the cover, if σ_i remains still uncovered, achieves tight competitive ratio $O(\sqrt{n})$. Let us note that TAKE-LARGEST-ON-FUTURE-ITEMS is a kind of online analog of GREEDY. Hence, analysis of its competitiveness is interesting by itself.

Note also that a basic and very interesting feature of the introduced models is their small memory requirements, since the only information needed is the binary encoding of the names of the sets. This is a major difference between our approach and that of [ALO 03]. There, anytime an element gets activated, the algorithm needs to calculate the value of a potential function using an updated weight parameter for each element and then chooses covering sets in a suitable way so that this potential is always non-increasing. The greedy online algorithm in our model needs only a constant number of memory places, making it more appropriate for handling very large instances with very few hardware resources. For instance, the rules used here use at most $O(m)$ space.

We are therefore faced with the power (rather the weakness) of greediness. Let us recall that the online models that we consider assume no knowledge of the topology of the final instance $(\mathcal{S}, C)$ and immediate processing of any arriving element σ_i. Obviously, the rules that we consider are the only ones to feasibly solve the problem in such situations. Furthermore, they are very efficient in time and in memory requirements, hence well adapted to really face online practical situations. On the other hand, since no a priori knowledge of the instance's topology is admitted, no algorithm can do complex preliminary calculations (as the potential calculations in [ALO 03]) in order to judiciously choose the set to be included in the cover under construction.

In many real-life problems, it is necessary to relax the main specification of the online setting, i.e., to keep a solution for any partially revealed instance, in order to achieve a better solution quality. In this sense, a possible relaxation is to consider that several algorithms collaborate in order to return the final solution. The costs of using these algorithms can be different from one another, depending upon the sizes of the solutions calculated, the time overheads they take in order to produce them, etc. Moreover, we can assume that an initial common budget is allotted to all these algorithms and that this budget is large enough to allow use of at least one of the algorithms at hand to solve the problem without exceeding it. A nice objective could be in this case, to use these algorithms in such a way that a maximum of the initial budget is saved.

For the case of set-covering, the following budget-model, giving rise to what we call *maximum budget saving problem* is considered in section 3.7. We assume that two algorithms collaborate to solve it: for example, TAKE-LARGEST-ON-FUTURE-ITEMS and the greedy (offline) algorithm. The application-cost of the former is just the cardinality of the solution it finally calculates, while, for the latter, its application cost

is the cardinality of its solutions augmented by an overhead due, for example, to the fact that it is allowed to wait before making its decisions. For an instance x of set-covering, the initial budget considered is $B(x) = \sqrt{n}\,\mathrm{opt}(x)$ (this is so that at least TAKE-LARGEST-ON-FUTURE-ITEMS is able to calculate a solution of x without exceeding the budget for any x). Denote by $c(x, y)$ the cost of using A in order to calculate a cover y for x. The objective is to maximize the quantity $B(x) - c(x, y)$ and, obviously, the maximum possible economy on x is $B(x) - \mathrm{opt}(x)$. We show in section 3.7 that there exists a natural algorithm-cost model such that GREEDY is asymptotically optimal for maximum budget saving.

Before closing this section, let us quote another approach that could be considered to be at midway between semi-online and reoptimization approaches, developed in [GAM 97]. There, the problem tackled is the maintenance of the approximation ratio achieved by an algorithm while the set-covering instance undergoes limited changes. More precisely, assume that a set-covering instance $(\mathcal{S}, C)$ and a solution $\mathcal{S}'$ for it are given. How many insertions of some of the ground elements in subsets that did not previously contain these elements produce an instance for which the solution $\mathcal{S}'$ of the initial instance guarantees the same approximation ratio in both of them? In [GAM 97] it is shown that if solution $\mathcal{S}'$ has been produced by application of the natural greedy algorithm achieving approximation ratio $O(\log n)$ ([CHV 79]), then after $O(\log n)$ such insertions the initial solution $\mathcal{S}'$ still guarantees the same approximation ratio. In the same spirit lies also the similar set-covering model in [TEL 04].

3.3. The price of ignorance

In this section we consider the first version of the online model sketched in section 3.2. Assume an arrival sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of the elements of C, and the objective is to find, for any $i \in \{1, \ldots, n\}$, a family $\mathcal{S}'_i \subseteq \mathcal{S}$ such that $\{\sigma_1, \ldots, \sigma_i\} \subseteq \cup_{S \in \mathcal{S}'_i} S$. Once an element σ_i, $i = 1, \ldots, n$ switches on, only the encodings of the members of F_i are also revealed.

For this case we first prove that when no additional information is given, any rule that has to cover a new element without any look-ahead behaves rather badly. In this sense strategies that choose elements either randomly or based upon observations of the past (for instance include the subset that has appeared the most frequently or the most rarely until now) are highly inefficient.

PROPOSITION 3.1.– *If no information other than their encodings is given about the sets revealed with an arriving, not yet covered ground element, then the competitive ratio of implementation of any greedy principle is* $\Omega(n)$.

Proof. The adversary reveals a first uncovered element along with the encodings of N sets, say $S_1, \ldots, S_N$, covering it in the final instance. He then keeps revealing uncovered elements along with all sets from $S_1, \ldots, S_N$ which are not already included into the cover, until the algorithm has included all N sets into the cover.

Suppose without loss of generality that the algorithm has included $S_1, \ldots, S_{l_1}$ at the first step, $S_{l_1+1}, \ldots, S_{l_2}$ at the second step, and so on, $\ldots$, $S_{l_{k-1}+1}, \ldots, S_{l_k} = S_N$ at the kth and final step.

The adversary can give the following interpretation to the instance:

– the family of ground subsets is $\mathcal{S} = \{S_1, \ldots, S_N\}$;

– there exist $n = \log N + k$ ground elements, namely $\{\sigma_1, \ldots, \sigma_{\log N}, 1, \ldots, k\}$ (notice also that $k \leqslant N$);

– the set S_i included at step j contains the elements σ_p, for all places p where the binary expression of i has a 1, plus elements in $\{1, \ldots, j\}$;

– the arrival sequence is $1, \ldots, k$.

In this interpretation, the set S_N, included at the last step, is the ground set itself; thus, in such a setting, the competitive ratio of the algorithm would be N, i.e., $\Omega(n)$. ∎

It can be immediately seen that under the given model, any deterministic algorithm that includes a specific set containing a new (uncovered) element σ_i (for example, the set of F_i that comes first in lexicographic order) achieves competitive ratio $O(n)$. Indeed, it chooses at most n sets for an optimum greater than, or equal to 1.

3.4. Competitiveness of `TAKE-ALL` and `TAKE-AT-RANDOM`

3.4.1. `TAKE-ALL` *algorithm*

Recall that the `TAKE-ALL` algorithm, whenever a newly revealed element σ_i is not already covered by sets already included in the solution due to former arrivals, includes in the solution the whole family F_i the names of the members of which have been revealed together with σ_i.

PROPOSITION 3.2.– *The competitive ratio of `TAKE-ALL` is bounded above by f. This ratio is tight.*

Proof. Denote by $\sigma_1, \ldots, \sigma_k$ the critical elements of Σ, i.e., the elements having entailed the introduction of $S_1, \ldots, S_k$ in $\mathcal{S}'$. Denote also by $\mathcal{S}^*$ an optimum offline solution. Obviously, for any of the critical elements, a distinct set is needed to cover it, in any feasible cover for C; hence:

$$|\mathcal{S}^*| \geqslant k \qquad [3.1]$$

On the other hand, since, for $i = 1, \ldots, k$, $f_i \leqslant f$:

$$|\mathcal{S}'| \leqslant kf \qquad [3.2]$$

By combining [3.1] and [3.2], the competitive ratio is immediately derived.

In order to show tightness, consider an instance with ground set $C = \{1, \ldots, n\}$ and the family of all 2^{n-1} sets formed by 1 union any other set in $2^{C\setminus\{1\}}$. With an arrival sequence starting with 1, the competitive ratio of rule TAKE-ALL would be $2^{n-1} = f$. ∎

Note that TAKE-ALL is similar to the approximation algorithm for minimum set-covering presented in [HOC 82] and, furthermore, it guarantees the same approximation ratio.

Note also that, according to Proposition 3.2, TAKE-ALL gives a much worse competitiveness than n.

3.4.2. TAKE-AT-RANDOM *algorithm*

Recall that TAKE-AT-RANDOM chooses at random one set in F_i per revealed uncovered element σ_i and puts it in the solution. It can be immediately seen that, with the same arguments as those at the end of section 3.3, TAKE-AT-RANDOM achieves competitive ratio $O(n)$. In the following proposition, we show that even its expected competitive ratio, if it chooses one of the sets covering σ_i with uniform probability, cannot be much better than $O(n)$.

THEOREM 3.1.– *For any $\epsilon > 0$, there exists an instance of the online set-covering with n ground elements such that the expected competitive ratio of TAKE-AT-RANDOM is $\Omega(n^{1-\epsilon})$.*

Proof. For any $\epsilon > 0$, fix an integer $k > 1/\epsilon$ and let $N > 2^k$. Consider the instance with ground set $C = \{1, \ldots, n = N^k\}$. Family $\mathcal{S}$ contains the following sets:

– a partition of the ground set into *class* sets $S(i) = \{j \in C : (j-1) \div N = (i-1)\}$; clearly, $|S(i)| = N$ and there exist N^{k-1} class sets;

– for any $j \in S(i)$ for some i, there exist 2^{N-1} *internal* sets, each one containing j plus the elements of one of all possible subsets of $S(i)$ (including the empty set);

– the ground set C itself.

Consider now an arbitrary arrival sequence and calculate the expected value of the cover, which will be equal to the expected competitive ratio of TAKE-AT-RANDOM (equality holds, since the optimum for this instance is C).

Every element belongs to one class set and to 2^{N-1} internal sets. We denote by $\mathbf{E}(q)$ the expected size of the solution provided by TAKE-AT-RANDOM on the instance of q elements defined as before. Then:

$$\begin{aligned}
\mathbf{E}\left(N^k\right) &= \frac{1}{2^{N-1}+1}\left(1+\sum_{l=0}^{N-1}\binom{N-1}{l}\left(1+\mathbf{E}\left(N^k-l-1\right)\right)\right) \\
&\geqslant \frac{1}{2^{N-1}+1}\left(1+2^{N-1}+2^{N-1}\mathbf{E}\left(N^k-N\right)\right) \\
&\approx 1+\mathbf{E}\left(N^k-N\right)
\end{aligned}$$

The recursive relation yields then directly $\mathbf{E}(N^k) \geqslant N^{k-1}$, i.e., $\mathbf{E}(n) = \Omega(n^{1-\epsilon})$. ∎

EXAMPLE 3.1.– We now give an example of construction of Theorem 3.1. Consider $N = 3$ and $k = 3$ (these values of N and k do not conform to their definition but, at first, we use them for simplicity). Then $C = \{1, 2, \ldots, 27\}$ and we have:

– class sets: $\{1,2,3\}$, $\{4,5,6\}$, $\{7,8,9\}$, $\{10,11,12\}$, $\{13,14,15\}$, $\{16,17,18\}$, $\{19,20,21\}$, $\{22,23,24\}$ and $\{25,26,27\}$;

– for any class set $\{a,b,c\}$, there exist the internal sets: $\{a\}$, $\{b\}$, $\{c\}$, $\{a,b\}$, $\{a,c\}$ and $\{b,c\}$;

– finally $C = \{1,2,\ldots,27\} \in \mathcal{S}$.

Let us assume that $\sigma_1 = 17$. With it will be revealed the following $2^2 + 1 = 5$ sets: $\{17\}$, $\{16,17\}$, $\{17,18\}$, $\{17,18,19\}$ and $\{1,2,\ldots,27\}$. The average cover for the whole instance will be of size 9, independently on the arrival sequence.

For $k = 3$, $N \geqslant 8$. Taking $N = 8$, $n = 512$. In this case, the class sets would be the partition of $\{1,2,\ldots,512\}$ into 64 subsequent 8-tuples. For any class set there would be 254 internal sets. With any element of the arrival sequence there would arrive 1 class set, plus 63 internal sets plus the set $\{1,2,\ldots,512\}$, i.e., $8^{3-1}+1 = 65$ sets. The average cover size would be in this case 64.

3.5. The nasty flaw of greediness

In this section, we consider a slightly enriched model for online set-covering by assuming that, together with any element σ_i of the arrival sequence, arrive not only the names of the sets containing them in the final instance but also some information about the covering potential of some of these sets.

In what follows, we show that, even for this revised model, no online algorithm can achieve competitive ratio better than $\sqrt{n/2}$, even if it is allowed to choose at any step more than one set to be introduced in the solution.

THEOREM 3.2.– *Consider an online model for set-covering where, together with any element σ_i of the arrival sequence, arrive not only the names of the sets containing them in the final instance but also some information about the covering potential of some of these sets. Then, no online algorithm for this model such that, at any step, it includes in the cover at least one set containing some not yet covered arriving element can achieve competitive ratio less than $\sqrt{n/2}$, even if we assume that with any σ_i, $\tilde{S}_i$ is also revealed.*

Proof. Consider the following set-covering instance built, for any integer N, upon a ground set $C = \{x_{ij} : 1 \leqslant j \leqslant i \leqslant N\}$; obviously, $|C| = n = N(N+1)/2$. A *path-set of order* i is defined as a set containing $N - i + 1$ elements $\{x_{ij_i}, \ldots, x_{Nj_N}\}$. The set-system $\mathcal{S}$ of the instance contains all possible path-sets of each order i, $1 \leqslant i \leqslant N$. Clearly, there exist $N!/0!$ path-sets of order 1, $N!/1!$ path-sets of order 2, and so on and, finally, $N!/(N-1)!$ path-sets of order N, i.e., in all $N!(1+\ldots+1/(N-1)!) \approx eN!$ path-sets. Finally, the set-system $\mathcal{S}$ is completed with an additional set Y containing all elements of C except those of some path-set of order 1, that will be specified later (hence, $|Y| = n - N$).

As long as there exist uncovered elements, the adversary may choose to have an uncovered element x_{ij} of the lowest possible i arriving, which will be contained only in all path-sets of order less than or equal to i. Notice that as long as algorithm A has $r < N$ sets inserted in the cover, there will be at least one element $x_{(r+1)j}$ for some j, $1 \leqslant j \leqslant k+1$, not yet covered. Suppose that after the arrival of σ_t, the size of the cover calculated by A becomes equal to, or greater than, N. Clearly, $1 \leqslant t \leqslant N$. At time $t+1$, a new element arrives, contained in some path-sets and in Y, which can be now specified as consisting of all elements in C except for the elements of some path-set S^* of order 1 containing $\sigma_1, \ldots, \sigma_t$; the rest of the arrival sequence is indifferent.

Clearly the optimum cover in this case would have been path-set S^* together with set Y; hence, $k_{\mathtt{A}}/k^* \geqslant N/2$, with N tending to $\sqrt{2n}$ as n increases.

It is easy to see that the above construction can be directly generalized so that the same result also holds in the case that the online algorithm is allowed to include more than one set at a time in the cover: if $\sigma_1 = x_{11}$, then as long as the size of the online cover is less than N, there always exists some $i_{\ell-1} < i_\ell \leqslant N$ and some j_{i_ℓ} for which $x_{i_\ell j_{i_\ell}}$ is yet uncovered. Hence, if σ_ℓ is this element, then the algorithm will have to put some sets in the cover. Finally, the algorithm will have put N sets in the cover, while the optimum will always be of size 2. ∎

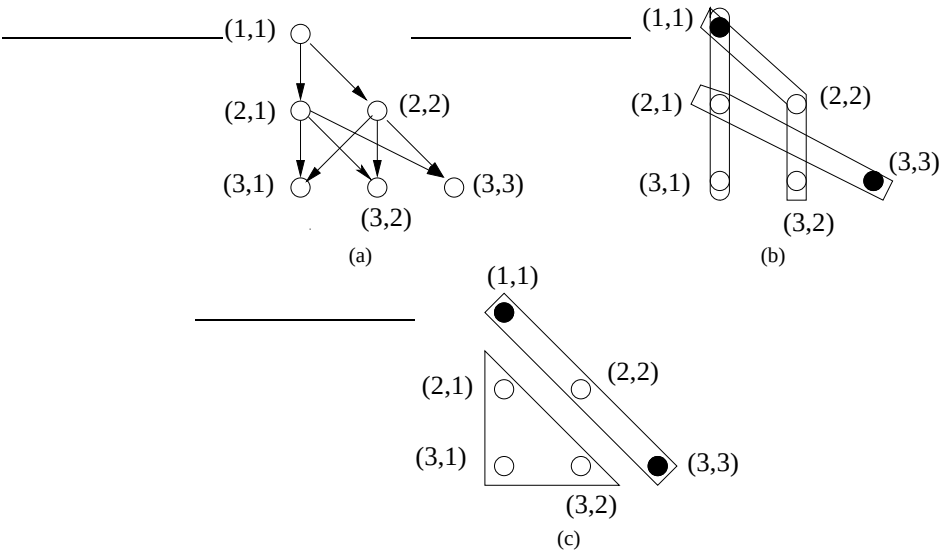


Figure 3.1. *The counter-example of Theorem 3.2 for* $N = 3$

EXAMPLE 3.2.– In order to illustrate the construction in Theorem 3.2, consider the instance of Figure 3.1, with $N = 3$ (the elements of C are depicted as circles labeled by (i, j) for $1 \leqslant j \leqslant i \leqslant 3$). The S_i sets can be thought of as paths terminating to a sink on the directed graph of Figure 3.1(a). Assume that $(1, 1)$ arrives and algorithm A chooses sets $\{(1, 1), (2, 1), (3, 1)\}, \{(1, 1), (2, 2), (3, 2)\}$ for covering it; the uncovered element $(3, 3)$ arrives next, so A has to cover it by, for example, the set $\{(2, 1), (3, 3)\}$ (Figure 3.1(b)). The optimal cover might consist of set $\{(1, 1), (2, 2), (3, 3)\}$ together with a big set consisting of the rest of the elements, that could not have been revealed to A upon arrival of $(1, 1)$, or of $(3, 3)$ (Figure 3.1(c)).

Let us now see what the situation is if we assume that together with F_i is also revealed the name of some set $\hat{S}_i \in \operatorname{argmax}\{|S_i^j|, j = 1, \ldots, f_i\}$ and we implement TAKE-LARGEST that consists of including $\hat{S}_i$ in the solution.

Observe first that the discussion about the competitiveness of deterministic algorithms that include a specific set containing a new (uncovered) element, holds also for TAKE-LARGEST. Hence, its competitive ratio is bounded above by n. We now show that this ratio is asymptotically tight.

Consider the following set-covering instance: a ground set $C = \{1, \ldots, 2N\}$, a family of sets $\mathcal{S} = \{S_0, \ldots, S_N\}$ with $S_i = \{i, \ldots, N + i\}$. Assume an arrival sequence starting with $N, N + 1, \ldots, 2N$. Then, TAKE-LARGEST might include into

the cover sets $S_1, \ldots, S_N$, while the optimum cover would consist of only S_0, S_N, thus yielding a competitive ratio of at least $N/2$.

3.6. The power of look-ahead

The discussion in section 3.5 shows a large gap between the lower bound provided for any algorithm for this model in Theorem 3.2 and the (tight) competitiveness of TAKE-LARGEST, which is one of the most natural rules one could think about. So, an immediate question can be addressed: "what must an online algorithm know in order to achieve a competitive ratio $O(\sqrt{n})$? Is it possible to devise such an algorithm?" The following result tackles these questions.

THEOREM 3.3.– *Consider an instance $(\mathcal{S}, C)$ of minimum set-covering with $|C| = n$. Assume an arrival sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ and suppose that once an element σ_i, $i = 1, \ldots, n$, switches on, the encoding for $\tilde{S}_i \in \operatorname{argmax}\{\delta_i^j, j = 1, \ldots, f_i\}$ is revealed together with σ_i. Consider an implementation of TAKE-LARGEST-ON-FUTURE-ITEMS where, if σ_i is not already covered in one of the previous steps, it includes $\tilde{S}_i$ in the solution. Denote by $\mathcal{S}^* = \{S_1^*, \ldots, S_{k^*}^*\}$ an optimal offline solution on $(\mathcal{S}, C)$. The competitive ratio of TAKE-LARGEST-ON-FUTURE-ITEMS is bounded above by $\min\{\sqrt{2n/k^*}, \sqrt{n}\}$. Furthermore, there exist large enough instances for which this ratio is equal to $\sqrt{n/2}$.*

Proof. Fix an arrival sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ and denote by $c_1, \ldots, c_k$, its *critical elements*, i.e., the elements having entailed introduction of a set in $\mathcal{S}'$. In other words, critical elements of Σ are all elements c_i such that c_i was not yet covered by the cover under construction upon its arrival. Assume also that the final cover $\mathcal{S}'$ consists of k sets, namely, $S_1, \ldots, S_k$, where S_1 has been introduced in $\mathcal{S}'$ due to c_1, S_2 due to c_2 and so on.

Let $\delta(S_i)$ be the increase of the number of covered elements just after having included S_i in the greedy cover (let us recall that if S_i has been added in $\mathcal{S}'$ for the critical element $c_i = \sigma_j$, $\delta(S_i) = \max\{\delta_j^1, \ldots, \delta_j^{f_j}\}$). We have:

$$\delta(S_1) = |S_1| \qquad [3.3]$$

$$\delta(S_i) = \left|\bigcup_{\ell=1}^{i} S_\ell\right| - \left|\bigcup_{\ell=1}^{i-1} S_\ell\right|, \quad 2 \leqslant i \leqslant k \qquad [3.4]$$

Now fix an optimal offline solution $\mathcal{S}^*$ of cardinality k^*. Any of the critical elements $c_1, \ldots, c_k$ can be associated with the smallest-index set in $\mathcal{S}^*$ containing it.

For any $S_i^* \in \mathcal{S}$, we denote by $\hat{S}_i^*$ the set of the critical elements associated with S_i^* (obviously, $\hat{S}_i^* \subseteq S_i^*$). The *critical content* $h(S_i^*)$ of any $S_i^* \in \mathcal{S}^*$ is defined as the number of critical elements associated with it as described before, i.e., $h(S_i^*) = |\hat{S}_i^*|$.

Let $S_1^*, \ldots, S_r^*$ be the sets in $\mathcal{S}^*$ of positive critical contents $h(S_1^*), \ldots, h(S_r^*)$, respectively. Clearly:

$$\sum_{i=1}^{r} h\left(S_i^*\right) = k \qquad [3.5]$$

$$r \leqslant k^* \qquad [3.6]$$

For any S_i^*, let $c_i^1, \ldots, c_i^{h(S_i^*)}$ be the elements of its critical content ordered according to their position in the arrival sequence Σ; in other words, following our assumptions, $\hat{S}_i^* = \{c_i^1, \ldots, c_i^{h(S_i^*)}\}$ (let us recall that $\hat{S}_i^* \subseteq S_i^*$).

Suppose, without loss of generality, that, for $\ell = 1, \ldots, h(S_i^*)$, the set $S_{j_\ell} \in \mathcal{S}$ has been introduced in $\mathcal{S}'$ when the critical element c_i^ℓ has been activated. At the moment of the arrival of c_i^1, the set S_i^* is also a candidate set for $\mathcal{S}'$. The fact that S_{j_1} has been chosen instead of S_i^* means that $\delta(S_{j_1}) \geqslant \delta(S_i^*)$; hence, since, as noticed just above, $\hat{S}_i^* \subseteq S_i^*$, the following holds immediately: $\delta(S_{j_1}) \geqslant \delta(S_i^*) \geqslant |\hat{S}_i^*| = h(S_i^*)$.

When c_i^2 gets activated, the set S_i^* has lost some of its elements that have been covered by some sets already chosen by the algorithm. In any case, it has lost c_i^1 (covered by S_{j_1}). So, following the arguments developed just above for S_{j_1}, $\delta(S_{j_2}) \geqslant h(S_i^*) - 1$ and so on (quantities $\delta(\cdot)$ are defined either by [3.3], or by [3.4]). So, when dealing with c_i^ℓ, the following holds:

$$h\left(S_i^*\right) - \ell + 1 \leqslant \delta\left(S_{j_\ell}\right) \qquad [3.7]$$

For example, consider the illustration of Figure 3.2. Let S^* be a set of the fixed optimal cover $\mathcal{S}^*$ and denote by $\hat{S}$ the set of its critical elements, c^1, c^2 and c^3 (ranged in the order they have been activated). Let S be the set chosen to cover c^2. The shadowed parts of S^*, $\hat{S}$ and S correspond to elements already covered at the moment of arrival of c^2. At this moment, S must contain at least as many uncovered elements as S^* does and *a fortiori* at least one uncovered element for any yet uncovered critical element of S^* (two uncovered elements for S appear below the dashed line for c^3 and c^4).

Summing up inequalities [3.7], for $\ell = 1, \ldots, h(S_i^*)$, and setting:

$$\sum_{\ell=1}^{h(S_i^*)} \delta\left(S_{j_\ell}\right) = n_i$$

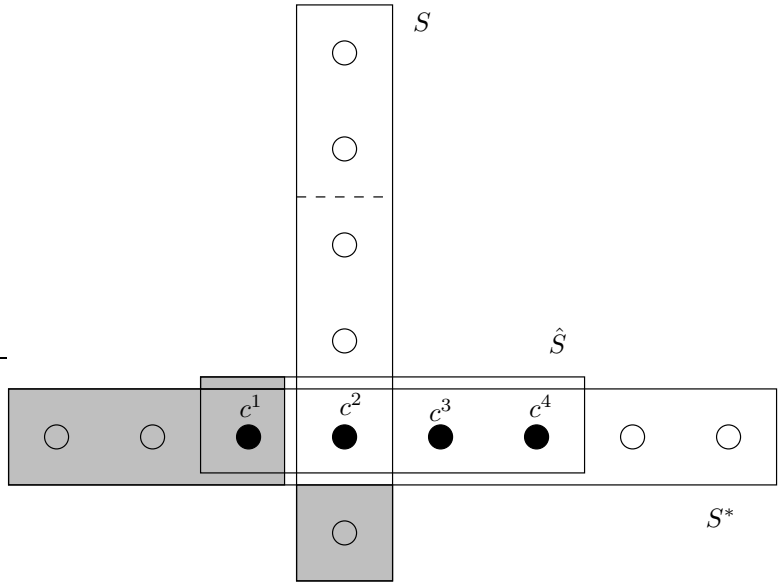


Figure 3.2. *An example for [3.7]*

we finally get for S_i:

$$\frac{h\left(S_i^*\right)\left(h\left(S_i^*\right)+1\right)}{2} \leqslant \sum_{\ell=1}^{h(S_i^*)} \delta\left(S_{j_\ell}\right) = n_i \Longrightarrow h\left(S_i^*\right) \leqslant \sqrt{2n_i} \qquad [3.8]$$

Set, for $1 \leqslant i \leqslant r$, $n_i = \alpha_i n$, for some $\alpha_i \in [0,1]$. Then, $\sum_{i=1}^{r} \alpha_i = 1$ and:

$$\sum_{i=1}^{r} \sqrt{\alpha_i} \leqslant \sqrt{r} \qquad [3.9]$$

Using [3.5], [3.6], [3.8] and [3.9], we get:

$$k = \sum_{i=1}^{r} h\left(S_i^*\right) \leqslant \sqrt{2n} \sum_{i=1}^{r} \sqrt{\alpha_i} \leqslant \sqrt{r}\sqrt{2n} \leqslant \sqrt{k^*}\sqrt{2n} \qquad [3.10]$$

By dividing the first and the last members of [3.10] by k^*, we get:

$$\frac{k}{k^*} \leqslant \sqrt{\frac{2n}{k^*}} \qquad [3.11]$$

On the other hand, note that, if $k^* = 1$, i.e., if there exists $S^* \in \mathcal{S}$ such that $\mathcal{S}^* = \{S^*\}$, then TAKE-LARGEST-ON-FUTURE-ITEMS would have chosen it from the beginning of its running in order to cover σ_1; next, no additional set would have entered the $\mathcal{S}'$. Consequently, we can assume that $k^* \geqslant 2$ and, using [3.11]:

$$\frac{k}{k^*} \leqslant \sqrt{n} \qquad [3.12]$$

Combination of [3.11] and [3.12] concludes the competitive ratio claimed by the statement of the theorem.

Fix an integer N and consider the following instance $(\mathcal{S}, C)$ of minimum set-covering:

$$\begin{aligned}
C &= \left\{1, \ldots, \frac{N(N+1)}{2}\right\} \\
S_1 &= \{1, \ldots, N\} \\
S_2 &= \{N+1, \ldots, 2N-1\} \\
&\vdots \\
S_N &= \left\{\frac{N(N+1)}{2}\right\} \\
S_{N+1} &= \left\{(i-1)N - \frac{i(i-3)}{2} : i = 1, \ldots, N\right\} \\
S_{N+2} &= C \setminus S_{N+1}
\end{aligned}$$

Consider the arrival sequence $(1, \ldots, N(N+1)/2)$. The algorithm might calculate the cover $\mathcal{S}' = \{S_i, 1 \leqslant i \leqslant N\}$, while the optimal one is $\mathcal{S}^* = \{S_{N+1}, S_{N+2}\}$. Hence, the competitive ratio in this case would be $N/2$, with $N = (-1 + \sqrt{1+8n})/2$, which is asymptotically equal to $\sqrt{n/2}$ as claimed.

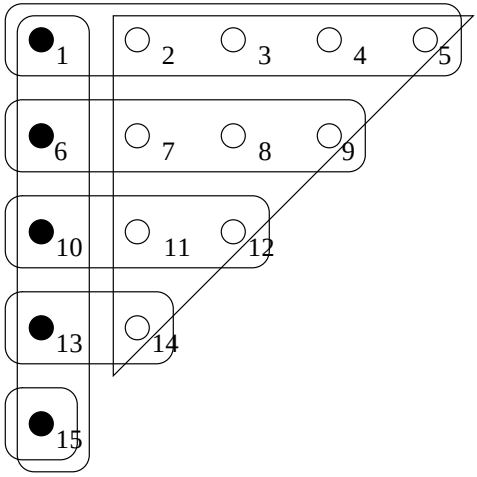


Figure 3.3. *The ratio $\sqrt{n/2}$ for* `TAKE-LARGEST-ON-FUTURE-ITEMS` *is asymptotically attained*

For example, consider Figure 3.3. If Σ starts with 1, 6, 10, 13, 15, TAKE-LARGEST-ON-FUTURE-ITEMS may have chosen $\{1,2,3,4,5\}$, $\{6,7,8,9\}$, $\{10,11,12\}$ and $\{13,14\}, \{15\}$, respectively, while the optimal cover would consist of two sets, namely, $\{1,6,10,13,15\}$ and $\{2,3,4,5,7,8,9,11,12,14\}$. The proof of the theorem is now complete. ∎

Revisit [3.11], set $\Delta = \max_{S_i \in \mathcal{S}}\{|S_i|\}$ and take into account the obvious inequality: $k^* \geqslant n/\Delta$. Then, the following result is immediately derived from Theorem 3.3.

COROLLARY 3.1.– The competitive ratio of TAKE-LARGEST-ON-FUTURE-ITEMS is bounded above by $\sqrt{2\Delta}$.

It can be easily seen from the proof of Theorem 3.3 that it also works even if we assume that the arrival sequence does not contain all the elements of C but only a part of them. In this case, the competitive ratio achieved is $\sqrt{2n/k^*} \leqslant \sqrt{2n}$, since the hypothesis on the size of k^* (discussed at the end of the proof of Theorem 3.3) is no longer valid. This is important for TAKE-LARGEST-ON-FUTURE-ITEMS because it can be seen as an algorithm working also for the online model in [ALO 03] with non-trivial upper competitive bound.

COROLLARY 3.2.– Assuming that only a subset of C will finally be revealed, the competitive ratio of TAKE-LARGEST-ON-FUTURE-ITEMS is bounded above by $\sqrt{2n}$.

The counter-example instance given in the proof of Theorem 3.3 can be slightly modified to fit the case where, at each step, whenever a yet uncovered element arrives, the algorithm is allowed to include in the cover a constant number of sets containing it and such that the number of elements yet switched off that belong to these sets is maximized.

Consider a modification of TAKE-LARGEST-ON-FUTURE-ITEMS where, for a fixed number ρ, when a new ground element σ_i arrives, the ρ sets in F_i covering most of the still uncovered elements are included in the solution. Then, the following holds.

PROPOSITION 3.3.– *The modified version of TAKE-LARGEST-ON-FUTURE-ITEMS has competitive ratio bounded below by* $\sqrt{\rho n}/2$.

Proof. For some $\rho > 1$ and for some integer N, consider the following instance:

$$\begin{aligned}
\mathcal{S} &= \left\{X, Y, S_i^j : 1 \leqslant i \leqslant N, 1 \leqslant j \leqslant \rho\right\} \\
C &= \bigcup_{i=1}^{N}\bigcup_{j=1}^{\rho} S_i^j \quad \left(|C| = \rho\frac{N(N-1)}{2} + N = n\right) \\
X &= \{x_1, \ldots, x_N\} \\
\left|S_i^j\right| &= N - i + 1 \text{ for } i = 1, \ldots, N \\
S_i^j \bigcap S_l^k &= \emptyset, \text{ if } i \neq l \\
S_i^j \bigcap S_i^k &= \{x_i\}, \text{ if } j \neq k \\
Y &= C \setminus X
\end{aligned}$$

Consider the arrival sequence where $x_1, \ldots, x_N$ are firstly revealed. TAKE-LARGEST-ON-FUTURE-ITEMS might include in the cover all the S_i^j, while the optimal cover is $\{X, Y\}$. In this case, the competitive ratio is $\rho N/2$, with:

$$N = \frac{\rho - 2}{2\rho} + \sqrt{\left(\frac{\rho - 2}{2\rho}\right)^2 + 2\frac{n}{\rho}}$$

i.e., the value of the ratio is asymptotically $\sqrt{\rho n/2}$.

For example, set $\rho = 2$ and $N = 5$ and consider the instance of Figure 3.4. For Σ starting with x_1, x_2, x_3, x_4, x_5, the algorithm may insert into the cover the

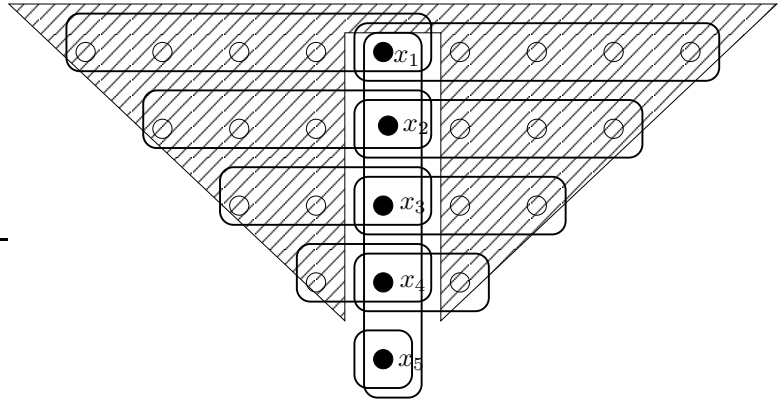


Figure 3.4. *A counter-example for the case where the algorithm is allowed to include a constant number of sets containing a recently arrived element*

sets depicted as "rows", while the optimal cover would consist of the "column"-set $\{x_1, x_2, x_3, x_4, x_5\}$ together with the "big" set containing the rest of the elements (drawn with diagonal lines in Figure 3.4). ∎

In the weighted version of set-covering, any set S of $\mathcal{S}$ is assigned with a non-negative weight $w(S)$, and a cover $\mathcal{S}'$ of the least possible total-weight $\sum_{S \in \mathcal{S}'} w(S)$ has to be calculated. A natural modification of `TAKE-LARGEST-ON-FUTURE-ITEMS` in order to deal with weighted set-covering is to put in the cover, whenever a still uncovered element arrives, a set S_i containing it that minimizes the quantity $w(S_i)/\delta(S_i)$. Unfortunately, this modification cannot perform satisfactorily. Consider, for example, an instance of weighted set-covering consisting of a ground set $C = \{x_1, \ldots, x_n\}$ and three sets $S = C$ with $w(S) = n$, $X = \{x_1\}$ with $w(X) = 1$ and $Y = C \setminus \{x_1\}$ with $w(Y) = 0$. If x_1 arrives first, the algorithm could have chosen S to cover it, thus yielding a cover for the overall instance of total weight n, while the optimal cover would be $\{X, Y\}$ of total weight 1.

3.7. The maximum budget saving problem

In this section, we study a kind of dual version of the minimum set-covering: the maximum budget saving problem. Here, we are allotted an initial budget $B(\mathcal{S}, C)$ destined to cover the cost of an algorithm that solves minimum set-covering on $(\mathcal{S}, C)$. Any such algorithm has its own cost that is a function of the size of the solution produced, of the time overheads it takes in order to calculate it, etc. Our objective is to maximize our savings, i.e., the difference between the initial budget and the cost of the algorithm. For simplicity, we assume that the maximum saving ever possible

to be performed is $B(\mathcal{S}, C) - k^*$, where, as previously, k^* is the size of an optimum set-cover of $(\mathcal{S}, C)$.

We consider here that the set-covering instance arrives online. If a purely online algorithm is used to solve it, then its cost equals the size of the solution calculated; otherwise, if the algorithm allows itself to wait in order to solve the instance (partly or totally) offline, then its cost is the sum of the size of the solution calculated plus a fine that is equal to some root, of order strictly smaller than 1, of the solution that would be calculated by a purely online algorithm. We suppose that the budget allotted is equal to $k^*\sqrt{n}$, where $n = |C|$. This assumption on $B(\mathcal{S}, C)$ is quite natural. It corresponds to a kind of feasible cost for an algorithm; we assume that this algorithm is `TAKE-LARGEST-ON-FUTURE-ITEMS`, which is the best among the ones seen in the chapter.

The interpretation of this model is the following. We are allotted a budget corresponding to the cost of an algorithm always solving set-covering. In this way, we are sure that we can always construct a feasible solution for it. Furthermore, by the second part of Theorem 3.3, it is very risky to be allotted less than $k^*\sqrt{n}$ since there are instances where the bound $\sqrt{n}$ is attained. On the other hand, we can have at our disposal a bunch of online or offline set-covering algorithms, any one having its proper cost as described just above, from which we have to choose the one whose use will allow us to perform the maximum possible economy with respect to our initial budget. The fact that the measure of the optimum solution for maximum budget saving is $B(\mathcal{S}, C) - k^*$, has also a natural interpretation: we can assume that there exists an arrival sequence Σ for C such that, for any $\sigma_i \in \Sigma$, an oracle can always choose to cover σ_i with the same set with which σ_i is covered in an optimum offline solution, for instance $(\mathcal{S}, C)$. Under this assumption for the measure of the optimum budget saving solution, this problem is clearly **NP**-hard since it implies the calculation of an optimum solution for minimum set-covering. Finally, denoting by $c_{\mathtt{A}}(\mathcal{S}, C)$ the cost of algorithm `A` when solving minimum set-covering on $(\mathcal{S}, C)$, the approximation ratio of maximum set saving is equal to:

$$\frac{B(\mathcal{S}, C) - c_{\mathtt{A}}(\mathcal{S}, C)}{B(\mathcal{S}, C) - k^*} \qquad [3.13]$$

Obviously this ratio is smaller than 1 and, furthermore, the closer the ratio is to 1, the better the algorithm achieving it.

THEOREM 3.4.– *Under the model adopted, `GREEDY` is asymptotically optimum for maximum budget saving.*

Proof. Consider an instance $(\mathcal{S}, C)$ of minimum set-covering problem and denote by k_{L} and k_{F} the sizes of the solutions calculated by `TAKE-LARGEST-ON-FUTURE-ITEMS` and `GREEDY`, respectively. According to what has been assumed just above, when denoting by c_{F} the cost of using `GREEDY`, there exists some $\epsilon > 0$ such that:

$$c_{\text{F}}(\mathcal{S}, C) = k_{\text{F}} + k_{\text{L}}^{1-\epsilon} \quad [3.14]$$

Moreover, the following inequalities hold, the first one from [SLA 96] and the second one from Theorem 3.3:

$$k_{\text{F}} \leqslant k^* \log n \quad [3.15]$$

$$k_{\text{L}} \leqslant k^* \sqrt{n} \quad [3.16]$$

Using [3.14], [3.15] and [3.16], we get the following inequality for $c_{\text{F}}(\mathcal{S}, C)$:

$$c_{\text{F}}(\mathcal{S}, C) \leqslant {k^*}^{1-\epsilon} n^{\frac{1-\epsilon}{2}} + k^* \log n \leqslant \left(n^{\frac{1-\epsilon}{2}} + \log n \right) k^* \quad [3.17]$$

On the other hand, as assumed above:

$$B(\mathcal{S}, C) = k^* \sqrt{n} \quad [3.18]$$

Using [3.13], [3.17] and [3.18], we obtain:

$$\begin{aligned} \frac{B(\mathcal{S}, C) - c_{\text{F}}(\mathcal{S}, C)}{B(\mathcal{S}, C) - k^*} &\geqslant \frac{k^* \sqrt{n} - \left(n^{\frac{1-\epsilon}{2}} + \log n \right) k^*}{k^* \sqrt{n} - k^*} \\ &= \frac{\sqrt{n} - \left(n^{\frac{1-\epsilon}{2}} + \log n \right)}{\sqrt{n} - 1} \end{aligned} \quad [3.19]$$

It is easy to see that, for n large enough, the last term of [3.19] tends to 1 and the statement claimed by the theorem is true. ∎

3.8. Discussion

We have introduced several simple online models for set-covering and analyzed greedy rules for them. Many of these rules are strongly competitive since no online algorithm for the models that they treat can achieve better ratios than they do. One of the features of the models studied here is that they are very economic and thus suitable to solve very large instances. Indeed, their memory requirements are extremely reduced since the only information needed are the names of m sets. Note that this is not the case for the intensive calculations implied by the very interesting model of [ALO 03].

Next, we have introduced and studied the maximum budget saving problem. Here, we have relaxed irrevocability in the solution construction by allowing the algorithm to delay its decisions modulo some fine to be paid. For such a model we have shown that the natural greedy offline algorithm is asymptotically optimum.

Finally, let us note that the online models described in the chapter can be extended to apply to a different but related problem: the minimum dominating set[1]. Consistently with the model that we have adopted for the set-covering problem, our model for this latter problem is as follows. Given a graph $G(V, E)$ with $|V| = n$, assume that its vertices switch on one-by-one. Any time a vertex σ_i switches on, the names of its neighbors are announced.

Consider the following traditional reduction from minimum dominating set to set-covering:

– $\mathcal{S} = C = V$;

– the set $S_i \in \mathcal{S}$, corresponding to the vertex $v_i \in V$, contains elements $c_{i_1}, c_{i_2}, \ldots$, of C corresponding to the neighbors $v_{i_1}, v_{i_2}, \ldots$, of v_i in G.

The set-covering instance $(\mathcal{S}, C)$ so constructed, has $|\mathcal{S}| = |C| = n$. Furthermore, it is easy to see that any set cover of size k in $(\mathcal{S}, C)$ corresponds to a dominating set of the same size in G and vice versa. Also note that the dominating set model just assumed on G is exactly, with respect to $(\mathcal{S}, C)$, the set-covering model which is tackled in the chapter.

3.9. Bibliography

[ALO 03] ALON N., AWERBUCH B., AZAR Y., BUCHVINDER N., NAOR S., "The online set cover problem", *Proc. STOC'03*, p. 100–105, 2003.

1. Given a graph $G(V, E)$ the minimum dominating set problem consists of determining a minimum-size subset $V' \subseteq V$ such that, for all $u \in V \setminus V'$, there exists $v \in V'$ for which $(u, v) \in E$.

[CHV 79] CHVÁTAL V., "A greedy-heuristic for the set covering problem", *Math. Oper. Res.*, vol. 4, p. 233–235, 1979.

[GAM 97] GAMBOSI G., PROTASI M., TALAMO M., "Preserving approximation in the min-weighted set cover problem", *Discrete Appl. Math.*, vol. 73, p. 13–22, 1997.

[HOC 82] HOCHBAUM D. S., "Approximation algorithms for the set covering and vertex cover problems", *SIAM J. Comput.*, vol. 11, num. 3, p. 555–556, 1982.

[JOH 74] JOHNSON D. S., "Approximation algorithms for combinatorial problems", *J. Comput. System Sci.*, vol. 9, p. 256–278, 1974.

[KAR 72] KARP R. M., "Reducibility among combinatorial problems", MILLER R. E., THATCHER J. W., Eds., *Complexity of computer computations*, p. 85–103, Plenum Press, New York, 1972.

[LOV 75] LOVÁSZ L., "On the ratio of optimal integral and fractional covers", *Discrete Math.*, vol. 13, p. 383–390, 1975.

[PAS 97] PASCHOS V. T., "A survey about how optimal solutions to some covering and packing problems can be approximated", *ACM Comput. Surveys*, vol. 29, num. 2, p. 171–209, 1997.

[RAZ 97] RAZ R., SAFRA S., "A sub-constant error probability low-degree test and a sub-constant error probability PCP characterization of NP", *Proc. STOC'97*, p. 475–484, 1997.

[SLA 96] SLAVÍK P., "A tight analysis of the greedy algorithm for set cover", *Proc. STOC'96*, p. 435–441, 1996.

[SLE 85] SLEATOR D., TARJAN R. E., "Amortized efficiency of list update and paging rules", *Commun. ACM*, vol. 28, num. 2, p. 202–208, 1985.

[TEL 04] TELELIS O. A., ZISSIMOPOULOS V., Dynamic maintenance of approximate set covers, Manuscript, 2004.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 4

Comparison of Expressiveness for Timed Automata and Time Petri Nets

4.1. Introduction

Petri nets with time. The two main extensions of Petri Nets with time are Time Petri Nets (TPNs) [MER 74] and Timed Petri Nets [RAM 74]. For TPNs a transition can fire within a time interval, whereas for Timed Petri Nets it fires as soon as possible. Among Timed Petri Nets, time can be considered relative to places or transitions [SIF 80, PEZ 99]. The two corresponding subclasses, i.e., P-Timed Petri Nets and T-Timed Petri Nets, are expressively equivalent [SIF 80, PEZ 99]. The same classes are defined for TPNs, i.e., T-TPNs and P-TPNs, but both classes of Timed Petri Nets are included in both P-TPNs and T-TPNs [PEZ 99]. P-TPNs and T-TPNs are proved to be incomparable in [KHA 96]. Finally TPNs form a subclass of Time Stream Petri Nets [DIA 94] which were introduced to model multimedia applications. Timed Arc Petri Nets are also studied in more recent works [ABD 01, FRU 00].

Timed automata. Timed automata (TA) were introduced by Alur & Dill [ALU 94] and have since been extensively studied. This model is an extension of finite automata with (dense time) *clocks* and makes it possible to specify real-time systems. Theoretical properties of various classes of TA have been considered in the last decade. For instance, a class of determinizable TA such as *Event Clock Automata* are investigated

Chapter written by Béatrice BÉRARD, Franck CASSEZ, Serge HADDAD, Didier LIME and Olivier-Henri ROUX.

in [ALU 99] and form a strict subclass of TA. More general models of TA like *Rectangular Automata* or *Linear Hybrid Automata* have also been considered and their expressive power compared.

However, not much is known about the expressive power of TPN compared to TA.

Related work. In a previous work [CAS 04] we have proved that TPN forms a subclass of TA in the sense that every TPN can be simulated by a TA (weak timed bisimilarity). A similar result can be found in [LIM 03] with a completely different approach.

In another line of work [HAA 02], Haar, Kaiser, Simonot & Toussaint compare Timed State Machines and Time Petri Nets. They give a translation from one model to another that preserves timed languages. Nevertheless, in the translation from TSM to TPN they use a *weak* semantics for TPN and consider only the constraints with bounded and closed intervals.

Our contribution. In this article, we compare precisely the expressive power of TA vs. TPN using the notions of *Timed Language Acceptance* and *Timed Bisimilarity*. This extends previous results in this area in the following directions: *i)* we consider general types of constraints (strict, large, bounded, unbounded); *ii)* we then show that there is a TA $\mathcal{A}$ such that no TPN is (even weakly) timed bisimilar to $\mathcal{A}$; *iii)* this leads us to consider weaker notions of equivalence and we focus on timed language acceptance. We prove that TA (with general types of constraints) and TPN are equally expressive with respect to timed language acceptance; *iv)* to conclude we characterize the subclass of TA that is equally expressive to TPN without strict constraints with respect to timed bisimilarity, and show that the membership problem for this class is **PSPACE**-complete as well as the reachability problem. The results of the chapter are summarized in Table 4.1: all the results are new except the one on the first line obtained in [CAS 04]. We use the following notations: B-$\mathcal{TPN}_\varepsilon$ for the set of bounded TPNs with ε-transitions; 1-B-$\mathcal{TPN}_\varepsilon$ for the subset of B-$\mathcal{TPN}_\varepsilon$ with at most one token in each place (one safe TPN); B-$\mathcal{TPN}(\leqslant, \geqslant)$ for the subset of B-$\mathcal{TPN}_\varepsilon$ where only closed intervals are used; $\mathcal{TA}_\varepsilon$ for TA with ε-transitions; $\mathcal{TA}^-$ for the class of TA (to be defined precisely in section 4.6) that is equivalent to B-$\mathcal{TPN}(\leqslant, \geqslant)$.

Outline on the chapter. Section 4.2 introduces the semantics of TPNs and TA, timed languages and timed bisimilarity and section 4.3 is devoted to the comparison between various semantics for TPNs. In section 4.4, we prove negative results: we exhibit some timed automata for which there exist no (weakly) timed bisimilar TPN. In section 4.5 we focus on timed language acceptance and prove that TA and TPNs are equally expressive with respect to this equivalence. Section 4.6 is devoted to a characterization of the subclass of TA that is equivalent to TPN with respect to timed bisimilarity. Finally we give some hints on further work in section 4.7.

	timed language acceptance	**timed bisimilarity**
B-$\mathcal{TPN}_\varepsilon$	$\leqslant_{\mathcal{L}} \mathcal{TA}_\varepsilon$ ([CAS 04])	$\leqslant_{\mathcal{W}} \mathcal{TA}_\varepsilon$ ([CAS 04])
	$=_{\mathcal{L}}$ 1-B-$\mathcal{TPN}_\varepsilon$ $=_{\mathcal{L}} \mathcal{TA}_\varepsilon$	$<_{\mathcal{W}} \mathcal{TA}_\varepsilon$
B-$\mathcal{TPN}(\leqslant,\geqslant)$	—	$\approx_{\mathcal{W}} \mathcal{TA}^-$

	Emptiness Problem	**Universal Problem**
B-$\mathcal{TPN}_\varepsilon$	Decidable	Undecidable

	Membership Problem	**Reachability Problem**
$\mathcal{TA}^-$	**PSPACE**-complete	

Table 4.1. *Summary of the results*

4.2. Time Petri nets and timed automata

Notations. Let Σ be a set (or alphabet). Σ^* (resp. Σ^ω) denotes the set of finite (resp. infinite) sequences of elements (or words) of Σ and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. By convention if $w \in \Sigma^\omega$ then the *length* of w denoted by $|w|$ is ω; otherwise, if $w = a_1 \cdots a_n$, $|w| = n$. We also use $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ with $\varepsilon \notin \Sigma$, where ε is the empty word. B^A stands for the set of mappings from A to B. If A is finite and $|A| = n$, an element of B^A is also a vector in B^n. The usual operators $+, -, <$ and $=$ are used on vectors of A^n with $A = \mathbb{N}, \mathbb{Q}, \mathbb{R}$ and are the point-wise extensions of their counterparts in A. The set $\mathbb{B}$ denotes the Boolean values $\{\mathsf{tt}, \mathsf{ff}\}$ and $\mathbb{R}_{\geqslant 0}$ denotes the set of positive reals. A *valuation* ν over a set of variables X is an element of $\mathbb{R}_{\geqslant 0}^X$. For $\nu \in \mathbb{R}_{\geqslant 0}^X$ and $d \in \mathbb{R}_{\geqslant 0}$, $\nu + d$ denotes the valuation defined by $(\nu + d)(x) = \nu(x) + d$, and for $X' \subseteq X$, $\nu[X' \mapsto 0]$ denotes the valuation ν' with $\nu'(x) = 0$ for $x \in X'$ and $\nu'(x) = \nu(x)$ otherwise. $\mathbf{0}$ denotes the valuation such that $\forall x \in X, \nu(x) = 0$. An *atomic constraint* is a formula of the form $x \bowtie c$ for $x \in X$, $c \in \mathbb{Q}_{\geqslant 0}$ and $\bowtie \in \{<, \leqslant, \geqslant, >\}$. We denote by $\mathcal{C}(X)$ the set of *constraints* over a set of variables X which consists of the conjunctions of atomic constraints. Given a constraint $\varphi \in \mathcal{C}(X)$ and a valuation $\nu \in \mathbb{R}_{\geqslant 0}^X$, we denote by $\varphi(\nu) \in \mathbb{B}$ the truth value obtained by substituting each occurrence of x in φ by $\nu(x)$. Accordingly, each constraint $\varphi \in \mathcal{C}(X)$ defines a set of valuations $[\![\varphi]\!]$ defined by $[\![\varphi]\!] = \{\nu \in \mathbb{R}_{\geqslant 0}^X \mid \varphi(\nu) = \mathsf{tt}\}$.

A set I is a $\mathbb{Q}_{\geqslant 0}$-*interval* of $\mathbb{R}_{\geqslant 0}$ if there is a constraint φ of the form $a \prec_1 x \prec_2 b$ with $a \in \mathbb{Q}_{\geqslant 0}$, $b \in \mathbb{Q}_{\geqslant 0} \cup \{\infty\}$ and $\prec_1, \prec_2 \in \{<, \leqslant\}$, such that $I = [\![\varphi]\!]$. We let $I^\downarrow = [\![0 \leqslant x \prec_2 b]\!]$ be the *downward closure* of I and $I^\uparrow = [\![a \prec_1 x]\!]$ be the *upward closure* of I. We denote by $\mathcal{I}(\mathbb{Q}_{\geqslant 0})$ the set of $\mathbb{Q}_{\geqslant 0}$-intervals of $\mathbb{R}_{\geqslant 0}$. Let $g \in \mathbb{N}_{>0}$, we write $\mathbb{N}_g = \{\frac{i}{g} \mid i \in \mathbb{N}\}$. A vector $\mathbf{v} \in \mathbb{Q}^n$ belongs to the g-*grid* if $\mathbf{v}(k) \in \mathbb{N}_g$ for all $1 \leqslant k \leqslant n$.

4.2.1. *Timed transition systems and equivalence relations*

Let Σ be a fixed finite alphabet such that $\varepsilon \notin \Sigma$.

DEFINITION 4.1.– *[Timed Words] A timed word w over Σ_ε is a finite or infinite sequence $w = (a_0, d_0)(a_1, d_1)\cdots(a_n, d_n)\cdots$ such that for each $i \geqslant 0$, $a_i \in \Sigma_\varepsilon$, $d_i \in \mathbb{R}_{\geqslant 0}$ and $d_{i+1} \geqslant d_i$.*

A timed word w over Σ_ε can be viewed as a pair $(v, \tau) \in \Sigma_\varepsilon^\infty \times \mathbb{R}_{\geqslant 0}^\infty$ such that $|v| = |\tau|$. The value d_k gives the absolute time (from the initial instant 0) of action a_k. We write *Untimed*$(w) = v$ for the untimed part of w, and *Duration*$(w) = \sup_{d_k \in \tau} d_k$ for the duration of the timed word w. For a timed word $(a_i, d_i)_{i \geqslant 0}$ we define the *relative time stamp Rstamp*(a_i) of a_i as *Rstamp*$(a_i) = d_i - d_{i-1}$ with the convention that $d_{-1} = 0$ and extend this notion to timed words by defining *Rstamp*$(w) = (a_i, \textit{Rstamp}(a_i))_{i \geqslant 0}$. Note that, conversely, from such a sequence, we can retrieve a timed word with *absolute time stamps* by cumulating the successive delays.

Since ε-transitions correspond to the empty word and are not visible, we can remove from each timed word $w \in \Sigma_\varepsilon^\infty \times \mathbb{R}_{\geqslant 0}^\infty$ all the ε-actions and obtain a timed word in $\Sigma^\infty \times \mathbb{R}_{\geqslant 0}^\infty$.

DEFINITION 4.2.– *[Timed Languages] We denote by $\mathcal{TW}^*(\Sigma)$ (resp. $\mathcal{TW}^\omega(\Sigma)$) the set of finite (resp. infinite) timed words over Σ and $\mathcal{TW}^\infty(\Sigma) = \mathcal{TW}^*(\Sigma) \cup \mathcal{TW}^\omega(\Sigma)$. A timed language L over Σ is any subset of $\mathcal{TW}^\infty(\Sigma)$.*

Timed transition systems describe systems which combine discrete and continuous evolutions.

DEFINITION 4.3.– *[Timed Transition Systems] A timed transition system (TTS) over the set of actions Σ_ε is a tuple $S = (Q, Q_0, \Sigma_\varepsilon, \longrightarrow, F, R)$ where Q is a set of states, $Q_0 \subseteq Q$ is the set of initial states, Σ_ε is a finite set of actions disjoint from $\mathbb{R}_{\geqslant 0}$ and $\longrightarrow \subseteq Q \times (\Sigma_\varepsilon \cup \mathbb{R}_{\geqslant 0}) \times Q$ is a set of edges. If $(q, e, q') \in \longrightarrow$, we also write $q \xrightarrow{e} q'$. For a transition $q \xrightarrow{d} q'$ with $d \in \mathbb{R}_{\geqslant 0}$, the value d represents a relative time stamp. The sets $F \subseteq Q$ and $R \subseteq Q$ are respectively the sets of final and repeated states.*

We make the following common assumptions about TTS:

– 0-DELAY: $q \xrightarrow{0} q'$ if and only if $q = q'$;

– ADDITIVITY: if $q \xrightarrow{d} q'$ and $q' \xrightarrow{d'} q''$ with $d,\ d' \in \mathbb{R}_{\geqslant 0}$, then $q \xrightarrow{d+d'} q''$;

– CONTINUITY: if $q \xrightarrow{d} q'$, then for every d' and d'' in $\mathbb{R}_{\geqslant 0}$ such that $d = d' + d''$, there exists q'' such that $q \xrightarrow{d'} q'' \xrightarrow{d''} q'$;

– TIME-DETERMINISM: if $q \xrightarrow{d} q'$ and $q \xrightarrow{d} q''$ with $d \in \mathbb{R}_{\geqslant 0}$, then $q' = q''$.

A *run* ρ of length $n \geqslant 0$ is a finite or infinite ($n = \omega$) sequence of transitions of the form:

$$\rho = q_0 \overset{d_0}{\rightarrow} q'_0 \overset{a_0}{\rightarrow} q_1 \overset{d_1}{\rightarrow} q'_1 \overset{a_1}{\rightarrow} \cdots q_n \overset{d_n}{\rightarrow} q'_n \ldots$$

where discrete actions alternate with durations. We write $first(\rho) = q_0$ and if ρ is finite, we assume that it ends with an action transition and we set $last(\rho) = q_n$. We write $q \overset{*}{\rightarrow} q'$ if there is a run ρ such that $first(\rho) = q$, $last(\rho) = q'$.

A run is *initial* if $first(\rho) \in Q_0$. A run ρ is *accepting* if: i) either ρ is a finite initial run and $last(\rho) \in F$ or ii) ρ is infinite and there is a state $q \in R$ that appears infinitely often on ρ. From the sequence $(a_0, d_0)(a_1, d_1) \ldots$ associated with ρ, we obtain a timed word w by considering the absolute time stamps of actions : $w = (a_0, d_0)(a_1, d_0 + d_1) \ldots$ This word is *accepted* by S if ρ is an accepting run.

The *timed language* $\mathcal{L}(S)$ accepted by S is the set of timed words accepted by S.

DEFINITION 4.4.– *[Strong Timed Similarity] Let $S_1 = (Q_1, Q_0^1, \Sigma_\varepsilon, \longrightarrow_1, F_1, R_1)$ and $S_2 = (Q_2, Q_0^2, \Sigma_\varepsilon, \longrightarrow_2, F_2, R_2)$ be two TTS and $\preceq$ be a binary relation over $Q_1 \times Q_2$. We write $s \preceq s'$ for $(s, s') \in \preceq$. The relation $\preceq$ is a strong (timed) simulation relation of S_1 by S_2:*

1) if $s_1 \in F_1$ (resp. $s_1 \in R_1$) and $s_1 \preceq s_2$, then $s_2 \in F_2$ (resp. $s_2 \in R_2$);

2) if $s_1 \in Q_0^1$ there is some $s_2 \in Q_0^2$ such that $s_1 \preceq s_2$;

3) if $s_1 \overset{d}{\rightarrow}_1 s'_1$ with $d \in \mathbb{R}_{\geqslant 0}$ and $s_1 \preceq s_2$, then $s_2 \overset{d}{\rightarrow}_2 s'_2$ for some s'_2, and $s'_1 \preceq s'_2$;

4) if $s_1 \overset{a}{\rightarrow}_1 s'_1$ with $a \in \Sigma_\varepsilon$ and $s_1 \preceq s_2$, then $s_2 \overset{a}{\rightarrow}_2 s'_2$ and $s'_1 \preceq s'_2$.

A TTS S_2 strongly simulates S_1 if there is a strong (timed) simulation relation of S_1 by S_2. We write $S_1 \preceq_{\mathcal{S}} S_2$ in this case.

When there is a strong simulation relation $\preceq$ of S_1 by S_2 and $\preceq^{-1}$ is also a strong simulation relation[1] of S_2 by S_1, we say that $\preceq$ is a *strong (timed) bisimultion relation* between S_1 and S_2 and use $\approx$ instead of $\preceq$. Two TTS S_1 and S_2 are *strongly (timed) bisimilar* if there exists a strong (timed) bisimulation relation between S_1 and S_2. We write $S_1 \approx_{\mathcal{S}} S_2$ in this case.

Let $S = (Q, Q_0, \Sigma_\varepsilon, \longrightarrow, F, R)$ be a TTS. We define the ε-abstract TTS $S^\varepsilon = (Q, Q_0^\varepsilon, \Sigma, \longrightarrow_\varepsilon, F, R)$ (which is not time-deterministic and has no ε-transitions) by:

– $s \overset{d}{\rightarrow}_\varepsilon s'$ iff there is a run $\rho = s \overset{*}{\rightarrow} s'$ with *Untimed*$(\rho) = \varepsilon$ and *Duration*$(\rho) = d$;

– $s \overset{a}{\rightarrow}_\varepsilon s'$ with $a \in \Sigma$ if and only if there is a run $\rho s \overset{*}{\rightarrow} s'$ with *Untimed*$(\rho) = a$ and *Duration*$(\rho) = 0$;

1. $s_2 \preceq^{-1} s_1 \iff s_1 \preceq s_2$.

– $Q_0^\varepsilon = \{s \mid \exists s' \in Q_0 \mid \ s' \xrightarrow{w} s$ *and* $\mathit{Duration}(w) = 0 \wedge \mathit{Untimed}(w) = \varepsilon\}$.

DEFINITION 4.5.– *[Weak Timed Similarity] Let* $S_1 = (Q_1, Q_0^1, \Sigma_\varepsilon, \longrightarrow_1, F_1, R_1)$ *and* $S_2 = (Q_2, Q_0^2, \Sigma_\varepsilon, \longrightarrow_2, F_2, R_2)$ *be two TTS and* $\preceq$ *be a binary relation over* $Q_1 \times Q_2$. $\preceq$ *is a weak (timed) simulation relation of* S_1 *by* S_2 *if it is a strong timed simulation relation of* S_1^ε *by* S_2^ε. *A TTS* S_2 *weakly simulates* S_1 *if there is a weak (timed) simulation relation of* S_1 *by* S_2. *We write* $S_1 \preceq_{\mathcal{W}} S_2$ *in this case.*

When there is a weak simulation relation $\preceq$ of S_1 by S_2 and $\preceq^{-1}$ is also a weak simulation relation of S_2 by S_1, we say that $\preceq$ is a *weak (timed) bisimulation relation* between S_1 and S_2 and use $\approx$ instead of $\preceq$. Two TTS S_1 and S_2 are *weakly (timed) bisimilar* if there exists a weak (timed) bisimulation relation between S_1 and S_2. We write $S_1 \approx_{\mathcal{W}} S_2$ in this case.

Note that if $S_1 \preceq_{\mathcal{S}} S_2$ then $S_1 \preceq_{\mathcal{W}} S_2$ and if $S_1 \preceq_{\mathcal{W}} S_2$ then $\mathcal{L}(S_1) \subseteq \mathcal{L}(S_2)$. Moreover, proving that $S_1 \preceq_{\mathcal{W}} S_2$ usually amounts to proving that if $q_1 \preceq q_2$, then each move $q_1 \xrightarrow{e}_1 q_1'$ can be simulated by a set of moves $q_2 \xrightarrow{e}_{2,\varepsilon} q_2'$ such that $q_2 \preceq q_2'$.

Let $S = (Q, Q_0, \Sigma_\varepsilon, \longrightarrow, F, R)$ be a TTS. We define the *time-abstract* TTS $S^\Delta = (Q, Q_0, \Sigma_\varepsilon \cup \{\delta\} \longrightarrow_\Delta, F, R)$ with $\delta \notin \Sigma_\varepsilon$ by:

– $s \xrightarrow{\delta}_\Delta s'$ iff $s \xrightarrow{d} s'$ for some $d \in \mathbb{R}_{\geqslant 0}$,

– $s \xrightarrow{a}_\Delta s'$ with $a \in \Sigma$ iff $s \xrightarrow{a} s'$ for some $a \in \Sigma_\varepsilon$.

Notice that S^Δ has no transition $s \xrightarrow{d} s'$ with $d \in \mathbb{R}_{\geqslant 0}$.

4.2.2. *Time Petri Nets*

Time Petri Nets (TPN) were introduced in [MER 74] and extend Petri Nets with timing constraints on the firings of transitions. In TPN, a time interval is associated with each transition. An implicit clock can then be associated with each enabled transition, and gives the elapsed time since it was last enabled. An enabled transition can be fired if its clock value belongs to the interval of the transition. Furthermore, time cannot progress beyond any upper bound of an interval associated with a transition. The following definitions formalize these principles. We consider here a generalized version[2] of TPN with accepting and repeated markings and prove our results for this general model.

DEFINITION 4.6.– *[Labeled Time Petri Nets] A Labeled Time Petri Net* $\mathcal{N}$ *is a tuple* $(P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, F, R)$ *where:* P *is a finite set of places and* T *is a finite*

2. This is required to be able to define Büchi timed languages, which is not possible in the original version of TPN of [MER 74].

set of transitions with $P \cap T = \emptyset$*;* $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ *is a finite set of actions and* ε *the empty word i.e. the silent action;* ${}^\bullet(.) \in (\mathbb{N}^P)^T$ *is the backward incidence mapping;* $(.)^\bullet \in (\mathbb{N}^P)^T$ *is the forward incidence mapping;* $M_0 \in \mathbb{N}^P$ *is the initial marking;* $\Lambda : T \to \Sigma_\varepsilon$ *is the labeling function;* $I : T \to \mathcal{I}(\mathbb{Q}_{\geqslant 0})$ *associates with each transition a firing interval;* $R \subseteq \mathbb{N}^P$ *is the set of final markings and* $F \subseteq \mathbb{N}^P$ *is the set of repeated markings. An unlabeled TPN is a TPN such that* $\Sigma = T$ *and* $\Lambda(t) = t$ *for all* $t \in T$*.*

A TPN $\mathcal{N}$ is a g-TPN if for all $t \in T$, $I(t)$ is an interval with bounds in $\mathbb{N}_g$. We also use ${}^\bullet t$ (resp. $t^\bullet$) to denote the set of places ${}^\bullet t = \{p \in P \mid {}^\bullet t(p) > 0\}$ (resp. $t^\bullet = \{p \in P \mid t^\bullet(p) > 0\}$) as it is common in other works[3].

Semantics of Time Petri Nets. The semantics of TPNs is given in terms of timed Transition Systems. A *marking* M of a TPN is a mapping in $\mathbb{N}^P$ and $M(p)$ is the number of tokens in place p. A transition t is *enabled* in a marking M iff $M \geqslant {}^\bullet t$. We denote by $\mathit{En}(M)$ the set of enabled transitions in M. To decide whether a transition t can be fired, we need to know for how long it has been enabled: if this amount of time lies within the interval $I(t)$, t can actually be fired, otherwise it cannot. On the other hand time can progress only if the enabling duration still belongs to the downward closure of the interval associated with an enabled transition. Let $\nu \in (\mathbb{R}_{\geqslant 0})^{\mathit{En}(M)}$ be a *valuation* such that each value $\nu(t)$ is the time elapsed since transition t was last enabled. A *configuration* of the TPN $\mathcal{N}$ is a pair (M, ν). An *admissible configuration* of a TPN is a configuration (M, ν) such that $\forall t \in \mathit{En}(M), \nu(t) \in I(t)^\downarrow$. We let $\mathit{ADM}(\mathcal{N})$ be the set of admissible markings.

When defining the semantics of a TPN, three kinds of policies must be fixed.

– **The choice policy** concerns the choice of the next event to be fired (scheduled). For TPNs (and also timed automata), this choice is non-deterministic (possible alternatives use priorities, probabilities, etc.).

– **The service policy** concerns the possibility of simultaneous instances of a same event to occur. In the context of Petri nets, this is formalized by the enabling *degree* of a transition. Here we adopt the *single-server policy* (at most one instance of a firing per transition in every state). The results presented are also valid for the other standard policies (multiple or infinite server), at least for the important case of bounded Petri nets. However, taking them explicitely into account would lead to intricate notations.

– **The memory policy** concerns the updating of timing information when a discrete step occurs. The key issue[4] in the semantics is to define when we reset the clock

3. Whether ${}^\bullet t$ (resp. $t^\bullet$) stands for a vector of $(\mathbb{N}^P)^T$ or a subset of P will be unambiguously defined by the context.

4. The new marking obtained after firing a transition t from a marking M is given by the untimed semantics of Petri Nets, i.e. $M' = M - {}^\bullet t + t^\bullet$.

measuring the time since a transition was last enabled. This can only happen when we fire a transition. We let $\uparrow enabled(t', M, t) \in \mathbb{B}$ be true if t' is *newly enabled* by the firing of transition t from marking M and false otherwise. Let M be a marking and $t \in En(M)$. The firing of t leads to a new marking $M' = M - {}^\bullet t + t^\bullet$. Three semantics are possible:

I: The *intermediate* semantics (I) considers that the firing of a transition is performed in two steps: consuming the input tokens in ${}^\bullet t$ and then producing output tokens in $t^\bullet$. The fact that a transition t' is newly enabled on the firing of a transition $t \neq t'$ is determined with respect to the intermediate marking $M - {}^\bullet t$. When a transition t is fired it is newly enabled irrespective of the intermediate marking. We denote by $\uparrow enabled_I(t', M, t)$ the newly enabled predicate in this case. This mapping is defined by:

$$\uparrow enabled_I(t', M, t) = \quad (t' \in En(M - {}^\bullet t + t^\bullet) \wedge \big(t' \notin En(M - {}^\bullet t) \vee (t = t')\big)$$

A: The *atomic semantics* considers that the firing of a transition is obtained by an atomic step. The corresponding mapping $\uparrow enabled_A(t', M, t)$ is defined by:

$$\uparrow enabled_A(t', M, t) = \quad (t' \in En(M - {}^\bullet t + t^\bullet)) \wedge \big(t' \notin En(M) \vee (t = t')\big)$$

PA: The *persistent atomic semantics* considers that the firing of a transition is also obtained by an atomic step. The difference with the A semantics in only on the value of $\uparrow enabled_A(t', M, t)$ when $t = t'$. The transition being fired is not always newly enabled:

$$\uparrow enabled_{PA}(t', M, t) = t' \in En(M - {}^\bullet t + t^\bullet) \wedge (t' \notin En(M))$$

Note that we have the relation:

$$\uparrow enabled_{PA}(t, M, t') \Rightarrow \uparrow enabled_A(t, M, t') \Rightarrow \uparrow enabled_I(t, M, t')$$

The intermediate semantics I, based on [BER 91, AUR 00] is the most common. However, depending on the systems to be modeled, another semantics may be more appropriate. The relative expressive power of the three semantics has not been investigated so far: we address this problem in section 4.3.

We now define the semantics of a TPN: this is a parameterized semantics that depends on the choice of the semantics for the $\uparrow enabled$ predicate.

DEFINITION 4.7.– *[Semantics of TPN] Let $s \in \{I, A, PA\}$. The s-semantics of a TPN $\mathcal{N} = (P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, F, R)$ is a timed transition system $S_\mathcal{N} = (Q, \{q_0\}, T, \rightarrow, F', R')$ where: $Q = ADM(\mathcal{N})$, $q_0 = (M_0, \mathbf{0})$, $F' = \{(M, \nu) \mid M \in F\}$ and $R' = \{(M, \nu) \mid M \in R\}$, and $\longrightarrow \in Q \times (T \cup \mathbb{R}_{\geqslant 0}) \times Q$ consists of the discrete and continuous transition relations:*

– the discrete transition relation is defined $\forall t \in T$ by: $(M, \nu) \overset{\Lambda(t)}{\rightarrow} (M', \nu')$ iff

- *$t \in En(M) \wedge M' = M - {}^\bullet t + t^\bullet$;*

- $\nu(t) \in I(t)$;
- $\forall t \in \mathbb{R}_{\geqslant 0}^{\mathit{En}(M')},\ \nu'(t) = \begin{cases} 0 & \text{if } \uparrow enabled_s(t', M, t), \\ \nu(t) & \text{otherwise.} \end{cases}$

– *the continuous transition relation is defined* $\forall d \in \mathbb{R}_{\geqslant 0}$ *by:*

$(M, \nu) \xrightarrow{d} (M, \nu')$ *iff* $\nu' = \nu + d$ *and* $\forall t \in \mathit{En}(M), \nu'(t) \in I(t)^{\downarrow}$

A run ρ *of* $\mathcal{N}$ *is an initial run of* $S_{\mathcal{N}}$. *The timed language accepted by* $\mathcal{N}$ *is* $\mathcal{L}(\mathcal{T}) = \mathcal{L}(S_{\mathcal{N}})$. *An unlabeled TPN accepts a timed language in* $(T \times \mathbb{R}_{\geqslant 0})^{\infty}$.

We simply write $(M, \nu) \xrightarrow{w}$ to emphasize that a sequence of transitions w can be fired in $S_{\mathcal{N}}$ from (M, ν). If *Duration*$(w) = 0$ we say that w is an *instantaneous firing sequence*. The set of *reachable markings* of $\mathcal{N}$ is $\mathsf{Reach}(\mathcal{N}) = \{M \in \mathbb{N}^P \mid \exists (M, \nu) \mid (M_0, \mathbf{0}) \xrightarrow{w} (M, \nu)\}$.

4.2.3. *Timed automata*

We first define the notion of timed automata.

DEFINITION 4.8.– *[Timed automata] A timed automaton* $\mathcal{A}$ *is a tuple* $(L, \ell_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ *where:* L *is a finite set of locations;* $\ell_0 \in L$ *is the initial location;* X *is a finite set of non-negative real-valued clocks;* $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ *is a finite set of actions and* ε *is the silent action;* $E \subseteq L \times \mathcal{C}(X) \times \Sigma_\varepsilon \times 2^X \times L$ *is a finite set of edges,* $e = \langle \ell, \gamma, a, R, \ell' \rangle \in E$ *represents an edge from the location* ℓ *to the location* ℓ' *with the guard* γ, *the label* a *and the reset set* $R \subseteq X$; $Inv \in \mathcal{C}(X)^L$ *assigns an invariant to any location. We restrict the invariants to conjuncts of terms of the form* $x \preceq r$ *for* $x \in X$ *and* $r \in \mathbb{N}$ *and* $\preceq \in \{<, \leqslant\}$. $F \subseteq L$ *is the set of final locations and* $R \subseteq L$ *is the set of repeated locations.*

DEFINITION 4.9.– *[Semantic of timed automata] The semantics of a timed automaton* $\mathcal{A} = (L, \ell_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ *is a timed transition system* $S_{\mathcal{A}} = (Q, q_0, \Sigma_\varepsilon, \rightarrow,$ $F', R')$ *with* $Q = L \times (\mathbb{R}_{\leqslant 0})^X$, $q_0 = (\ell_0, \mathbf{0})$ *is the initial state,* $F' = \{(\ell, v) \mid \ell \in F\}$ *and* $R' = \{(\ell, v) \mid \ell \in R\}$, *and* $\rightarrow$ *is defined by:*

$$(\ell, v) \xrightarrow{a} (\ell', v') \text{ if and only if } \exists\, (\ell, \gamma, a, R, \ell') \in E \text{ such that } \begin{cases} \gamma(v) = \mathit{tt}, \\ v' = v[R \mapsto 0] \\ Inv(\ell')(v') = \mathit{tt} \end{cases}$$

$$(\ell, v) \xrightarrow{d} (\ell', v') \text{ if and only if } \begin{cases} \ell = \ell' \qquad v' = v + d \quad \text{and} \\ \forall\, 0 \leqslant d' \leqslant d,\ Inv(\ell)(v + d') = \mathit{tt} \end{cases}$$

A run ρ *of* $\mathcal{A}$ *is an initial run of* $S_{\mathcal{A}}$. *The timed language accepted by* $\mathcal{A}$ *is* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(S_{\mathcal{A}})$.

Recall [ALU 94] that, if m is the maximal constant appearing in atomic formulae $x \bowtie c$ of $\mathcal{A}$, an equivalence relation with finite index can be defined on clock valuations, leading to a partition of $(\mathbb{R}_{\geqslant 0})^X$, with the following property: two equivalent valuations have the same behavior under progress of time and reset operations, with respect to the constraints. Note that a partition using any $K \geqslant m$ would have the same property. Also, the construction can be extended to a g-grid, by taking all constants of the form $\frac{i}{g}$, $0 \leqslant i \leqslant K{\cdot}g$ instead of $\{0, 1, \ldots, K\}$. Finally, taking $K = +\infty$ (as depicted in Figure 4.1 on the left) leads to a similar structure except for the fact that the partition is infinite. When it is possible, we will sometimes use such a partition in order to simplify some proofs. Indeed, with this partition, the extremal case where x is greater than K is not to be distinguished from the standard case.

In this chapter, the elements of the partition are called *elementary zones* and we consider a slight variation for the definition of elementary zones: we take the constant $K = m+1$ and with each clock $x \in X$, we associate an interval in the set $\{\{0\},]0, 1[, \{1\}, \ldots, \{K-1\},]K-1, K[, [K, +\infty[\}$, instead of keeping $\{K\}$ separately. As usual, we also specify the ordering on the fractional parts for all clocks x such that $x < K$. Such a partition is represented in Figure 4.1 (on the right) for the set of two clocks $X = \{x, y\}$ and $K = 3$. For this example, elementary zones Z_1 and Z_2 are described by the constraints: $Z_1 : (2 < x < 3) \wedge (1 < y < 2) \wedge (0 < frac(y) < frac(x))$ and $Z_2 : (x \geqslant 3) \wedge (1 < y < 2)$. When considering diagonal constraints (also with constants up to K), another partition (Figure 4.1 in the middle, with $K = 2$) must be considered.

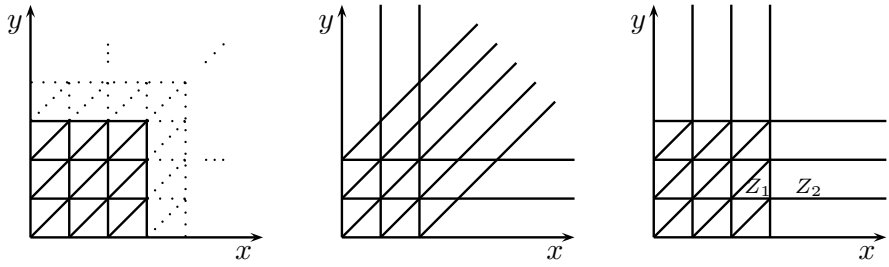


Figure 4.1. *Partitions of* $(\mathbb{R}^+)^2$ *with* $K = +\infty$, $K = 2$ *(with diagonal constraints) and* $K = 3$ *(no diagonal constraints)*

The future of a zone Z is defined by $fut(Z) = \{v + d \mid v \in Z, d \in \mathbb{R}_{\geqslant 0}\}$. If Z and Z' are elementary zones, Z' is a time successor of Z, written $Z \leqslant Z'$, if for each valuation $v \in Z$, there is some $d \in \mathbb{R}_{\geqslant 0}$ such that $v + d \in Z'$. For each elementary zone Z, there is at most one elementary zone such that (i) Z' is a time successor of Z, (ii) $Z \neq Z'$ and (iii) there is no time successor Z'' such that $Z \leqslant Z'' \leqslant Z'$. When

it exists, this elementary zone is called the immediate successor of Z and is denoted by $succ(Z)$. Note that $fut(Z) \subseteq \cup_{Z \leqslant Z'} Z'$, with a strict inclusion when no diagonal constraints are permitted.

Finally recall that a finite automaton $R(\mathcal{A})$, called the *region automaton*, can be built from $\mathcal{A}$. This automaton is time abstract bisimilar to the original automaton $\mathcal{A}$. Its states, called here *regions*, are of the form (ℓ, Z), where ℓ is a location of $\mathcal{A}$ and Z an elementary zone of $(\mathbb{R}_{\geqslant 0})^X$. They are built from the initial region $(\ell_0, \mathbf{0})$ by transitions of the form $(\ell, Z) \xrightarrow{\delta} (\ell, Z')$ for a time successor Z' of Z, if $Inv(\ell)(Z) =$ tt or $(\ell, Z) \xrightarrow{a} (\ell', Z')$ if there is a transition $(\ell, \gamma, a, R, \ell') \in E$ such that $\gamma(Z) =$ tt and $Z' = Z[R \mapsto 0]$, with $Inv(\ell')(Z') =$ tt. A region (ℓ, Z) is said to be maximal in $R(\mathcal{A})$ with respect to ℓ if no δ-transition is possible from (l, Z). The automaton $R(\mathcal{A})$ is restricted to the regions reachable from the initial region $(\ell_0, \mathbf{0})$, and accepts the language:

$$Untime(L(\mathcal{A})) = \{a_1 a_2 \ldots \mid (a_1, d_1)(a_2, d_2) \ldots \in L(\mathcal{A}) \text{ for some } d_1, d_2, \ldots \in \mathbb{R}_{\geqslant 0}\}$$

We also consider another automaton, called *class automaton*, in which the states, called *classes*, are of the form $(l, fut(Z) \cap Inv(\ell))$, where Z is a zone. In this case, the second component is not an elementary zone anymore (but a general zone) and the automaton is built from the initial class $(\ell_0, fut(\mathbf{0}) \cap Inv(\ell_0))$ by the following transitions: $(\ell, Z_1) \xrightarrow{a} (\ell', Z_2)$ if there exists $(l, \gamma, a, R, l') \in E$ such that $Z_1 \cap [\![\gamma]\!] \neq \emptyset$, and $Z_2 = fut((Z_1 \cap [\![\gamma]\!])[R \mapsto 0]) \cap Inv(\ell')$.

Note that the class automaton also accepts $Untime(L(\mathcal{A}))$. Moreover, since a class can be represented by a Difference Bounded Matrix [DIL 89], its size is at most $(4K + 2)^{(|X|+1)^2}$, which is exponential in the size of $\mathcal{A}$, as for the region automaton.

4.2.4. *Expressiveness and equivalence problems*

If B, B' are either TPN or TA, we write $B \approx_{\mathcal{S}} B'$ (resp. $B \approx_{\mathcal{W}} B'$) for $S_B \approx_{\mathcal{S}} S_{B'}$ (resp. $S_B \approx_{\mathcal{W}} S_{B'}$). Let $\mathcal{C}$ and $\mathcal{C}'$ be two classes of TPNs or TA.

DEFINITION 4.10.– *[Expressiveness with respect to timed language acceptance] The class $\mathcal{C}$ is more expressive than $\mathcal{C}'$ with respect to timed language acceptance if for all $B' \in \mathcal{C}'$ there is a $B \in \mathcal{C}$ such that $\mathcal{L}(B) = \mathcal{L}(B')$. We write $\mathcal{C}' \leqslant_{\mathcal{L}} \mathcal{C}$ in this case. If moreover there is some $B \in \mathcal{C}$ such that there is no $B' \in \mathcal{C}'$ with $\mathcal{L}(B) = \mathcal{L}(B')$, then $\mathcal{C}' <_{\mathcal{L}} \mathcal{C}$ (read "strictly more expressive"). If both $\mathcal{C}' \leqslant_{\mathcal{L}} \mathcal{C}$ and $\mathcal{C} \leqslant_{\mathcal{L}} \mathcal{C}'$, then*

$\mathcal{C}$ and $\mathcal{C}'$ are equally expressive with respect to timed language acceptance, and we write $\mathcal{C} =_{\mathcal{L}} \mathcal{C}'$.

DEFINITION 4.11.– *[Expressiveness with respect to timed bisimilarity] The class $\mathcal{C}$ is more expressive than $\mathcal{C}'$ with respect to strong (resp. weak) timed bisimilarity if for all $B' \in \mathcal{C}'$ there is a $B \in \mathcal{C}$ such that $B \approx_{\mathcal{S}} B'$ (resp. $B \approx_{\mathcal{W}} B'$). We write $\mathcal{C}' \leqslant_{\mathcal{S}} \mathcal{C}$ (resp. $\mathcal{C}' \leqslant_{\mathcal{W}} \mathcal{C}$) in this case. If moreover there is a $B \in \mathcal{C}$ such that there is no $B' \in \mathcal{C}'$ with $B \approx_{\mathcal{S}} B'$ (resp. $B \approx_{\mathcal{W}} B'$), then $\mathcal{C}' <_{\mathcal{S}} \mathcal{C}$ (resp. $\mathcal{C}' <_{\mathcal{W}} \mathcal{C}$). If both $\mathcal{C}' <_{\mathcal{S}} \mathcal{C}$ and $\mathcal{C} <_{\mathcal{S}} \mathcal{C}'$ (resp. $<_{\mathcal{W}}$), then $\mathcal{C}$ and $\mathcal{C}'$ are equally expressive with respect to strong (resp. weak) timed bisimilarity and we write $\mathcal{C} \approx_{\mathcal{S}} \mathcal{C}'$ (resp. $\mathcal{C} \approx_{\mathcal{W}} \mathcal{C}'$).*

Later on we will compare various subclasses of TPNs and TA. We denote by $\mathcal{TPN}$ the class of TPNs and by $\mathcal{TA}$ the class of TA, according to definitions 4.6 and 4.8. We recall the following theorem adapted from [CAS 04]:

THEOREM 4.1.– *For any $\mathcal{N} \in$ B-$\mathcal{TPN}_{\varepsilon}$ there is a TA $\mathcal{A}$ such that $\mathcal{N} \approx_{\mathcal{W}} \mathcal{A}$, hence B-$\mathcal{TPN}_{\varepsilon} \leqslant_{\mathcal{W}} \mathcal{TA}_{\varepsilon}$.*

4.3. Comparison of semantics I, A and PA

In section 4.3.1, we establish two relations between these semantics for TPN, which hold in the general case. In section 4.3.2, we complete these results with a third one, restricted to bounded time Petri nets, with only closed intervals for transitions. Since we prove results concerning weak timed bisimulation, we consider unlabeled TPN, where all states are final and repeated states. For all figures in this section, a transition is filled in black when its firing interval is $[0, 0]$.

4.3.1. *A first comparison between the different semantics of TPNs*

We first show the following result.

PROPOSITION 4.1.– *Let $\mathcal{N}$ be a time Petri net with intermediate semantics. There exists a TPN $\overline{\mathcal{N}}$ with atomic semantics which is (weakly timed) bisimilar to $\mathcal{N}$.*

Proof. The construction is quite easy. The set of places of $\overline{\mathcal{N}}$ is obtained by adding to the set of places of $\mathcal{N}$ a new place for each transition t from $\mathcal{N}$: $\overline{P} = P \cup \{p_t, t \in T\}$. The transitions T of $\mathcal{N}$ are duplicated in $\overline{\mathcal{N}}$: $\overline{T} = T^+ \cup T^-$ and the construction follows Figure 4.2, from left to right.

We consider the equivalence relation $\mathcal{R}$ which contains all pairs $((M, \nu), (\overline{M}, \overline{\nu}))$ such that:

– for all $p \in P$, $M(p) = \overline{M}(p) + \Sigma_{t \in T} t^{\bullet}(p).\overline{M}(p_t)$;

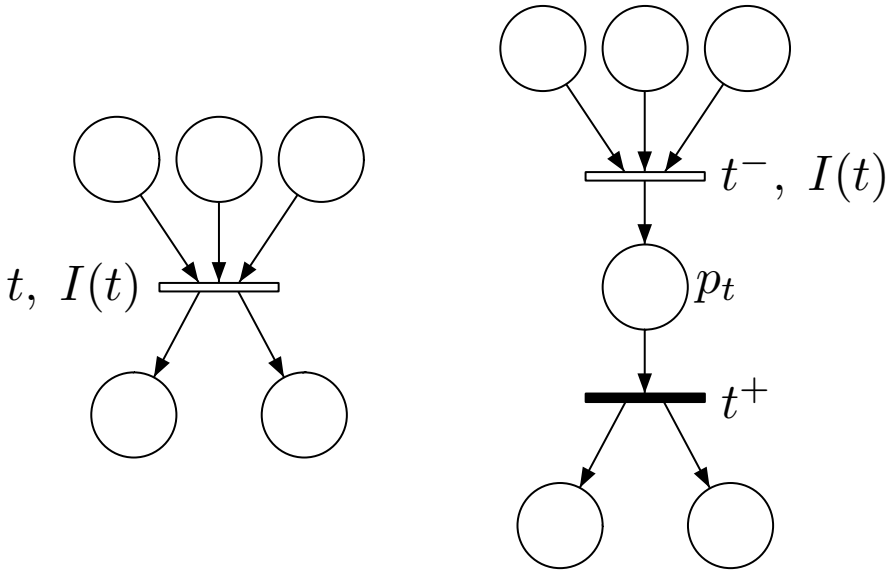


Figure 4.2. *From I to A*

– for all $t \in En(M)$, $\nu(t) = \overline{\nu}(t^-)$ if t^- is enabled in $\overline{M}$ and 0 otherwise. The latter case corresponds in $\mathcal{N}$ to a newly enabled transition.

To prove that $\mathcal{R}$ is a bisimulation, we first note that, with the definition above for markings, from any configuration $(\overline{M}, \overline{\nu})$, we can reach instantaneously a configuration $(\overline{M}_1, \overline{\nu}_1)$ such that $\overline{M}_1(p_t) = 0$ for all t, with the firing of a (possibly empty) sequence of transitions in T^+. Moreover, the relation between valuations implies that $(\overline{M}_1, \overline{\nu}_1)$ is still equivalent to (M, ν).

Consider now a pair $((M, \nu), (\overline{M}, \overline{\nu})) \in \mathcal{R}$:

– if $(M, \nu) \xrightarrow{t} (M', \nu')$, then from the remark above, we first fire a sequence from $\overline{M}$ to empty all places p_t, leading instantaneously to $(\overline{M}_1, \overline{\nu}_1)$, which is equivalent to (M, ν). Then transition t^- can be fired from $(\overline{M}_1, \overline{\nu}_1)$, immediately followed by t^+, leading to $(\overline{M}', \overline{\nu}')$, where all places p_t are empty again. Moreover, the transitions newly enabled by t^+ in $\overline{\mathcal{N}}$ are exactly those which were newly enabled by t in $\mathcal{N}$, so that $(M', \nu')\mathcal{R}(\overline{M}', \overline{\nu}')$.

– Conversely, suppose that a transition is fired from $(\overline{M}, \overline{\nu})$ in $\overline{\mathcal{N}}$. If the transition is some t^+, then the new configuration $(\overline{M}_1, \overline{\nu}_1)$ is still equivalent to (M, ν) (as above),

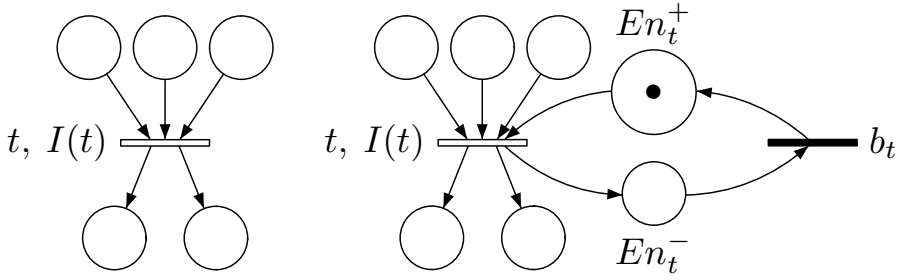


Figure 4.3. *From A to PA*

thus no move at all is necessary in $\mathcal{N}$. If $(\overline{M},\overline{\nu}) \stackrel{t^-}{\hookrightarrow} (\overline{M}',\overline{\nu}')$, then t can be fired from (M,ν) and the resulting marking is, (M',ν'), equivalent to $(\overline{M}',\overline{\nu}')$.

– if $(M,\nu) \stackrel{d}{\rightarrow} (M,\nu+d)$, for some delay d, then again we have to apply the emptying sequence from $(\overline{M},\overline{\nu})$, to reach a configuration $(\overline{M}_1,\overline{\nu}_1)$ which is still equivalent to (M,ν), where time can elapse. The relation between ν and $\overline{\nu}_1$ implies that this is possible, leading to $(\overline{M}_1,\overline{\nu}_1+d)$.

– Conversely, if $(\overline{M},\overline{\nu}) \stackrel{d}{\rightarrow} (\overline{M},\overline{\nu}+d)$, then all places p_t are empty in $\overline{M}$, so that the move $(M,\nu) \stackrel{d}{\rightarrow} (M,\nu+d)$ is also possible in $\mathcal{N}$.

Thus $\mathcal{R}$ is a bisimulation. ∎

PROPOSITION 4.2.– *Let $\mathcal{N}$ be a time Petri net with atomic semantics. There exists a TPN $\overline{\mathcal{N}}$ with persistent atomic semantics which is (weakly timed) bisimilar to $\mathcal{N}$.*

Proof. Here again, the construction is simple. Note that the only difference between the two semantics concerns the question of whether a transition t can newly enable itself. With atomic semantics, this is the case as soon as t is enabled in the new marking while with persistent atomic semantics, this is never possible. In order to ensure that a transition t will be newly enabled if it is enabled in the new marking, we add an input place En_t^+ and an output place En_t^- to the transition, with an instantaneous loop b_t leading back to En_t^+, once the transition has been fired. The construction is represented in Figure 4.3, again from left to right.

We consider the equivalence relation $\mathcal{R}$ which contains all pairs $((M,\nu),(\overline{M},\overline{\nu}))$ such that:

– $M(p) = \overline{M}(p)$ for all places p in P;

– for a transition $t \in En(M)$, $\nu(t) = \overline{\nu}(t)$ if t is enabled in $\overline{M}$ and 0 otherwise. Again the latter case corresponds in $\mathcal{N}$ to a newly enabled transition.

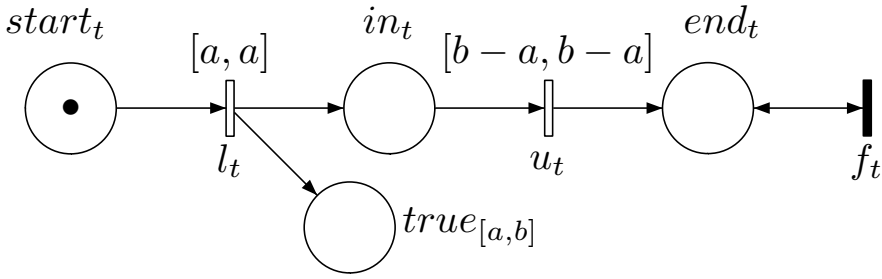


Figure 4.4. *Time subnet for transition t with interval $[a,b]$*

Like in the previous proposition, the proof is mostly based on the fact that from any configuration $(\overline{M},\overline{\nu})$, we can reach instantaneously a configuration $(\overline{M}_1,\overline{\nu}_1)$ such that $\overline{M}_1(En_t^+)=1$ for all t, with the firing of a (possibly empty) sequence of transitions b_t, with again $(\overline{M}_1,\overline{\nu}_1)$ still equivalent to (M,ν). ∎

4.3.2. *A second comparison for standard bounded TPN*

We now restrict ourselves to bounded TPNs, with the standard definition, *i.e.* with closed intervals ($[a,b]$ or $[a,\infty[$) for the transitions. Thus, this third result holds only for the subclass B-$\mathcal{TPN}(\leqslant,\geqslant)$.

PROPOSITION 4.3.– *Let $\mathcal{N}$ be a TPN in B-$\mathcal{TPN}(\leqslant,\geqslant)$ with persistent atomic semantics. There exists a TPN $\overline{\mathcal{N}}$ with intermediate semantics which is (weakly timed) bisimilar to $\mathcal{N}$.*

In this case, the construction of $\overline{\mathcal{N}}$ is more involved. Like above, we show how to simulate a transition t equipped with interval $[a,b]$, for $a \leqslant b$, or $[a,+\infty[$. We first build a time subnet for t (Figure 4.4), in order to simulate time elapsing since a reset operation until reaching (and staying inside) interval $[a,b]$. The token is in place $start_t$ if the transition is enabled in the initial marking. The double arrow at the end indicates that the place $term_t$ is both an input and an output place for the corresponding transition: time cannot progress. Of course, the time subnet for a transition with interval $[a,+\infty[$ is reduced by removing u_t, end_t and f_t.

Now, using the fact that the TPN is bounded, we consider its upper bound B and we associate with each place p a complementary place $\overline{p}$ such that for any reachable marking M, $M(\overline{p}) = B - M(p)$. Figure 4.5 represents a part of the subnet (on the right) for transition t (on the left), where $test_1$ is the beginning of the test step for what timing updates are required by the firing of t, and $Mutex$ ensures that the updates are done (instantaneously) before anything else, as explained further on.

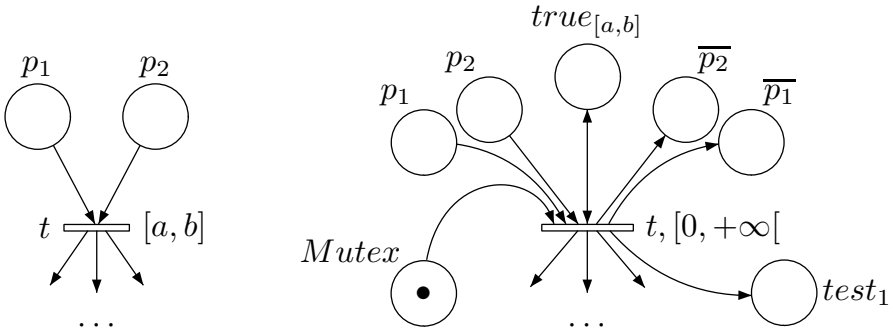


Figure 4.5. *From PA to I*

The remaining part of $\overline{\mathcal{N}}$ is devoted to the test of the other transitions from the original TPN, including t itself. Consider a given transition (for example, t_i), with again two input places p_1^i and p_2^i. The corresponding subnet consists of 4 modules, one for each case, depending on whether t_i can be fired or not before and after t. For this, two additional places are associated with t_i: E_{t_i}, which contains a token if t_i was enabled before the firing of t and NE_{t_i} which is its complementary place. If t_i is initially enabled, then E_{t_i} is initially marked otherwise NE_{t_i} is marked. This group of 4 modules has a common input place $test_i$ and a common output place $test_{i+1}$, which means that the tests are to be executed sequentially (and instantaneously), except for the last one where all outgoing transitions are linked to $Mutex$. These places are not shown in the following figures.

Case 1: transition t_i is enabled both before and after t. To test this case, we use the simple module on the left of Figure 4.6, where E_{t_i} (test before t) and p_1^i and p_2^i (test after t) are input and output places.

Case 2: t_i is not enabled before but it is enabled after t. The module is very similar to the previous one and is on the right of Figure 4.6. Note that, in this case only and because of the PA semantics, there must be a reset on the valuation of the transition, which explains why the initial place $start_{t_i}$ of the time module for t_i is an output place.

Case 3: t_i is enabled neither before nor after t. To test this, we must find an input place of t_i, where the current number of tokens disable t_i. Here is the point where the boundedness hypothesis is required. In order to perform this test, we check whether $B - {}^\bullet t_i(p) + 1$ tokens can be removed from a complementary place $\overline{p}$.

Case 4: t_i is enabled before but not after t. In this case, we have a module (see Figure 4.8) similar to the one above, except that we must also test for all the different

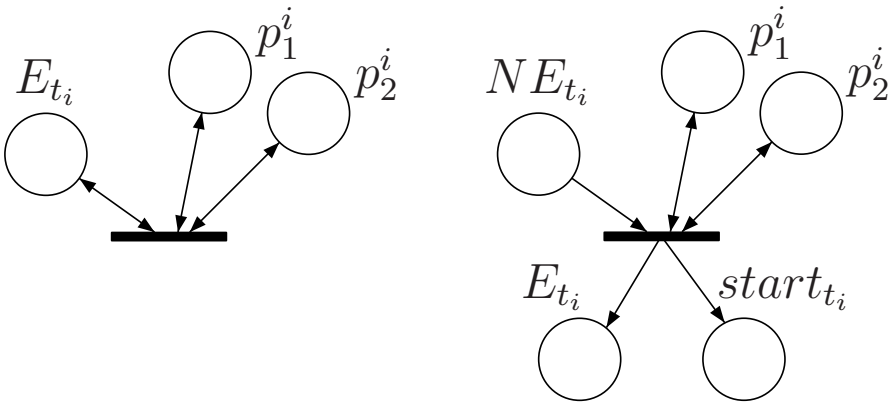


Figure 4.6. *Testing transition t_i: cases 1 and 2*

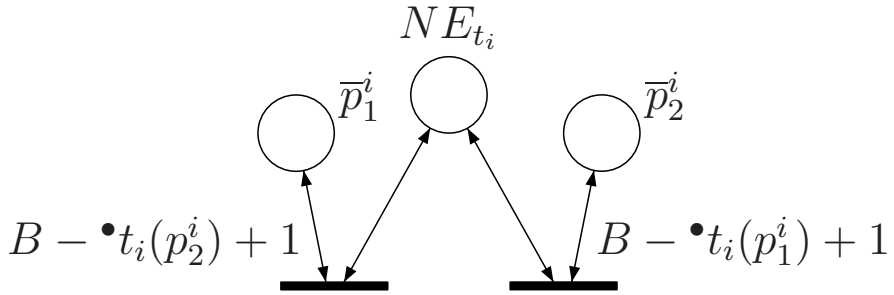


Figure 4.7. *Testing transition t_i: case 3*

configurations of the time subnet corresponding to t_i in order to disable the transitions by removing the tokens.

It can be seen in Figure 4.8 that there is a transition for each pair $(\overline{p}, state)$, where p is an input place of t_i and $state$ may be either the place $start_{t_i}$, the pair of places $(in_{t_i},\ true_{[a_i,b_i]})$ or the pair $(end_{t_i},\ true_{[a_i,b_i]})$. Like above, an edge from $\overline{p}$ to a transition must be labeled with $B - {}^\bullet t_i(p) + 1$ (which is omitted in the figure).

We consider the equivalence relation $\mathcal{R}$ containing all pairs $((M, \nu), (\overline{M}, \overline{\nu}))$ such that:

– M in $\mathcal{N}$ is obtained by projection: $M(p) = \overline{M}(p)$ for each place $p \in P$;

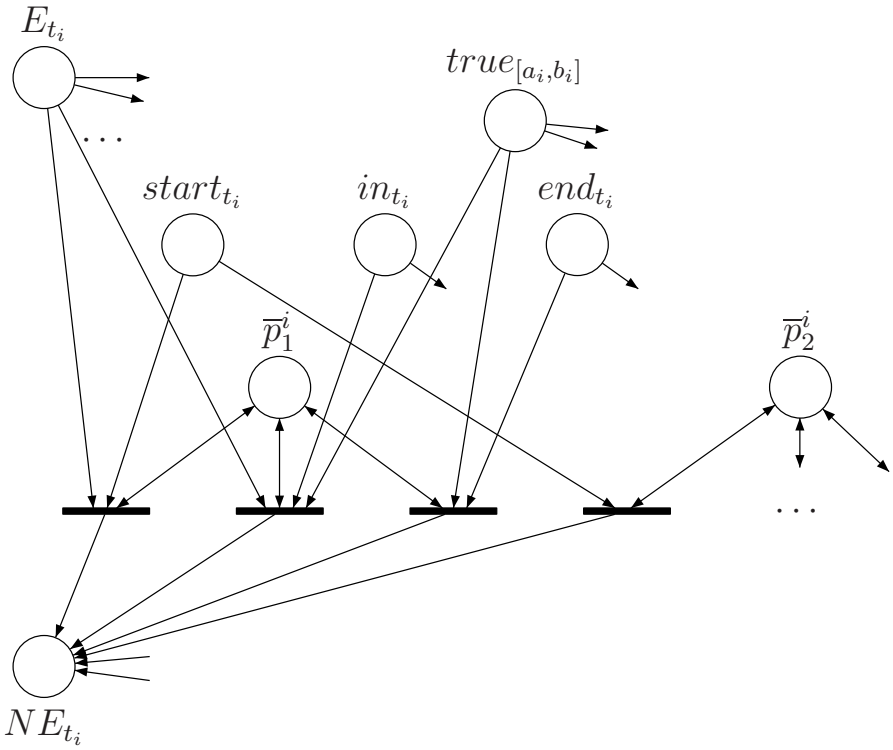


Figure 4.8. *Testing transition t_i: case 4*

– for a transition t in T enabled by M: $\nu(t) = 0$ if the time subnet of t is empty, $\nu(t) = \overline{\nu}(l_t)$ if the place $start_t$ contains a token, $\nu(t) = a + \overline{\nu}(u_t)$ if the place in_t contains a token and $\nu(t) = b$ if the place end_t contains a token. Note that in both latter cases, $true_{[a,b]}$ also contains a token and the transition t can be fired in $\overline{\mathcal{N}}$.

Also note that if $\overline{M}(start_t) = 1$ and $\overline{\nu}(u_t) = a$, then with instantaneously firing u_t, transition t can also be fired. By a development similar to the previous ones, we can show that $\mathcal{R}$ is a bisimulation relation.

More precisely, the proof is mainly based on emptying sequences from a configuration $(\overline{M}, \overline{\nu})$ of $\overline{\mathcal{N}}$: it is always possible to reach instantaneously a configuration $(\overline{M}_1, \overline{\nu}_1)$ such that the testing subnet is empty, with $(\overline{M}_1, \overline{\nu}_1)$ still equivalent to (M, ν). The details are omitted.

We can conclude this section with:

THEOREM 4.2.– *For the class B-$\mathcal{TPN}(\leqslant,\geqslant)$, the three semantics I, A and PA are equally expressive with respect to weak time bisimulation.*

4.4. Strict ordering results

In this section, we establish results proving that $\mathcal{TPN}$ are strictly less expressive with respect to weak timed bisimilarity than various classes of TA. For this, we consider the two automata $\mathcal{A}_0 \in \mathcal{TA}(<)$ and $\mathcal{A}_1 \in \mathcal{TA}(\leqslant)$ in Figure 4.9.

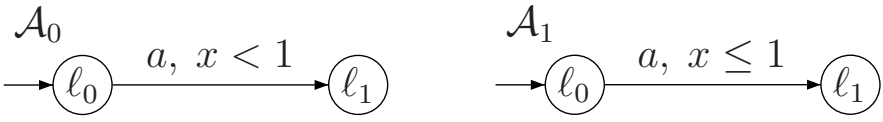


Figure 4.9. *Timed automata $\mathcal{A}_0$ and $\mathcal{A}_1$*

We will prove that no TPN can be weakly timed bisimilar to either $\mathcal{A}_0$ or $\mathcal{A}_1$. The proof relies on the following lemma, which states that in a TPN, waiting in some marking cannot disable transitions. The proof is easy and is thus omitted. Note that the results holds without modification for any semantics of $\uparrow enabled_s(t', M, t)$.

LEMMA 4.1.– *Let (M,ν) be an admissible configuration of a TPN, $d \in \mathbb{R}_{\geqslant 0}$ and let $w = t_1 t_2 \cdots t_k$ be an instantaneous firing sequence. If $(M,\nu) \xrightarrow{w}$, then $(M,\nu+d) \xrightarrow{w}$.*

THEOREM 4.3.– *There is no TPN weakly timed bisimilar to $\mathcal{A}_0$.*

Proof. Assume that there is a TPN $\mathcal{N}$ that is weakly timed bisimilar to $\mathcal{A}_0$ and let $\approx$ be a weak timed bisimulation between $S_{\mathcal{N}}$ and $S_{\mathcal{A}_0}$. Let $(M_0, \mathbf{0})$ be the initial state of $S_{\mathcal{N}}$ and $(\ell_0, v(x) = 0)$ the initial state of $S_{\mathcal{A}_0}$. In $S_{\mathcal{A}_0}$ there is a run of duration 1 leading to configuration $(\ell_0, 1)$ and thus there is a run $(M_0, \mathbf{0}) \xrightarrow{\varepsilon^{i_0} d_1 \varepsilon^{i_1} d_2 \varepsilon^{i_2} \cdots d_n \varepsilon^{i_n}} (M_1, \nu_1)$ in $\mathcal{N}$, with $i_k \geqslant 1$ for $1 \leqslant k \leqslant n-1$, $i_0 \geqslant 0$, $i_n \geqslant 0$ and $\sum_{1 \leqslant k \leqslant n} d_k = 1$. We can further assume $d_k > 0$ for all k and also $i_n = 0$ because the configuration reached after d_n is also bisimilar to $(\ell_0, 1)$.

Then $(M_0, \mathbf{0}) \xrightarrow{\varepsilon^{i_0} d_1 \varepsilon^{i_1} d_2 \varepsilon^{i_2} \cdots d_{n-1} \varepsilon^{i_{n-1}}} (M', \nu')$, where (M', ν') is bisimilar to a configuration (ℓ_0, d') with $d' = 1 - d_n < 1$. This entails that $(M', \nu') \xrightarrow{\varepsilon^* a}$.

Since $(M', \nu') \xrightarrow{d_n} (M_1, \nu_1)$, if follows from Lemma 4.1 that $(M_1, \nu_1) \xrightarrow{\varepsilon^* a}$, thus contradicting the fact that $(M_1, \nu_1) \approx (\ell_0, 1)$ from which no a can be fired. ∎

The result is also true with large constraints:

THEOREM 4.4.– *There is no TPN weakly timed bisimilar to* $\mathcal{A}_1$.

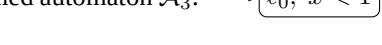
Proof. Again assume that there is a TPN $\mathcal{N}$ that is weakly timed bisimilar to $\mathcal{A}_1$. Since $(\ell_0, 0) \xrightarrow{1} (\ell_0, 1)$, we have $(M_0, \mathbf{0}) \xrightarrow{1}_{\varepsilon} (M_1, \nu_1)$, where $(\ell_0, 1)$ and (M_1, ν_1) are weakly timed bisimilar. Since a can be fired from $(\ell_0, 1)$, a transition labeled a can also be fired from all the configurations (M_1', ν_1') reachable from (M_1, ν_1) in null duration (ε transitions). Also there must be one such configuration (M', ν') such that some duration $d > 0$ can elapse from (M', ν') reaching (M'', ν''). According to Lemma 4.1, some a can be fired from (M'', ν''). However, (M'', ν'') is weakly timed bisimilar to the configuration $(\ell_0, 1 + d)$ which prevents a being fired. Hence, a contradiction. ∎

From Theorems 4.1, 4.3 and 4.4 we immediately obtain:

THEOREM 4.5.– $\mathcal{TPN} <_{\mathcal{W}} \mathcal{TA}(<)$ *and* $\mathcal{TPN} <_{\mathcal{W}} \mathcal{TA}(\leqslant)$.

The next theorem shows that the expressive power of TPNs depends on the chosen semantic even in the bounded case.

THEOREM 4.6.– *There exists a bounded TPN* $\mathcal{N}$ *with persistent atomic semantics such that no TPN (even unbounded) with atomic semantics is bisimilar to* $\mathcal{N}$.

Proof. Consider the following (Zeno) timed automaton $\mathcal{A}_3$: 

It is bisimilar to the TPN with PA semantics composed by a single transition t labeled by ε with firing interval $[0, 1[$ (or any interval $]a, 1[$ or $[a, 1[$).

Suppose that there is a TPN $\mathcal{N}$ with atomic semantics bisimilar to $\mathcal{A}_3$ and let d_{min} be the minimum of the non-zero upper bounds occuring in the intervals associated with the transitions of $\mathcal{N}$ and 0.5 (in fact any value less than 1 would be convenient).

There must be a sequence $(M_0, \nu_0) \xrightarrow{d_0 t_1 \cdots t_k d_k} (M, \nu)$ with $\Sigma_{i=0}^{k} d_i = 1 - d_{min}/2$ and (M, ν) bisimilar to $(\ell_0, 1 - d_{min}/2)$.

From (M, ν), we fire or disable the transitions enabled at this configuration, which leads to a new configuration (M', ν') bisimilar to some $(\ell_0, 1 - \delta')$ with $0 < \delta' \leqslant d_{min}/2$.

Now, since (M', ν') is bisimilar to $(\ell_0, 1 - \delta')$, there must be a sequence $(M', \nu') \xrightarrow{d_0' t_1' \cdots t_{k'}' d_{k'}'}$ with $0 < \Sigma_{i=0}^{k'} d_i' < \delta'$.

Choose the first $d_i' > 0$ and let (M^*, ν^*) be the state reached before the duration d_i'. Since time may elapse in this state, all enabled transitions have non-zero upper bounds for their interval, hence these bounds are greater than or equal to d_{min}.

Since the transitions have been enabled at or after configuration (M', ν'), we have $\forall t, \nu^*(t) \leqslant d_{min}/2 - \delta' < d_{min}/2$, thus $(M^*, \nu^*) \overset{d_{min}/2}{\rightarrow}$. However, (M^*, ν^*) is bisimilar to $(\ell_0, 1 - \delta')$ which cannot let time elapse for a duration of $d_{min}/2$. This is a contradiction. ∎

Following this negative result, we compare the expressiveness of TPNs and TA with respect to to timed language acceptance and exhibit a subclass of TA that admits bisimilar TPNs.

4.5. Equivalence with respect to timed language acceptance

In this section, we prove that TA and labeled TPNs are equally expressive with respect to timed languages acceptance and give an effective syntactical translation from TA to TPNs.

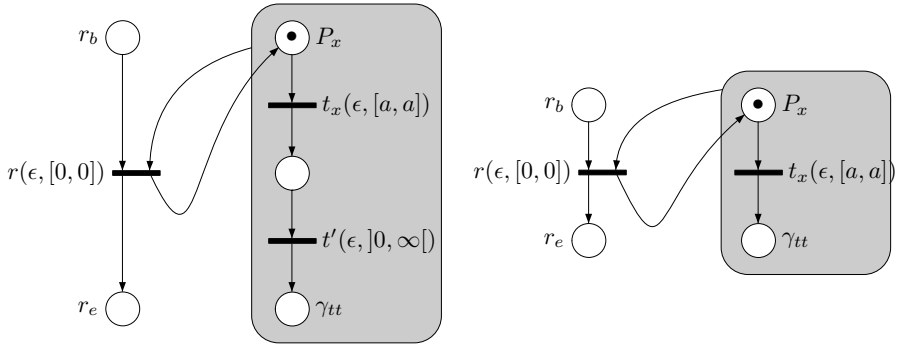
Let $\mathcal{A} = (L, l_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ be a TA. Since we are concerned in this section with the language accepted by $\mathcal{A}$ we assume the invariant function is uniformly true. Let $\mathcal{C}_x$ be the set of atomic constraints on clock x that are used in $\mathcal{A}$. The Time Petri Net resulting from our translation will be built from "elementary blocks" modeling the truth value of the constraints in $\mathcal{C}_x$. We next link them with blocks for resetting clocks. In the next section we show how to encode atomic constraints into TPNs.

As a consequence of Corollary 4.2, the semantics I, A and *PA* for TPNs are equivalent with respect to language acceptance. In this section, we use the I semantics.

4.5.1. *Encoding atomic constraints*

Let $\varphi \in \mathcal{C}_x$ be an atomic constraint on x. From φ, we define the TPN $\mathcal{N}_\varphi$ given by the widgets of Figure 4.10 ((a) and (b)) and Figure 4.11. In the figures, a transition is written $t(\ell, I)$ where t is the name of the transition, $\ell \in \Sigma_\varepsilon$ and $I \in \mathcal{I}(\mathbb{Q}_{\geqslant 0})$.

To avoid drawing too many arcs, we have adopted the following semantics: the grey box is seen as a macro place; an arc from this grey box means that there are as many copies of the transition as places in the grey box. For instance, the TPN of Figure 4.10(b) has 2 copies of the target transition r: one with input places P_x and r_b and output places r_e and P_x and another fresh copy of r with input places r_b and γ_{tt} and output places r_e and P_x. Note that in the widgets of Figure 4.11 we put a token in γ_{tt} when firing r only on the copy of r with input place P_i (otherwise the number of tokens in place γ_{tt} could be unbounded). Also, we assume that the automaton $\mathcal{A}$ has no constraint $x \geqslant 0$ (as it evaluates to true they can be safely removed) and thus that the widget of Figure 4.10(b) only appears with $a > 0$.



(a) Widget $\mathcal{T}_{x>a}$ (b) Widget $\mathcal{T}_{x\geqslant a}$ (assume $a > 0$)

Figure 4.10. *Widgets for $\mathcal{T}_{x>a}$ and $\mathcal{T}_{x\geqslant a}$*

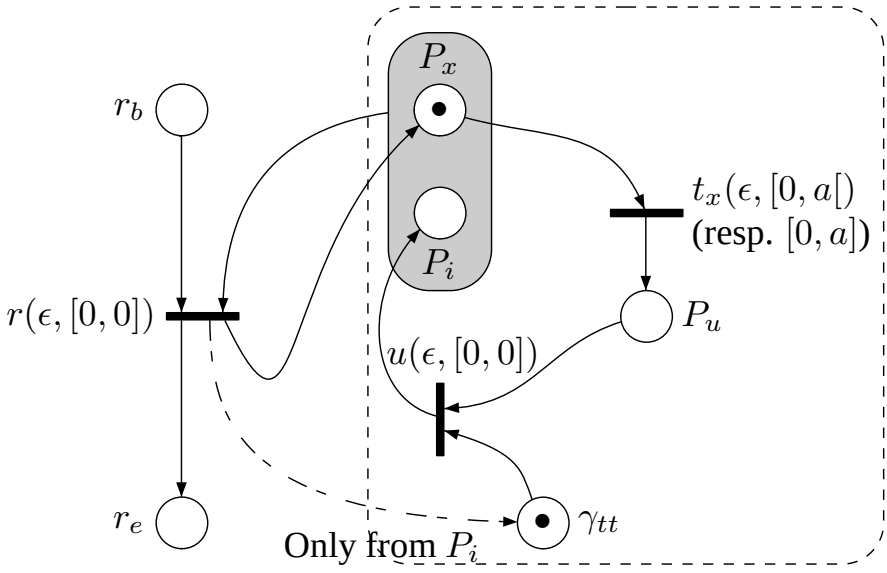


Figure 4.11. *Widget $\mathcal{T}_{x<a}$ (resp., $\mathcal{T}_{x\leqslant a}$)*

Each of these TPNs basically consists of a "constraint" subpart (in the grey boxes for Figure 4.10 and in the dashed box for Figure 4.11) that models the truth value of the atomic constraint, and another "reset" subpart that will be used to update the truth value of the constraint when the clock x is reset. The "constraint" subpart features the place γ_{tt}: the intended meaning is that when a token is available in this place, the corresponding atomic constraint φ is true.

When a clock x is reset, all the grey blocks modeling an x-constraint must be set to their *initial* marking which has one token in P_x for Figure 4.10 and one token in P_x and γ_{tt} for Figure 4.11. Our strategy to reset a block modeling a constraint is to put a token in the r_b place (r_b stands for "reset begin"). Time cannot elapse from there on (strong semantics for TPNs), as there will be a token in one of the places of the grey block and thus transition r will be enabled.

We first prove three useful lemmas, the first one providing a structural invariant for the grey boxes of the widgets:

LEMMA 4.2.– *For each widget of Figure 4.10, each reachable configuration (M, ν) (from the initial marking) has exactly one token in one of the places of the grey box.*

LEMMA 4.3.– *For the widgets of Figure 4.11, each reachable configuration (M, ν) (from the initial marking) satisfies either i) $M(P_x) = 1$, $M(\gamma_{tt}) = 1$ and $M(P_i) = 0$ or ii) $M(P_u) = 1$ and $M(\gamma_{tt}) = 1$ or iii) $M(P_x) = 0$, $M(\gamma_{tt}) = 0$ and $M(P_i) = 1$.*

Proof. The proof is easy for the widgets of Figure 4.10. For the widgets of Figure 4.11, just notice that as soon as t_x is fired, the output transition u is enabled (there must be a token in γ_{tt} as it can only be removed by the firing of u). Later on, either the token remains in P_i forever or, if the copy of r from P_i is fired, a token is put in γ_{tt} and P_x. ∎

From Lemmas 4.2 and 4.3 we obtain the following:

LEMMA 4.4.– *If there is a token in r_b, exactly one (instance of a) copy of r is fireable and due to the time constraint $[0, 0]$, time cannot progress until it is fired.*

4.5.2. *Resetting clocks*

Assume that $R \subseteq X$ is a non-empty subset of clocks. Let $D(R)$ be the set of atomic constraints that are in the scope of R (the clock of the constraint is in R). We write $D(R) = \{\varphi_1, \varphi_1, \cdots, \varphi_n\}$. To update all the widgets $\mathcal{N}_{\varphi_i}$, we connect the places r_b and r_e of each widget $\mathcal{N}_\varphi$ as described on Figure 4.12. The picture inside the dashed box denotes the widget $\mathcal{N}_{\mathit{Reset}(R)}$. We denote by $r_b^1(R)$ the first place of this widget and by $r_b^n(R)$ the last one. To update the (truth value of the) widgets $\mathcal{N}_{\varphi_i}$ it then suffices to put a token in $r_b^1(R)$. In zero duration it will go to $r_b^n(R)$ and have the effect of updating each widget $\mathcal{N}_{\varphi_i}$ on its way.

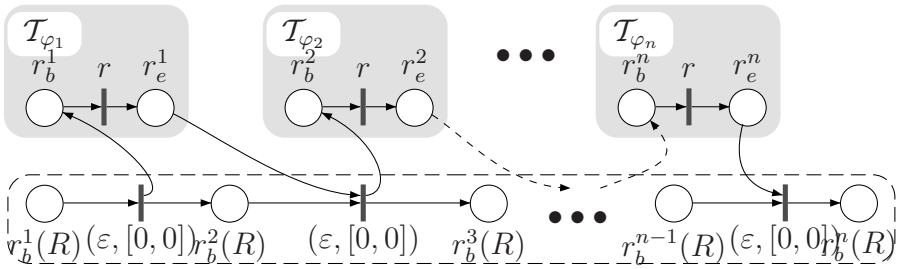


Figure 4.12. *Widget $\mathcal{N}_{\textit{Reset}(R)}$ to reset the widgets $\mathcal{N}_{\varphi_i}$, $1 \leqslant i \leqslant n$*

4.5.3. *The complete construction*

First we create fresh places P_ℓ for each $\ell \in L$ and another place *Firing* just for convenience: it will allow us to define a simulation relation more succintly. Then we build the widgets $\mathcal{N}_\varphi$, for each atomic constraint φ that appears in $\mathcal{A}$. Finally, for each $R \subseteq X$ such that there is an edge $e = (\ell, \gamma, a, R, \ell') \in E$, we build a reset widget $\mathcal{N}_{\textit{Reset}(R)}$.

Then, for each edge $(\ell, \gamma, a, R, \ell') \in E$ with $\gamma = \wedge_{i=1,n}\varphi_i$ and $n \geqslant 0$ we proceed as follows:

1) we assume $\gamma = \wedge_{i=1,n}\varphi_i$ and $n \geqslant 0$;

2) we create a transition $f(a, [0, \infty[)$ and, if $n \geqslant 1$, another one $r(\varepsilon, [0, 0])$;

3) we connect them to the places of the widgets $\mathcal{N}_{\varphi_i}$ and $\mathcal{N}_{\textit{Reset}(R)}$ as described on Figure 4.13. In the case where $\gamma = \mathsf{tt}$ (or $n = 0$) there is only one input place to $f(a, [0, \infty[)$ which is P_ℓ. In the case where $R = \emptyset$ there is no transition $r(\varepsilon, [0, 0])$ and the output place of $f(a, [0, \infty[)$ is $P_{\ell'}$ instead of *Firing*.

The place *Firing* is just added for convenience: it has a token only during the reset phase of the TPN $\mathcal{N}_e$ and thus means "we are firing transitions in the reset widget $\mathcal{N}_{\textit{Reset}(R)}$".

To complete the construction we just need to put a token in the place P_{ℓ_0} if ℓ_0 is the initial location of the automaton and set each widget $\mathcal{T}_\varphi$ to its initial marking, for each atomic constraint φ that appears in $\mathcal{A}$, and this defines the initial marking M_0. The set of final markings is defined by the set of markings M such that $M(P_\ell) = 1$ for $\ell \in F$ and the set of repeated markings by the set of markings M such that $M(P_\ell) = 1$ for $\ell \in R$. We denote by $\Delta(\mathcal{A})$ the TPN obtained as described previously. Notice that by construction $\Delta(\mathcal{A})$ is 1-safe and, moreover, in each reachable marking M of $\Delta(\mathcal{A})$ $\big(\sum_{\ell \in L} M(P_\ell)\big) + M(\textit{Firing}) = 1$.

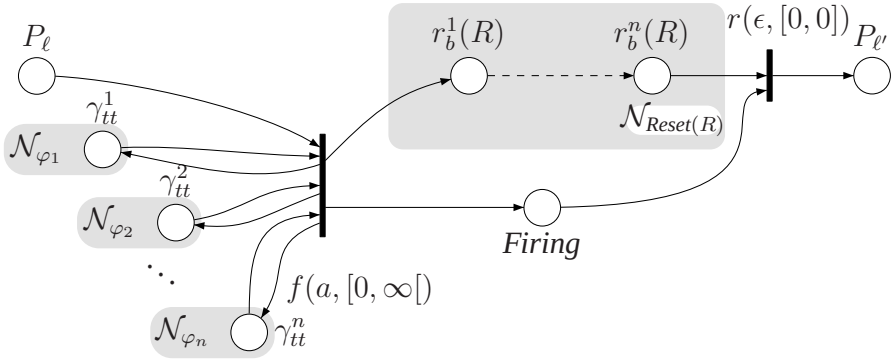


Figure 4.13. *Widget $\mathcal{N}_e$ of an edge $e = (\ell, \gamma, a, R, \ell')$*

4.5.4. $\Delta(\mathcal{A})$ *and* $\mathcal{A}$ *accept the same timed language*

We now prove the following theorem:

THEOREM 4.7.– *If $\Delta(\mathcal{A})$ is defined as above, then $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\Delta(\mathcal{A}))$.*

Proof. The proof works as follows: we first show that $\Delta(\mathcal{A})$ weakly simulates $\mathcal{A}$, which implies $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\Delta(\mathcal{A}))$. Then we show that we can define a TA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ and $\mathcal{A}'$ weakly simulates $\Delta(\mathcal{A})$, which entails $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$. It is sufficient to give the proof for the case where $\mathcal{A}$ has no ε transitions. In case $\mathcal{A}$ has ε transitions we rename them with a fresh letter $\mu \notin \Sigma_\varepsilon$ and obtain an automaton $\mathcal{A}_\mu$ with no ε transitions. We apply our construction to $\mathcal{A}_\mu$ and obtain a TPN in which we replace every label μ by ε.

Recall that $\mathcal{A} = (L, \ell_0, X, \Sigma_\varepsilon, E, Inv, F, R)$ and $\Delta(\mathcal{A}) = (P, T, \Sigma_\varepsilon, {}^\bullet(.), (.)^\bullet, M_0, \Lambda, I, F_\Delta, R_\Delta)$ and write $X = \{x_1, \cdots, x_k\}$, $P = \{p_1, \cdots, p_m\}$ and $T = \{t_1, \cdots, t_n\}$. We assume that the set of atomic constraints of $\mathcal{A}$ is $\mathcal{C}_\mathcal{A}$. The place γ_{tt} of a widget $\mathcal{N}_{x \bowtie c}$ (for $x \bowtie c$ an atomic constraint of $\mathcal{A}$) is written $\gamma_{tt}^{x \bowtie c}$.

• Proof that $\Delta(\mathcal{A})$ simulates $\mathcal{A}$.

We define the relation $\preceq \subseteq (L \times \mathbb{R}_{\geqslant 0}^n) \times (\mathbb{N}^p \times \mathbb{R}_{\geqslant 0}^m)$ by:

$$(\ell, v) \preceq (M, \nu) \text{ if and only if} \qquad [4.1]$$

$$\begin{cases} (1)\, M(P_\ell) = 1 \vee M(\textit{Firing}) = 1 \\ (2)\, \text{for each } \varphi = x \bowtie c, \bowtie \in \{<, \leqslant\},\ M(P_u) = 0 \\ (3)\, \text{for each } \varphi \in \mathcal{C}_\mathcal{A},\ v \in [\![\varphi]\!] \iff M(\gamma_{tt}^\varphi) = 1 \end{cases} \qquad [4.2]$$

Now we prove that $\preceq$ is a weak simulation relation of $\mathcal{A}$ by $\Delta(\mathcal{A})$, and this by checking the 4 conditions of Definition 4.4:

1) final and repeated states: by definition of $\Delta(\mathcal{A})$ and Lemmas 4.2 and 4.3 and the definition of $\preceq$;

2) initial states: it is clear that $(l_0, \mathbf{0}) \preceq (M_0, \mathbf{0})$;

3) continuous transitions: let $(\ell, v) \xrightarrow{d} (\ell, v+d)$. Take (M, ν) such that $(\ell, v) \preceq (M, \nu)$. As the widgets $\mathcal{N}_{\varphi_i}$ are non-blocking, time d can elapse from (M, ν) and there is a run $(M, \nu) \xrightarrow{\rho} (M', \nu')$ with *Duration*(*trace*$(\rho)) = d$ and *Untimed*(*trace*$(\rho)) = \varepsilon$. We can choose ρ without any transitions $f(a, [0, \infty[)$ so that a token remains in P_ℓ and $M'(P_\ell) = 1$. Thus in order to prove $(\ell, v+t) \preceq (M', \nu')$ it remains to prove items (2) and (3) of equation [4.1].

Let $\varphi = x \bowtie c$ with $\bowtie \in \{<, \leqslant\}$:

- if $\varphi(v) = \mathsf{tt}$ and $\varphi(v+d) = \mathsf{ff}$, then there is some $d' \leqslant d$ such that transition t_x of widget $\mathcal{N}_\varphi$ is enabled and it must be fired before φ becomes false. Thus, t_x is fired at d' (which is possible as there is no token in P_u and thus the token is in P_x) and subsequently u in the same widget, thus transfering the tokens from P_x, γ^φ_{tt} to P_i,

- if $\varphi(v) = \mathsf{tt}$ and $\varphi(v+d) = \mathsf{tt}$, it is possible to do nothing in widget $\mathcal{N}_\varphi$ and let the token in P_x and γ^φ_{tt},

- if $\varphi(v) = \mathsf{ff}$ then $\varphi(v+d) = \mathsf{ff}$, then there must be a token in P_i and we let time elapse without firing any transition.

Let $\varphi = x \bowtie c$ with $\bowtie \in \{>, \geqslant\}$:

- if $\varphi(v) = \mathsf{tt}$ then $\varphi(v+d) = \mathsf{tt}$ and $M(\gamma^\varphi_{tt}) = 1$. We just let time elapse in $\mathcal{N}_\varphi$,

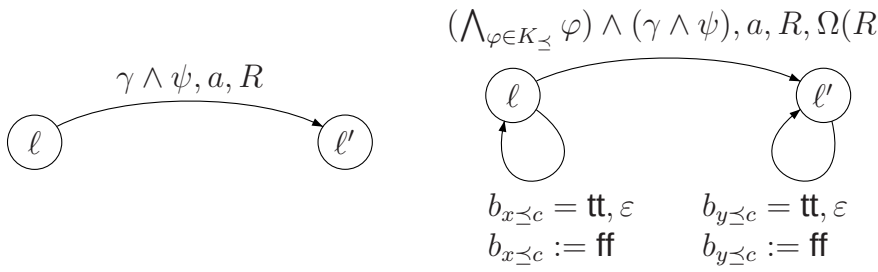
- if $\varphi(v) = \mathsf{ff}$ and $\varphi(v+d) = \mathsf{tt}$, there are $t' \leqslant t$ such that transitions t_x which must be fired (and t' can be fired at $d' + \xi$ with $\xi > 0$ for $\mathcal{N}_{x>c}$). We fire those transitions at d' and let $d - d'$ elapse,

- if $\varphi(v) = \mathsf{ff}$ and $\varphi(v+d) = \mathsf{ff}$ we also let time elapse and leave a token in P_x.

This way for each constraint $\varphi = x \bowtie c$, there is a run $\rho_\varphi = (M, \nu) \xrightarrow{d}_\varepsilon (M_\varphi, \nu_\varphi)$ such that (M_φ, ν_φ) satisfies requirements (2) and (3) of equation [4.1]. Taken separately we have for each constraint $(\ell, v) \preceq (M_\varphi, \nu_\varphi)$. It is not difficult[5] to build a run ρ with an interleaving of the previous runs ρ_φ such that $\rho = (M, \nu) \xrightarrow{t}_\varepsilon (M', \nu')$ and (M', ν') satisfies requirements (2) and (3) of equation [4.1] for each constraint φ and thus $(\ell, v) \preceq (M', \nu')$;

4) discrete transitions: let $(\ell, v) \xrightarrow{a} (\ell', v')$ and $(\ell, v) \preceq (M, \nu)$. Then there is an edge $e = (\ell, \gamma, a, R, \ell') \in E$ such that $\gamma = \wedge_{i=1,n} \varphi_i$, $n \geqslant 0$ and φ_i is an atomic constraint. According to Definition 4.9, $v \in [\![\varphi_i]\!]$ for $1 \leqslant i \leqslant n$. This implies $M(\gamma^{\varphi_i}_{tt}) = 1$ (definition of $\preceq$). Thus, the transition $f(a, [0, \infty[)$ is fireable in the widget $\mathcal{N}_e$ leading

5. Just find an ordering for all the date d' at which a transition must be fired and fire those transitions in this order with time elapsing between them.



(a) Edge $(\ell, \gamma \wedge \psi, a, R, \ell')$ in $\mathcal{A}$ (b) Extended edge in $\mathcal{A}'$.

Figure 4.14. *From $\mathcal{A}$ to $\mathcal{A}'$*

to (M', ν'). From there on we do not change the marking of widgets $\mathcal{N}_{\varphi_i}$ for the constraints φ_i that do not need to be reset (the clock of φ_i is not in R). We also use the widget $\mathcal{T}_e$ to reset the constraints φ_i with a clock in R and finally put a token in $P_{\ell'}$. The new state (M'', ν'') obtained this way satisfies $(\ell', v') \preceq (M'', \nu'')$.

This completes the proof that $\Delta(\mathcal{A})$ simulates $\mathcal{A}$ and thus $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\Delta(\mathcal{A}))$.

• Proof of $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$.

To prove this, we cannot easily exhibit a simulation of $\Delta(\mathcal{A})$ by $\mathcal{A}$. Indeed, $\Delta(\mathcal{A})$, because of the widgets $\mathcal{N}_{x \bowtie c}$ with $\bowtie \in \{<, \leqslant\}$, has to make a decision at some point to fire transition t_x and immedialty after u, i.e. it is as if it decides that $x \bowtie c$ is now false and the transitions with this guard cannot be fired anymore (until they are reset). To use the simulation framework, we build first a TA $\mathcal{A}'$ that accepts the same language as $\mathcal{A}$ but has the capability to sometimes (non-deterministically) decide it will not use a transition with a guard $x \bowtie c$ until it is reset. It is then possible to build a simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}'$.

We denote $\preceq$ for either $\{<, \leqslant\}$ and $\succeq$ for $\{>, \geqslant\}$. Let $K_\preceq$ be the set of contraints $x \preceq c$ in $\mathcal{A}$. For each $x \preceq c \in K_\preceq$ we introduce a Boolean variable $b_{x \preceq c}$. Each $b_{x \preceq c}$ is initially true.

We start with $\mathcal{A}' = \mathcal{A}$. The construction of the new features of $\mathcal{A}'$ is depicted on Figure 4.14. Let $(\ell, \gamma \wedge \psi, a, R, \ell')$ be an edge of $\mathcal{A}'$ with $\gamma = \wedge_{x \preceq c \in K_\preceq} x \preceq c$ and $\psi = \wedge_{x \succeq c \in K_\succeq} x \succeq c$. For such an edge we strengthen[6] the guard $\gamma \wedge \psi$ to obtain γ' as

6. We need an extended type of TA with Boolean variables; this does not add any expressive power to the model.

follows: $\gamma' = \gamma \wedge \psi \wedge \bigwedge_{x \preceq c \in K_\preceq} b_{x \preceq c}$. This way the transition $(\ell, \gamma \wedge \psi, a, R, \ell')$ can be fired in $\mathcal{A}'$ only if the corresponding guard in $\mathcal{A}$ and the conjunction of the $b_{x \preceq c}$ is true as well. We also reset to true all the variables $b_{x \preceq c}$ such that $x \in R$ on a transition $(\ell, \gamma \wedge \psi, a, R, \ell')$ and $\Omega(R)$ corresponds to the reset of all $b_{x \preceq c}$ such that $x \in R$, $\Omega(R) = \wedge_{x \in R} b_{x \preceq c} := \mathsf{tt}$.

Now let ℓ be the location of $\mathcal{A}'$. For each variable $b_{x \preceq c}$ we add a loop edge $(\ell, b_{x \preceq c} = \mathsf{tt}, \varepsilon, b_{x \preceq c} := \mathsf{ff}, \ell)$ in $\mathcal{A}'$, i.e. the automaton $\mathcal{A}'$ can decide non-deterministically[7] to set $b_{x \preceq c}$ to false if it is true (see Figure 4.14). There are as many loops on each location as the number of variables $b_{x \preceq c}$. The new non-deterministic TA $\mathcal{A}'$ accepts exactly the same language as $\mathcal{A}$, i.e. $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

We can now build a simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}'$. We denote by (ℓ, v, b) a configuration of $\mathcal{A}'$ with b the vector of b_φ variables. We define the relation $\preceq \subseteq (\mathbb{N}^p \times \mathbb{R}^m_{\geqslant 0}) \times (L \times \mathbb{R}^n_{\geqslant 0} \times \mathbb{B}^k)$ by:

$$(M, \nu) \preceq (\ell, v, b) \textit{ if and only if} \qquad [4.3]$$

$$\begin{cases} (1)\, M(P_\ell) = 1 \vee M(\textit{Firing}) = 1 \\ (2)\, \forall \varphi = x > c \in K_>,\ v \in [\![\varphi]\!] \iff M(\gamma^\varphi_{tt}) = 1 \\ (3)\, \forall \varphi = x \geqslant c \in K_\geqslant,\ v \in [\![\varphi]\!] \iff M(\gamma^\varphi_{tt}) = 1 \\ \qquad\qquad \vee\ (M(P^\varphi_x) = 1 \wedge \nu(t^\varphi_x) = c) \\ (4) \forall \varphi \in K_\preceq, M(P^\varphi_i) = 1 \iff (b_\varphi = \mathsf{ff} \vee v \notin [\![\varphi]\!]) \end{cases} \qquad [4.4]$$

Now we prove that $\preceq$ is a weak simulation relation of $\Delta(\mathcal{A})$ by $\mathcal{A}$:

– the property on final and repeated states is satisfied by the definition of $\mathcal{A}'$;

– for the initial configuration, it is clear that $(M_0, \mathbf{0}) \preceq (l_0, \mathbf{0}, b_0)$ (in b_0 all the variables b are true);

– continuous time transitions: let $(M, \nu) \xrightarrow{d} (M', \nu')$ with $d \geqslant 0$. Let $(M, \nu) \preceq (\ell, v, b)$. As there are no invariants in $\mathcal{A}'$, time d can elapse from (ℓ, v, b). If no ε transition fires in the TPN, then all the truth values of the constraints stay unchanged. Thus $(\ell, v, b) \xrightarrow{d} (\ell, v + d, b)$ and in $\mathcal{A}'$ such that $(M', \nu') \preceq (\ell, v + d, b)$;

– discrete transitions: let $(M, \nu) \xrightarrow{a} (M', \nu')$. We distinguish the cases $a = \varepsilon$ and $a \in \Sigma$. If $a \in \Sigma$, then we must fire a transition $f(a, [0, \infty[)$ of some widget $\mathcal{N}_e$ for $e = (\ell, \gamma, a, R, \ell')$. After firing f we end up in *Firing* and have left the input places γ_{tt} unchanged. According to equation [4.3] and the definition of $\mathcal{A}'$, we can fire a

7. This means that we add ε transitions to $\mathcal{A}'$; nevertheless the restriction we made at the beginning that $\mathcal{A}$ has no ε transitions is useful when proving that $\Delta(\mathcal{A})$ simulates $\mathcal{A}$ and is not required to prove that $\mathcal{A}'$ weakly simulates $\Delta(\mathcal{A})$.

matching transition in $\mathcal{A}'$ leading to a state (ℓ', v', b') and $(M', \nu') \preceq (\ell', v', b')$. If $a = \varepsilon$, then we are either updating some widgets $\mathcal{N}_\varphi$ or doing a reset. Assume that we fire a ε transition that is not a reset transition ($M(P_\ell) = 1$). We split the cases according to the different types of widgets:

- update of a widget $\mathcal{N}_{x>c}$: either t_x or t' is fired. If t_x is fired then the time elapsed since the x was last reset is equal to a. Thus, $M(\gamma_{tt}) = 0$ and $v(x) \leqslant c$ and $v \notin [\![x > c]\!]$. This implies $(M', \nu') \preceq (\ell, v)$. If t' is fired on the contrary, $v'(x) > c$ but again $(M', \nu') \preceq (\ell, v, b)$;

- update of a widget $\mathcal{N}_{x \geqslant c}$: the same reasoning as before can be used and leads to $(M', \nu') \preceq (\ell, v, b)$;

- update of a widget $\mathcal{N}_{x<c}$: in this case either t_x or u is fired. Assume t_x is fired. Thus, $M'(P_i) = 0$. The time elapsed since x was last reset is strictly less than c and $v \in [\![\varphi]\!]$. b_φ is true in (ℓ, v, b) as $M(P_i) = 0$. Thus, $(M', \nu') \preceq (\ell, v, b)$. Now assume that u is fired. Again $M(P_i) = 0$ and thus $v(x) < c$ and b_φ are true. This time $M'(P_i) = 1$. In the automaton $\mathcal{A}'$ we fire the transition setting b_φ to false and we end up in a state (ℓ, v, b') such that $(M', \nu') \preceq (\ell, v, b')$. The same reasoning applies for $\mathcal{N}_{x \geqslant c}$.

For the reset transitions and the last transition putting a token in $P_{\ell'}$ we can proceed similarly.

This completes the proof that $\mathcal{A}'$ simulates $\Delta(\mathcal{A})$ and thus $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A}')$ and $\mathcal{L}(\Delta(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$.

We can thus conclude that $\mathcal{L}(\Delta(\mathcal{A})) = \mathcal{L}(\mathcal{A})$, which concludes the proof of Theorem 4.7. ∎

4.5.5. *Consequences of the previous results*

Let k-$\mathcal{TPN}$ be the set of k-bounded TPNs (note that boundedness is not decidable for TPNs). Let B-$\mathcal{TPN} = \{T \mid \exists k \geqslant 0 \mid T \in k\text{-}\mathcal{TPN}\}$, i.e. the set of bounded TPNs. From the previous proposition we can state the following corollaries:

THEOREM 4.8.– *The classes B-$\mathcal{TPN}$ and $\mathcal{TA}$ are equally expressive with respect to timed language acceptance, i.e. B-$\mathcal{TPN} =_{\mathcal{L}} \mathcal{TA}$.*

Proof. From Theorem 4.1, we know that B-$\mathcal{TPN} \leqslant_{\mathcal{L}} \mathcal{TA}$. Theorem 4.7 proves that $\mathcal{TA} \leqslant_{\mathcal{L}} \mathcal{TPN}$ and hence B-$\mathcal{TPN} =_{\mathcal{L}} \mathcal{TA}$. ∎

COROLLARY 4.1.– k-$\mathcal{TPN} =_{\mathcal{L}} 1$-$\mathcal{TPN}$.

Proof. Let $T \in k$-$\mathcal{TPN}$. We use Theorem 4.1 and thus there is a TA A_T such that $\mathcal{L}(T) = \mathcal{L}(A_T)$. From A_T we use Theorem 4.7 and obtain $\Delta(A_T)$ which is a 1-safe TPN. ∎

4.6. Bisimulation of TA by TPNs

We now focus on the expressiveness of the models with respect to weak time bisimilarity. In what follows, we often abbreviate weak timed bisimilarity by bisimilarity.

First, we recall two related results:

– There are unbounded TPNs which do not admit a bisimilar TA. This is a direct consequence of the following observation: the untimed language of a TA is regular which is not necessarily the case for PNs (and thus for TPNs).

– For any bounded TPN, there is a TA which is bisimilar to it (see Theorem 4.1 from [CAS 04]).

This last result was proved by the construction of a synchronized product of automata enlarged with a vector of bounded integers (a model equivalent to standard automata). The proposed construction is structural and linear with respect to to the size of the PN. It has the additional advantage that the available tools exploit the product in order to reduce the complexity of verification. Here we are mainly concerned with expressivity. So we can also give a straightforward construction based on the reachability space:

– With each transition t, we associate a clock x_t.

– With each reachable marking M, we associate a location ℓ_M. The invariant of ℓ_M is given by $\bigwedge_{t \in En(M)} x_t \in I(t)^{\downarrow}$, (recall that $I(t)$ is the interval associated with t).

– With each firing $M \xrightarrow{t} M'$ (in the untimed PN), we associate an edge $e = (\ell_M, \gamma, t, R, \ell_{M'})$ with $\gamma = x_t \in I(t)$ and R defined according to the chosen semantic. For instance, if PA semantics is chosen, then $R = \{x_{t'} \mid t' \notin En(M) \wedge t' \in En(M')\}$.

In this section, we consider the TPNs originally defined by Merlin (i.e. without strict constraints) and labeled-free TA (i.e. where two different edges have different labels and no label is ε) and we develop the main result of the chapter: a characterization of the subclass of TA which admits a bisimilar TPN. From this characterization, we will deduce that given a TA, the problem of deciding whether there is a TPN bisimilar to it, is **PSPACE**-complete. Furthermore, we will provide two effective constructions for such a TPN: the first one with rational constants has a size linear with respect to the TA, while the other, which uses only integer constants, has an exponential size.

4.6.1. *Regions of a timed automaton*

Since our proofs are based on the regions of a timed automaton, we detail their definition. Recall that a *region* is a pair composed by a location and an *elementary*

time zone of the grid defined by the clocks and the granularity g. In what follows, the topology of the regions is implicitly derived from that of its associated zone. We now formally define the particular case of regions for a maximal constant $K = \infty$. Obviously it may lead to an infinite region automaton but will be a helpful tool for proving our characterization. Note also that the following definition is equivalent to the original one but is more appropriate for our theoretical developments.

DEFINITION 4.12.– *[Regions of an automaton with respect to the g-grid and constant $K = \infty$] A time-closed region r is given by:*

– ℓ_r the location of r;

– $min_r \in \mathbb{N}_g^X$ the minimal vector of the topological closure of r;

– the number $size_r$ of different fractional parts of clock values in the grid $\mathbb{N}_g^X$, with $1 \leqslant size_r \leqslant |X|$ and the onto mapping $ord_r : X \mapsto \{1, \ldots, size_r\}$ which gives the relative positions of these fractional parts.

The region is then $r = \{(\ell_r, min_r + \delta) \,|\, \delta \in \mathbb{R}_{\geqslant 0}^X \ \wedge \ \forall x, y \in X[ord_r(x) = 1 \Leftrightarrow \delta(x) = 0] \wedge \delta(x) < 1/g \ \wedge \ [ord_r(x) < ord_r(y) \Leftrightarrow \delta(x) < \delta(y)]\}$.

A time-open region r is defined with the same attributes as the time-closed region by: $r = \{(\ell_r, min_r + \delta + d) \,|\, d \in \mathbb{R}_{>0} \ \wedge \ \forall x \in X,\, \delta(x) + d < 1/g\}$.

The set $[X]_r$ is the set of equivalence classes of clocks with respect to their fractional parts, i.e. x and y are equivalent iff $ord_r(x) = ord_r(y)$.

This definition needs to be slightly modified when dealing with a constant $K < \infty$, by introducing a subset of *relevant* clocks, for which the value is less than K (recall that $K > m$ where m is the maximal constant in the constraints of the timed automaton).

DEFINITION 4.13.– *[Regions of an automaton with respect to the g-grid and finite constant K] A time-closed region r is given by:*

– ℓ_r the location of r;

– $min_r \in \mathbb{N}_g^X$ with $\forall x,\ min_r(x) \leqslant K$ the minimal vector of the topological closure of r;

– $ActX_r = \{x \in X \,|\, min_r(x) < K\}$ the subset of relevant clocks;

– the number $size_r$ of different fractional parts for the values of relevant clocks in the $\mathbb{N}_g^{ActX_r}$ grid, with $1 \leqslant size_r \leqslant Max(|ActX_r|, 1)$ and the onto mapping $ord_r : X \mapsto \{1, \ldots, size_r\}$ giving the ordering of the fractional parts. By convention, $\forall x \in X \setminus ActX_r, ord_r(x) = 1$.

Then $r = \{(\ell_r, min_r + \delta) \,|\, \delta \in \mathbb{R}_{\geqslant 0}^X \ \wedge \ \forall x, y \in ActX_r[ord_r(x) = 1 \Leftrightarrow \delta(x) = 0] \ \wedge \ \delta(x) < 1/g \ \wedge \ [ord_r(x) < ord_r(y) \Leftrightarrow \delta(x) < \delta(y)]\}$.

A time-open description of a region r is given by the same attributes (and conditions) as a time-closed region with: $r = \{(\ell_r, min_r + \delta + d) \,|\, d \in \mathbb{R}_{>0} \wedge \forall x \in ActX_r,\, \delta(x) + d < 1/g\}$.

Note that letting time elapse leads to an alternation of time-open regions (where time can elapse) and time-closed ones (where no time can elapse). We also note that $min_r \notin r$ except if there is a single class of clocks relative to r (for instance if r is a singleton). More generally, irrespective of the grid and the maximal constant, we note by $\overline{r}$ the topological closure of r: it is a finite union of regions and from the definition, min_r is the minimum vector of $\overline{r}$.

Reachability. Recall that a region is *reachable* if it belongs to the region automaton. However it does not mean that all the configurations of the region are reachable. Nevertheless, by induction on the reachability relation inside the region automaton it can be shown that every configuration is "quasi-reachable" in the following sense. For each reachable region r, there is a region $reach(r)$ with respect to the 1-grid and the constant ∞ such that:

– $reach(r) \subset r$;

– each configuration of $reach(r)$ is reachable;

– if $reach(r)$ is a time-open region then r admits a time-open description or else r admits a time-closed description.

Note that consequently $\forall x \in ActX_r, min_{reach(r)}(x) = min_r(x)$ and $\forall x \in X \setminus ActX_r, min_{reach(r)}(x) \geqslant K$ and that ord_r restricted to $ActX_r$ is identical to $ord_{reach(r)}$.

Let us define $\mathcal{R}$ by $(l, v)\mathcal{R}(l, v')$ iff $\forall x \in X, v'(x) = v(x) \vee (v(x) \geqslant K \wedge v'(x) \geqslant K)$. Then $\mathcal{R}$ is a strong time bisimulation relation. From the previous observations, we note that each configuration of a reachable region is strongly time bisimilar to a reachable configuration of this region. Thus speaking about reachability of regions is a slight abuse of notations.

4.6.2. *From bisimulation to uniform bisimulation*

As a first step towards our characterization, we prove that when a TPN and a TA are bisimilar, the condition can in fact be strengthened in what we call uniform bisimulation.

We first prove a lemma which is also a strengthened version of Lemma 4.1. It points out the effect of time granularity on the behavior of TPN when strict constraints are excluded.

LEMMA 4.5.– *Let (M,ν) and $(M,\nu+\delta)$ be two admissible configurations of a g-TPN with $\nu,\delta \in \mathbb{R}_{\geqslant 0}^{En(M)}$. Let w be an instantaneous firing sequence, then:*

(a) $(M,\nu) \xrightarrow{w}\Rightarrow (M,\nu+\delta) \xrightarrow{w}$

(b) If $\nu \in \mathbb{N}_g{}^{En(M)}$ *and* $\delta \in [0,1/g[^{En(M)}$ *then* $(M,\nu+\delta) \xrightarrow{w}\Rightarrow (M,\nu) \xrightarrow{w}$

Proof. There are two kinds of transitions firing in w: those corresponding to a firing of a transition (for example, t) still enabled from the beginning of the firing sequence and those corresponding to a newly enabled transition (for example, t').

Proof of (a) Since t is fireable from (M,ν), $\nu(t) \in I(t) \subset I(t)^{\uparrow}$, so $\nu(t)+\delta(t) \geqslant \nu(t)$ also belongs to $I(t)^{\uparrow}$. Since $t \in En(M)$ and $(M,\nu+\delta)$ are reachable, $\nu(t)+\delta(t) \in I(t)^{\downarrow}$. Thus $\nu(t)+\delta(t) \in I(t)$ and t is also fireable from $(M,\nu+\delta)$. Since t' is newly enabled, $0 \in I(t')$ and t' is also fireable when it occurs starting from $(M,\nu+\delta)$.

Proof of (b) The case of newly enabled transitions in w is handled as before. Now let t be fireable in $(M,\nu+\delta)$. Since $t \in En(M)$ and (M,ν) is reachable, $\nu(t) \in I(t)^{\downarrow}$. Since $\nu(t)+\delta(t) \in I(t)^{\uparrow}$ (denoting by $eft(t)$ the minimum of $I(t)^{\uparrow}$), we have $eft(t) \leqslant \nu(t)+\delta(t)$ but $eft(t)$ belongs to the g-grid, thus $eft(t) \leqslant \nu(t) \Leftrightarrow \nu(t) \in I(t)^{\uparrow}$. So t is fireable from (M,ν). ∎

LEMMA 4.6.– *[From bisimulation to uniform bisimulation] Consider a timed automaton $\mathcal{A}$ bisimilar to some g-TPN $\mathcal{N}$ via some relation $\mathcal{R}$. The semantics considered for $\mathcal{N}$ is PA as it gives the maximal expressivity. We consider the region automaton of $\mathcal{A}$ with respect to the grid $\mathbb{N}_g^X$ and the constant $K=\infty$. Then:*

– if a region r belongs to $R(\mathcal{A})$ then $\overline{r}$ also belongs to $R(\mathcal{A})$;

– with each reachable region r is associated a configuration of the net (M_r,ν_r) with $\nu_r \in \mathbb{N}_g^{En(M_r)}$ and a mapping $\phi_r : En(M_r) \to [X]_r$ which fulfill:

- if r is time-closed, then $\forall(\ell_r, min_r+\delta) \in \overline{r}$, $(\ell_r, min_r+\delta)\mathcal{R}(M_r,\nu_r+proj_r(\delta))$ where $proj_r(\delta)(t)=\delta(\phi_r(t))$;

- if r is time-open, then $\forall(\ell_r, min_r+\delta+d) \in \overline{r}$, $(\ell_r, min_r+\delta+d)\mathcal{R}(M_r,\nu_r+proj_r(\delta)+d)$.

Proof. We prove this uniform version of the bisimulation by induction on the reachability relation between regions. First note that the choice of a particular clock x in the class $\phi_r(t)$ is irrelevant when considering the value $\delta(x)$. Thus, the definition of $proj_r$ is sound.

We prove our assertion by induction on the transition relation in the region automaton. The basic case is straightforward with $\{(l_0,\mathbf{0})\}$ and $\{(M_0,\mathbf{0})\}$. For the induction

part, we consider 4 cases, according to the incoming or target region and to the nature of the step.

1. A time step from a time-closed region. Let r be a time-closed region which is not maximal and let us denote by $r' = succ(r)$ the immediate time successor of r. Let $(\ell_r, min_r + \delta_{\mathbf{0}})$ be some item of r. $(\ell_r, min_r + \delta_{\mathbf{0}}) \xrightarrow{d}$ for some $d > 0$. Thus (by induction hypothesis), in $\mathcal{N}$ there is a step sequence of $(M_r, \nu_r + proj_r(\delta_{\mathbf{0}})) \overset{d_0 t_1 \dots t_n d_n}{\longrightarrow}$ with all transitions labeled by ϵ and $\sum d_k = d$. Let d_k be the first non-zero elapsing of time. By application of Lemma 4.5(b), the firing sequence $t_1 \dots t_k$ is fireable from (M_r, ν_r).

Let us choose $(M_{r'}, \nu_{r'})$ as the configuration reached by this sequence. By application of Lemma 4.5(a), this firing sequence is also fireable from any $(M_r, \nu_r + proj_r(\delta))$ bisimilar to $(\ell_r, min_r + \delta) \in \overline{r}$ and it leads to $(M_{r'}, \nu_{r'} + proj_{r'}(\delta))$ (still bisimilar to $(\ell_r, min_r + \delta)$) where $\phi_{r'}$ (resp., $\nu_{r'}$) is equal to ϕ_r (resp., ν_r) for transitions always enabled during the firing sequence and $\phi_{r'}$ (resp. $\nu_{r'}$) is obtained by associating the class of index 1 (resp. by associating the value 0) with the newly enabled transitions. Since $(M_{r'}, \nu_{r'})$ let the time elapse and since $\mathcal{N}$ is a g-TPN, we note that $\forall t \in En(M_{r'}),\ \nu_{r'}(t) + 1/g \in I(t)^{\downarrow}$. Now let $(\ell_r, min_r + \delta + d) \in \overline{r'}$, and we have $\forall x \in X,\ \delta(x) + d \leqslant 1/g$. Thus, $\forall t \in En(M_{r'}), proj_{r'}(\delta(x)) + d \leqslant 1/g$, which implies $(M_{r'}, \nu_{r'} + proj_{r'}(\delta)) \xrightarrow{d} (M_{r'}, \nu_{r'} + proj_{r'}(\delta) + d)$; this last configuration is necessarily bisimilar to $(\ell_r, min_r + \delta + d)$.

2. A time step from a time-open region. Let r be a time-open region and let us denote $r' = succ(r)$. Let us define by X_r^{max} the class $[x]_r$ with maximal index. We note that $min_{r'} = min_r + \delta_{\mathbf{0}}$ where, if $x \in X_r^{max}$, then $\delta_{\mathbf{0}}(x) = 1/g$, otherwise $\delta_{\mathbf{0}}(x) = 0$. We choose $(M_{r'}, \nu_{r'}) = (M_r, \nu_r + proj_r(\delta_{\mathbf{0}}))$. Let $t \in En(M_r)$ and $x \in \phi_r(t)$ then $\phi_{r'}(t) = [x]_{r'}$ (letting time elapse does not split the classes). So $proj_r$ and $proj_{r'}$ are identical.

Now let $(l_{r'}, min_{r'} + \delta) \in \overline{r'}$. $(l_{r'}, min_{r'} + \delta) = (\ell_r, min_r + \delta_{\mathbf{0}} + \delta)$.

Now let $d = \delta(x)$ for x belonging to the class of index 1 in $[X_r]$. Then $(\ell_r, min_r + \delta_{\mathbf{0}} + \delta) = (\ell_r, min_r + \delta' + d)$ where, if $x \in X_r^{max}$, then $\delta'(x) = 1/g - d$, otherwise $\delta'(x) = \delta(x) - d$. $(\ell_r, min_r + \delta' + d)$ is bisimilar to $(M_r, \nu_r + proj_r(\delta') + d) = (M_r, \nu_r + proj_r(\delta' + d)) = (M_r, \nu_r + proj_r(\delta_{\mathbf{1}} + \delta)) = (M_r, \nu_r + proj_r(\delta_{\mathbf{1}}) + proj_r(\delta)) = (M_{r'}, \nu_{r'} + proj_{r'}(\delta)))$.

For this step, we have not used the characteristics of time Petri nets.

3. A discrete step into a time-closed region. We have two cases, namely a and b.

– **Case a.** We first consider the case where r is a time-closed region. Let $(\ell_r, \min_r + \delta_{\mathbf{0}})$ be some element of r. Suppose that $(\ell_r, \min_r + \delta_{\mathbf{0}}) \xrightarrow{e} (l', v' + \delta'_{\mathbf{0}})$ with $\forall x \in R(e), v'(x) = \delta'_{\mathbf{0}}(x) = 0, \forall x \notin R(e), v'(x) = \min_r(x) \wedge \delta'_{\mathbf{0}}(x) = \delta_{\mathbf{0}}(x)$.

Then in $\mathcal{N}$ there is a firing sequence $(M_r, \nu_r + proj_r(\delta_{\mathbf{0}})) \xrightarrow{w}$ labeled by e. Due to Lemma 4.5, this firing sequence is also fireable from any $(M_r, \nu_r + proj_r(\delta))$ bisimilar to $(\ell_r, min_r + \delta) \in \overline{r}$. According to bisimilarity, $(\ell_r, \min_r + \delta) \xrightarrow{e}$ for any $(\ell_r, min_r + \delta) \in \overline{r}$. Let r' be the region including $(\ell', v' + \delta'_{\mathbf{0}})$, then any configuration of $\overline{r'}$ is reachable by this discrete step. Note that $\ell_{r'} = l'$ and $min_{r'} = v'$. From $(M_r, \nu_r + proj_r(\delta))$, the sequence w leads to some (M', ν') bisimilar to $(\ell_{r'}, min_{r'} + \delta'))$. We now show how to define $M_{r'}$, $\nu_{r'}$ and $\phi_{r'}$. First $M_{r'} = M'$. Second, $\nu_{r'}(t) = \nu_r(t)$ for transitions t which are always enabled during the firing sequence and $\nu_{r'} = 0$ otherwise. At last, $\phi_{r'}$ is obtained from ϕ_r as follows. Let t be a newly enabled transition during the firing sequence, then $\phi_{r'}(t)$ is associated with the class of index 1. Let t be a transition always enabled during the firing sequence. There are three cases to consider for $\phi_{r'}(t)$: either there is a $x \in \phi_r(t)$ not reset, then $\phi_{r'}(t) = |x]_{r'}$ otherwise $\phi_{r'}(t)$ is the class of maximal index which precedes $\phi_r(t)$ and contains a clock not reset or else the class of index 1. The two last affectations are sound since it means that whatever the value of $\delta(t)$ fulfilling the order between classes, the firing sequence w leads to bisimilar configurations (as being bisimilar to the same configuration of the automaton).

– **Case b.** The case where r is a time-open region is handled in a similar way. Let $(\ell_r, \min_r + \delta_{\mathbf{0}} + d_0)$ be some element of r. Suppose that $(\ell_r, \min_r + \delta_{\mathbf{0}} + d_0) \xrightarrow{e}$ $(\ell', v' + \delta'_{\mathbf{0}})$ with $\forall x \in R(e), v'(x) = \delta'_{\mathbf{0}}(x) = 0, \forall x \notin R(e), v'(x) = \min_r(x) \wedge \delta'_{\mathbf{0}}(x) = \delta_{\mathbf{0}}(x) + d_0$. Then, in $\mathcal{N}$ there is a firing sequence $(M_r, \nu_r + proj_r(\delta_{\mathbf{0}}) + d_0) \xrightarrow{w}$ labeled by e. Due to Lemma 4.5, this firing sequence is also fireable from any $(M_r, \nu_r + proj_r(\delta) + d)$ bisimilar to $(\ell_r, min_r + \delta + d) \in \overline{r}$. According to bisimilarity, $(\ell_r, \min_r + \delta + d) \xrightarrow{e}$ for any $(\ell_r, min_r + \delta + d) \in \overline{r}$. Let r' be the region including $(l', v' + \delta'_{\mathbf{0}})$, then any configuration of $\overline{r'}$ is reachable by this discrete step. Note that $l_{r'} = l'$ and $min_{r'} = v'$. From $(M_r, \nu_r + proj_r(\delta) + d)$, the sequence w leads to some (M', ν') bisimilar to $(l_{r'}, min_{r'} + \delta'))$. We now show how to define $M_{r'}$, $\nu_{r'}$ and $\phi_{r'}$. First $M_{r'} = M'$. Second, $\nu_{r'}(t) = \nu_r(t)$ for transitions t always enabled during the firing sequence and $\nu_{r'} = 0$ otherwise. At last, $\phi_{r'}$ is obtained from ϕ_r as follows. Let t be a newly enabled transition during the firing sequence, then $\phi_{r'}(t)$ is associated with the class of index 1. There are three cases to consider for $\phi_{r'}(t)$: either there is a $x \in \phi_r(t)$ not reset, then $\phi_{r'}(t) = |x]_{r'}$ otherwise $\phi_{r'}(t)$ is the class of maximal index which preceedes $\phi_r(t)$ and contains a clock not reset or else the class of index 1. The two last assignments are sound since it means that whatever the value of $\delta(t)$ fulfilling the order between classes, the firing sequence w leads to bisimilar configurations (as being bisimilar to the same configuration of the automaton).

4. A discrete step into a time-open region. In order to reach a time-open region by a discrete step, the corresponding transition must start from a time-open region and must not reset any clock. Let $(\ell_r, min_r + \delta + d) \in r$ and $(\ell_r, min_r + \delta + d) \xrightarrow{e} (l', min_r + \delta + d)$. Here we have used the hypothesis that no clock is reset. Then there is a firing sequence $(M_r, \nu_r + proj_r(\delta) + d) \xrightarrow{w}$ labeled by e. Due to Lemma 4.5, $(M_r, \nu_r + proj_r(\delta)) \xrightarrow{w}$. $(\ell_r, v_r + \delta)$ is bisimilar to $(M_r, \nu_r + proj_r(\delta))$.

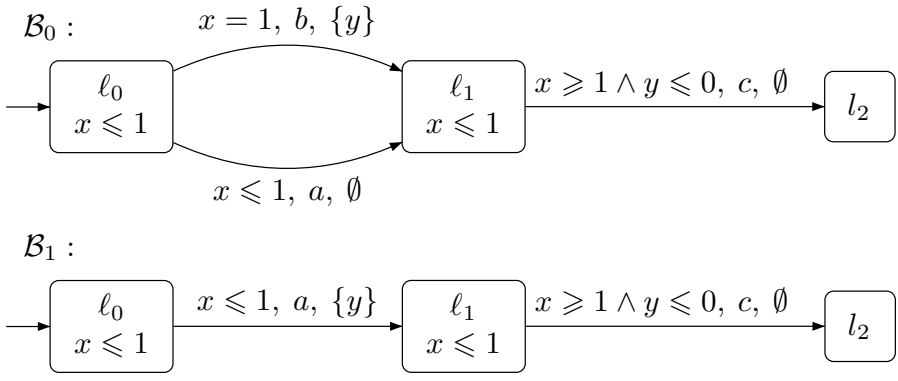


Figure 4.15. *Two automata with different behavior with respect to bisimulation with a TPN*

Thus $(\ell_r, min_r + \delta) \xrightarrow{e} (l', min_r + \delta) \xrightarrow{d} (l', min_r + \delta + d)$. Then this region can be reached via a discrete step into a time-closed region followed by a time step. So we do not need to examine this case. ∎

4.6.3. *A characterization of bisimilarity*

The characterization of TA bisimilar to some TPN is closely related to the topological closure of reachable regions: it states that any region intersecting the topological closure of a reachable region is also reachable and that a discrete step either from a region or from the minimal vector of its topological closure is possible in the whole topological closure. The two automata $\mathcal{B}_0$ and $\mathcal{B}_1$ in Figure 4.15 will illustrate our results: the automaton $\mathcal{B}_0$ admits a bisimilar TPN, whereas $\mathcal{B}_1$ does not. In what follows, we suppose that any atomic constraint related to a clock x occurring in the invariant of a location is added to the guard of each incoming transition which does not reset x.

THEOREM 4.9.– *[Characterization of TA bisimilar to some TPN] Let $\mathcal{A}$ be a (label-free) timed automaton, let $R(\mathcal{A})$ be its region automaton with respect to the 1-grid and a constant K strictly greater than any constant occurring in the automaton, then $\mathcal{A}$ is weakly timed bisimilar to a time Petri net iff: $\forall r \in R(\mathcal{A})$, $\forall e$ an edge of $\mathcal{A}$: (a) Every region r' such that $r' \cap \overline{r} \neq \emptyset$ is reachable; (b) $\forall (\ell_r, v) \in r$, $(\ell_r, v) \xrightarrow{e} \Rightarrow (\ell_r, min_r) \xrightarrow{e}$; (c) $\forall (\ell_r, v) \in \overline{r}$, $(\ell_r, min_r) \xrightarrow{e} \Rightarrow (\ell_r, v) \xrightarrow{e}$.*

Furthermore, if these conditions are satisfied then we can build a 1-bounded 2-TPN bisimilar to $\mathcal{A}$ whose size is linear with respect to the size of $\mathcal{A}$ and a 1-bounded 1-TPN bisimilar to $\mathcal{A}$ whose size is exponential with respect to the size of $\mathcal{A}$.

We note by $\mathcal{TA}^{-}$ this class of automata. Using the theorem, we justify why the automaton $\mathcal{B}_1$ does not admit a bisimilar TPN. The region $r = \{(\ell_1, x = 1 \wedge 0 < y < 1\}$ is reachable. The guard of edge c is true in $min_r = (\ell_1, (1, 0))$, whereas it is false in r.

We prove Theorem 4.9 in three steps in the next sections.

4.6.4. *Proof of necessity*

The fact that conditions *(a)*, *(b)*, and *(c)* are satisfied with respect to the g-grid and the constant $K = \infty$ is straightforward:

– *(a)* This assertion is included in the inductive assertions.

– *(b)* Let r be a reachable region, let $(\ell_r, min_r + \delta) \in r$ be a configuration with $\delta \in [0, 1/g[^X$, then $\exists (M, \nu)\ \nu \in \mathbb{N}_g^{En(M)}$ bisimilar to (ℓ_r, min_r) and $(M, \nu + \delta')$ with $\delta' \in [0, 1/g[^{En(M)}$ bisimilar to $(\ell_r, v + \delta)$. Suppose that $(\ell_r, min_r + \delta) \xrightarrow{e}$, then $(M, \nu + \delta') \xrightarrow{w}$ with w as an instantaneous firing sequence and $label(w) = e$. Now according to Lemma 4.5(b), $(M, \nu) \xrightarrow{w}$, thus $(\ell_r, min_r) \xrightarrow{e}$.

– *(c)* Let r be a region and $(\ell_r, min_r + \delta) \in \overline{r}$ with $\delta \in [0, 1/g]^X$ thus $\exists (M, \nu)$ bisimilar to (ℓ_r, min_r) and $(M, \nu + \delta')$ with $\delta' \in [0, 1/g]^{En(M)}$ bisimilar to $(\ell_r, min_r + \delta)$. Suppose that $(\ell_r, min_r) \xrightarrow{e}$, then $(M, \nu) \xrightarrow{w}$ with w as an instantaneous firing sequence and $label(w) = e$. Now according to Lemma 4.5(a), $(M, \nu + \delta') \xrightarrow{w}$, thus $(\ell_r, min_r + \delta) \xrightarrow{e}$.

In order to complete the proof, we successively show that if the conditions are satisfied with respect to the g-grid and infinite constant, they are satisfied with respect to the 1-grid and infinite constant and when satisfied w.r.t the 1-grid and infinite constant, they are satisfied w.r.t the 1-grid and the usual finite constant. This is done by the next two lemmas.

LEMMA 4.7.– *[About the conditions and the grid] Let $\mathcal{A}$ be a timed automaton and $g \in \mathbb{N}_{>0}$. If the conditions (a),(b),(c) are satisfied by the region automaton associated with the g-grid, then they are satisfied by the region automaton associated with the 1-grid (where in both cases the constant $K = \infty$).*

Proof. Let us denote by $R(\mathcal{A})^g$ the region automaton of A with respect to the g-grid. By definition of regions, we note that r, which is a region of $R(\mathcal{A})$, is a finite union of regions of $R(\mathcal{A})^g$ (for example, $r = \bigcup_{i=1..k} r_i$). Thus, $\overline{r} = \bigcup_{i=1..k} \overline{r_i}$ which proves the implication for (a).

Assume that (b) is satisfied by $R(\mathcal{A})^g$. Let $(\ell_r, min_r + \delta + d) \in r$ be a region of $R(\mathcal{A})$ and assume $(\ell_r, min_r + \delta + d) \xrightarrow{e}$. We define δ' by $\delta'(x) = \delta(x)/g$. Then since $\mathcal{A}$ has integer constraints, $(\ell_r, min_r + \delta' + d/g) \xrightarrow{e}$. Moreover, this configuration

belongs to r and then to a region $r' \in R(\mathcal{A})^g$ whose minimal vector is min_r. Then, by applying *(b)*, we obtain $(\ell_r, min_r) \xrightarrow{e}$.

Assume that *(c)* is satisfied by $R(\mathcal{A})^g$. Let $(\ell_r, v) \in \overline{r}$ where r is a region of $R(\mathcal{A})$ and assume $(\ell_r, min_r) \xrightarrow{e}$. Then there is an increasing path among the minimum vectors of regions of $R(\mathcal{A})^g$ all included in $\overline{r}$. This path is such that any two consecutive elements belong to the closure of some region; it starts at (ℓ_r, min_r) and finishes at (ℓ_r, min_{r_*}) such that $(\ell_r, v) \in \overline{r_*}$ (with r_* as a region of $R(\mathcal{A})^g$). Thus, by applying iteratively *(c)* yields $(\ell_r, v) \xrightarrow{e}$. ∎

LEMMA 4.8.– *[About the conditions and the constant K] Let $\mathcal{A}$ be a timed automaton. If the conditions (a),(b),(c) are satisfied by the region automaton associated with the 1-grid and constant $K = \infty$, then they are satisfied by the region automaton associated with the 1-grid and a finite constant.*

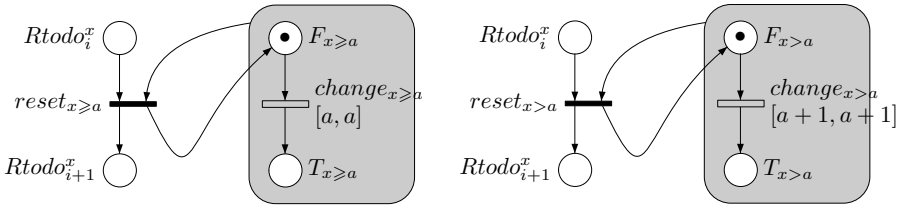
Proof. Let us denote by $R(\mathcal{A})^\infty$ the region automaton of A with respect to $K = \infty$. Let r be a reachable region in $R(\mathcal{A})$ and $reach(r)$ the associated region of $R(\mathcal{A})^\infty$. Note that $\ell_{reach(r)} = \ell_r$ and that $\forall x \in ActX_r, min_{reach(r)} = min_r$ and $\forall x \in X, min_{reach(r)} \geqslant min_r$. Suppose that $reach(r)$ is time-closed (resp. time-open) then r admits a time-closed (resp. time-open) description where the ord_r and $ord_{reach(r)}$ mappings are identical for clocks in $ActX_r$. Thus, $\forall (\ell_r, v) \in r, \exists (\ell_r, v') \in reach(r)$ such that $\forall x \in ActX_r, v'(x) = v(x)$.

Now take a convergent sequence $lim_{i\to\infty}(\ell_r, v_i) = (\ell_r, v)$ with $(\ell_r, v_i) \in r$ so that $(\ell_r, v) \in \overline{r}$. Then the corresponding sequence $\{(\ell_r, v'_i)\}$ being bounded admits an accumulation point $(\ell_r, v') \in \overline{r}$. It is routine to show that (ℓ_r, v) and (ℓ_r, v') belong to the same region in $R(\mathcal{A})$. This proves that condition *(a)* for $R(\mathcal{A})^\infty$ implies condition *(a)* for $R(\mathcal{A})$.

Assume that *(b)* is satisfied by $R(\mathcal{A})^\infty$. Let $(\ell_r, v) \in r$ be a reachable region of $R(\mathcal{A})$ and $(\ell_r, v) \xrightarrow{e}$. Let $reach(r)$ be the associated reachable region of $R(\mathcal{A})^\infty$ then $\exists (\ell_r, v') \in reach(r)$ strongly time bisimilar to (ℓ_r, v), thus $(\ell_r, v') \xrightarrow{e}$. Using condition *(b)*, $(\ell_r, min_{reach(r)}) \xrightarrow{e}$. Since $(\ell_r, min_{reach(r)})$ is strongly time bisimilar to (ℓ_r, min_r), we have $(\ell_r, min_r) \xrightarrow{e}$.

Assume that *(c)* is satisfied by $R(\mathcal{A})^\infty$ and consider $(\ell_r, v) \in \overline{r}$ where r is a region of $R(\mathcal{A})$ and $(\ell_r, min_r) \xrightarrow{e}$. Let $reach(r)$ be the associated reachable region of $R(\mathcal{A})^\infty$, then $\exists (\ell_r, v') \in \overline{reach(r)}$ strongly time bisimilar to (ℓ_r, v). Since $(\ell_r, min_{reach(r)})$ is strongly time bisimilar to (ℓ_r, min_r), $(\ell_r, min_{reach(r)}) \xrightarrow{e}$. Thus, using condition *(c)*, $(\ell_r, v') \xrightarrow{e}$. By bisimilarity, we obtain $(\ell_r, v) \xrightarrow{e}$. ∎

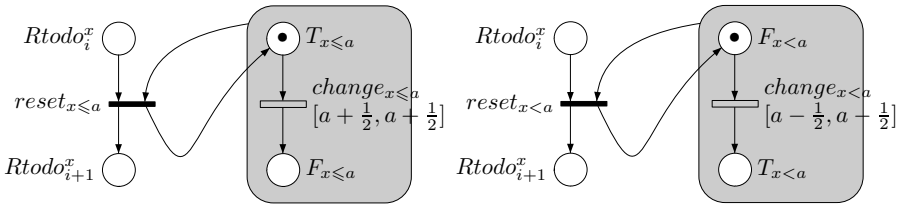
We now give the proof that the condition is sufficient. The proof is split into two parts, corresponding respectively to the construction of a 2-TPN and the construction of a 1-TPN.



(a) Widget for condition $x \geqslant a$ (assume $a > 0$)

(b) Widget for condition $x > a$

Figure 4.16. *Widgets for conditions $x \geqslant a$ and $x > a$*



(a) Widget for condition $x \leqslant a$

(b) Widget for condition $x < a$ (assume $a > 0$)

Figure 4.17. *Widgets for conditions $x \leqslant a$ and $x < a$*

4.6.5. *First construction*

Proof for the first construction of sufficiency. We first describe the construction of a 2-TPN $\mathcal{N}$ bisimilar to $\mathcal{A}$. The principles of this construction are similar to those used for the language equivalence. We build a subnet per elementary condition (including the part associated with the clock resetting). However, except for the conditions $x \geqslant c$ and the resetting part, all the constructions are different. We first note that $x < a$ occurring in an invariant may be safely omitted. Indeed (see the assumptions on timed automata), it never forbids the state to be entered. If it did forbid the progress of time in some configuration, then the associated region would be a maximal time-open region r. Due to condition (a), $\overline{r}$ is reachable but since r is time-open, $\overline{r} \cap succ(r) \neq \emptyset$, so that $succ(r)$ is reachable which contradicts the maximality of r.

All edges of $\mathcal{N}$ are weighted by 1. Unless explicitly stated, the transitions are labeled by ϵ.

– With each location ℓ of the automaton, we associate an eponymous place ℓ. The place ℓ is initially marked iff the location ℓ is the initial one.

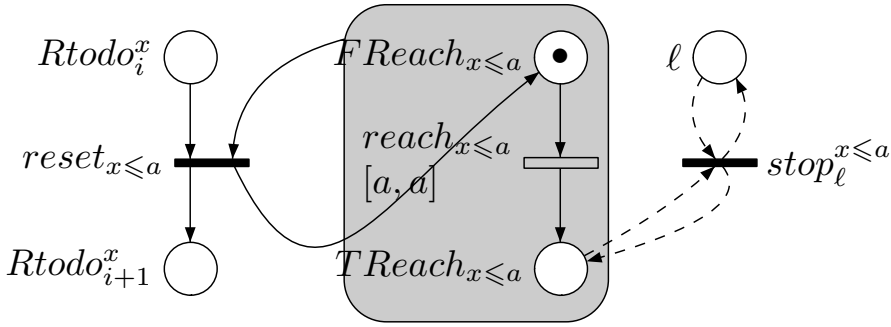


Figure 4.18. *Widget for invariant* $x \leqslant a$

– The conditions associated with a clock x are arbitrarily numbered from 1 to $n(x)$, where $n(x)$ is the number of such conditions. We consider that when $x \leqslant a$ ($a \neq 0$) occurs in at least one transition and in at least one invariant, it is associated with two different conditions. Then we add places $\{Rtodo_i^x\}_{i\leqslant n(x)+1}$ for the management of the resets.

– With each condition $x \geqslant a$ ($a \neq 0$) occurring in a transition of the automaton, we associate a widget (see Figure 4.16 (a)) composed by two places $T_{x\geqslant a}, F_{x\geqslant a}$ and three transitions $change_{x\geqslant a}, reset^1_{x\geqslant a}, reset^2_{x\geqslant a}$. The place $F_{x\geqslant a}$ is initially

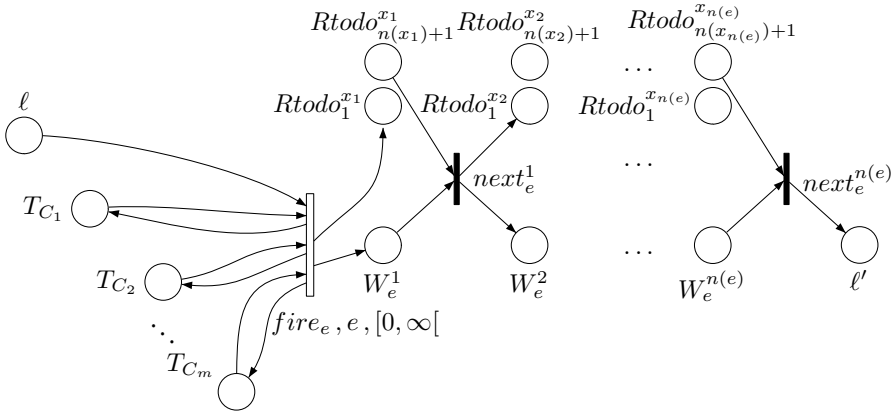


Figure 4.19. *Widget of an edge*
$(l, \gamma = \{c_1, \ldots, c_{m(e)}\}, e, R = \{x_1, \ldots, x_{n(e)}\}, l')$

marked while $T_{x\geqslant a}$ is unmarked. The interval associated with $change_{x\geqslant a}$ is $[a,a]$; ${}^\bullet change_{x\geqslant a} = \{F_{x\geqslant a}\}$ and $change_{x\geqslant a}{}^\bullet = \{T_{x\geqslant a}\}$. The interval associated with $reset^1_{x\geqslant a}$ and $reset^2_{x\geqslant a}$ is $[0,0]$. Let i be the number of the condition $x \geqslant a$. ${}^\bullet reset^1_{x\geqslant a} = \{F_{x\geqslant a}, Rtodo^x_i\}$ and $reset^1_{x\geqslant a}{}^\bullet = \{F_{x\geqslant a}, Rtodo^x_{i+1}\}$. ${}^\bullet reset^2_{x\geqslant a} = \{T_{x\geqslant a}, Rtodo^x_i\}$ and $reset^2_{x\geqslant a}{}^\bullet = \{F_{x\geqslant a}, Rtodo^x_{i+1}\}$.

– With each condition $x > a$ occurring in a transition of the automaton, we associate a widget (Figure 4.16 (b)) composed by two places $T_{x>a}, F_{x>a}$ and three transitions $change_{x>a}, reset^1_{x>a}, reset^2_{x>a}$. The place $F_{x>a}$ is initially marked while $T_{x>a}$ is unmarked. The interval associated with $change_{x>a}$ is $[a+1, a+1]$; ${}^\bullet change_{x>a} = \{F_{x>a}\}$ and $change_{x>a}{}^\bullet = \{T_{x>a}\}$. The interval associated with $reset^1_{x>a}$ and $reset^2_{x>a}$ is $[0,0]$. Let i be the number of the condition $x > a$. ${}^\bullet reset^1_{x>a} = \{F_{x>a}, Rtodo^x_i\}$ and $reset^1_{x>a}{}^\bullet = \{F_{x>a}, Rtodo^x_{i+1}\}$. ${}^\bullet reset^2_{x>a} = \{T_{x>a}, Rtodo^x_i\}$ and $reset^2_{x>a}{}^\bullet = \{F_{x>a}, Rtodo^x_{i+1}\}$.

– With each condition $x \leqslant a$ occurring in a transition of the automaton, we associate a widget (Figure 4.17 (a)) composed by two places $T_{x\leqslant a}, F_{x\leqslant a}$ and three transitions $change_{x\leqslant a}, reset^1_{x\leqslant a}, reset^2_{x\leqslant a}$. The place $T_{x\leqslant a}$ is initially marked while $F_{x\leqslant a}$ is unmarked. The interval associated with $change_{x\leqslant a}$ is $[a+1/2, a+1/2]$; ${}^\bullet change_{x\leqslant a} = \{T_{x\leqslant a}\}$ and $change_{x\leqslant a}{}^\bullet = \{F_{x\leqslant a}\}$. The interval associated with $reset^1_{x\leqslant a}$ and $reset^2_{x\leqslant a}$ is $[0,0]$. Let i be the number of the condition $x \leqslant a$. ${}^\bullet reset^1_{x\leqslant a} = \{T_{x\leqslant a}, Rtodo^x_i\}$ and $reset^1_{x\leqslant a}{}^\bullet = \{T_{x\leqslant a}, Rtodo^x_{i+1}\}$. ${}^\bullet reset^2_{x\leqslant a} = \{F_{x\leqslant a}, Rtodo^x_i\}$ and $reset^2_{x\leqslant a}{}^\bullet = \{T_{x\leqslant a}, Rtodo^x_{i+1}\}$.

– With each condition $x < a$ ($a \neq 0$) occurring in a transition of the automaton, we associate a widget (Figure 4.17 (b)) composed by two places $T_{x<a}$, $F_{x<a}$ and three transitions $change_{x<a}, reset^1_{x<a}, reset^2_{x<a}$. The place $T_{x<a}$ is initially marked while $F_{x<a}$ is unmarked. The interval associated with $change_{x<a}$ is $[a-1/2, a-1/2]$; ${}^\bullet change_{x<a} = \{T_{x<a}\}$ and $change_{x<a}{}^\bullet = \{F_{x<a}\}$. The interval associated with $reset^1_{x<a}$ and $reset^2_{x<a}$ is $[0,0]$. Let i be the number of the condition $x < a$. ${}^\bullet reset^1_{x<a} = \{T_{x<a}, Rtodo^x_i\}$ and $reset^1_{x<a}{}^\bullet = \{T_{x<a}, Rtodo^x_{i+1}\}$. ${}^\bullet reset^2_{x<a} = \{F_{x<a}, Rtodo^x_i\}$ and $reset^2_{x<a}{}^\bullet = \{T_{x<a}, Rtodo^x_{i+1}\}$.

– With each condition $x \leqslant a$ ($a \neq 0$) in an invariant, we associate a widget (Figure 4.18) composed by two places $TReach_{x\leqslant a}, FReach_{x\leqslant a}$ and three transitions $reach_{x\leqslant a}, resetR^1_{x\leqslant a}, resetR^2_{x\leqslant a}$. The place $FReach_{x\leqslant a}$ is initially marked while $TReach_{x\leqslant a}$ is unmarked. The interval associated with $reach_{x\leqslant a}$ is $[a,a]$; ${}^\bullet change_{x\leqslant a} = \{FReach_{x\leqslant a}\}$ and $change_{x\leqslant a}{}^\bullet = \{TReach_{x\leqslant a}\}$. The intervals associated with $resetR^1_{x\leqslant a}$ and $resetR^2_{x\leqslant a}$ is $[0,0]$. Let i be the number of the condition $x \leqslant a$. ${}^\bullet resetR^1_{x\leqslant a} = \{FReach_{x\leqslant a}, Rtodo^x_i\}$ and $resetR^1_{x\leqslant a}{}^\bullet = \{FReach_{x\leqslant a}, Rtodo^x_{i+1}\}$. ${}^\bullet resetR^2_{x\leqslant a} = \{TReach_{x\leqslant a}, Rtodo^x_i\}$ and $resetR^2_{x\leqslant a}{}^\bullet = \{FReach_{x\leqslant a}, Rtodo^x_{i+1}\}$.

– With each edge $(l, \gamma = \{c_1, \ldots, c_{m(e)}\}, e, R = \{x_1, \ldots, x_{n(e)}\}, l')$, we associate a widget (Figure 4.19) composed by places $\{W^i_e\}_{i\leqslant n(e)}$ and transitions

$fire_e, \{next_e^i\}_{i \leqslant n(e)}$. The transition $fire_e$ has the label e; its interval is $[0, \infty[$; ${}^\bullet fire_e = \{l, T_{c_1}, \ldots, T_{c_{m(e)}}\}$ and $fire_e{}^\bullet = \{W_e^1, Rtodo_1^{x_1}, T_{c_1}, \ldots, T_{c_{m(e)}}\}$. The interval associated with transitions $next_e^i$ is $[0, 0]$. $\forall i < n(e), {}^\bullet next_e^i = \{W_e^i, Rtodo_{n(x_i)+1}^{x_i}\}$ and $next_e^i{}^\bullet = \{W_e^{i+1}, Rtodo_1^{x_{i+1}}\}$. ${}^\bullet next_e^{n(e)} = \{W_e^{n(e)}, Rtodo_{n(x_{n(e)})+1}^{x_{n(e)}}\}$ and $next_e^{n(e)\bullet} = \{\ell'\}$. When $R = \emptyset$, the widget is reduced to the transition $fire_e$ with ${}^\bullet fire_e = \{\ell, T_{c_1}, \ldots, T_{c_{m(e)}}\}$ and $fire_e{}^\bullet = \{\ell', T_{c_1}, \ldots, T_{c_{m(e)}}\}$.

– If a condition $x \leqslant 0$ occurs in the invariant of l, then we add a transition $stop_\ell$ with interval $[0, 0]$, ${}^\bullet stop_\ell = stop_\ell{}^\bullet = \{\ell\}$. If a condition $x \leqslant a$ ($a \neq 0$) occurs in the invariant of l, then we add a transition $stop_\ell^{x \leqslant a}$ with interval $[0, 0]$, ${}^\bullet stop_\ell^{x \leqslant a} = stop_\ell^{x \leqslant a\bullet} = \{\ell, TReach_{x \leqslant a}\}$.

We decompose the reachable configurations (and markings) into *intermediate* (some W_e^i is marked) and *permanent* ones (some ℓ is marked). An easy induction shows that in permanent configurations (M, ν) the enabled timed transitions relative to a clock are "synchronized": $\nu(change_c) = \nu(change_{c'}) = \nu(reach_{c''})$ as soon as c, c', c'' relate to the same clock x. We define $\nu(x)$ as this common value if at least one such transition is enabled and otherwise $\nu(x) = K(x)$ where $K(x)$ is the maximal value relative to clock x occuring in the net $\mathcal{N}$. Furthermore, from any intermediate configuration (M, ν), the behavior of the net is quasi-deterministic until it reaches a permanent configuration: there are only firing sequences (i.e. no time step) and some of them lead to permanent configurations. Furthermore, these permanent configurations (for example, (M_{next}, ν_{next})) have the same marked place ℓ and the same values $\nu_{next}(x)$.

It is also obvious that once some $fire_e$ is fired, the construction ensures the existence of a "resetting" sequence which reinitializes the widgets associated with the clocks to be reset.

Bisimulation relation. We now define the relation $\mathcal{R}$ between reachable configurations of the automaton $\mathcal{A}$ and the net $\mathcal{N}$. Let us define $(\ell, v)\mathcal{R}(M, \nu)$ iff:

– either M is a permanent marking and $M(\ell)$ is marked and if $\nu(x) < K(x)$ then $v(x) = \nu(x)$, otherwise $v(x) \geqslant K(x)$;

– or M is an intermediate marking leading to some permanent (M_{next}, ν_{next}) and $(\ell, v)\mathcal{R}(M_{next}, \nu_{next})$. This definition is sound due to the common features of the different (M_{next}, ν_{next}).

It remains to prove that $\mathcal{R}$ is a bisimulation, which is done in the next lemma.

LEMMA 4.9.– *The relation $\mathcal{R}$ defined above is a weak timed bisimulation.*

Proof. We first consider moves from $\mathcal{A}$.

Case 1: $(\ell, v) \xrightarrow{e} (\ell', v')$

First, let us prove that $(M, \nu) \xrightarrow{\sigma}$ with σ labeled by e. At first, σ begins by σ' which consists of firing all the $change_c$ fireable leading to some (M', ν') (with $(\ell, v)\mathcal{R}(M', \nu')$). Now we prove that $(M', \nu') \xrightarrow{fire_e}$. By definition of $\mathcal{R}$, $M(\ell)$ is marked. Let c be a condition occuring in the guard of e. If $c = [x \geqslant a]$ then $v(x) \geqslant a$ which implies that $\nu(x) \geqslant a$ and that $T_{x \geqslant a}$ is marked (eventually with the help of σ'). If $c = [x > a]$ then let r be the region to which (ℓ, v) belongs. $min_r(x) = \lfloor v(x) \rfloor$. Using condition (b), $(l, min_r) \xrightarrow{e}$. Thus, $v(x) \geqslant min_r(x) \geqslant a + 1$, which implies $\nu(x) \geqslant a + 1$ and that $T_{x>a}$ is marked (eventually with the help of σ'). If $c = [x \leqslant a]$ then $v(x) \leqslant a$ which implies $\nu(x) \leqslant a$ and that $T_{x \leqslant a}$ is marked (remember that $change_{x \leqslant a}$ fires when $\nu(x) = a + 1/2$). If $c = [x < a]$ then let r be the region to which (ℓ, v) belongs. Then there exists $(\ell, v_1) \in \overline{r}$ with $v_1(x) = \lceil v(x) \rceil$. Using condition (b) and then (c), $(l, v_1) \xrightarrow{e}$. Thus, $v(x) \leqslant v_1(x) \leqslant a - 1$, which implies $\nu(x) \leqslant a - 1$ and that $T_{x<a}$ is marked (remember that $change_{x<a}$ fires when $\nu(x) = a - 1/2$). Thus $fire_e$ is fireable from (M', ν'). We complete σ by the "resetting" sequence leading to a configuration bisimilar to (ℓ', v'). If M is an intermediate marking, we fire a sequence leading to some (M_{next}, ν_{next}) and perform the previous simulation.

Case 2: $(\ell, v) \xrightarrow{d} (\ell, v + d)$

Suppose that $x \leqslant a$ belongs to the invariant of ℓ. This means that $v(x) + d \leqslant a$. Thus, from (M, ν), we let a time d elapse interleaved with possible firings of $change$ transitions. The $stop$ transitions associated with ℓ will be possibly fireable but only at the end of this step sequence.

If M is an intermediate marking, we fire a sequence leading to some (M_{next}, ν_{next}) and perform the previous simulation.

Conversely, we consider moves from $\mathcal{N}$.

Case 3: $(M, \nu) \xrightarrow{t} (M', \nu')$

If t is labeled by ϵ, then by construction $(\ell, v)\mathcal{R}(M', \nu')$.

Thus, we only need to examine the case of $fire_e$ (M is then a permanent marking). Let r be the region to which (ℓ, v) belongs. We will show that $(\ell, min_r) \xrightarrow{e}$. Then by condition (c), we will obtain that $(\ell, v) \xrightarrow{e}$.

Let c be a condition occuring in the guard of e. If $c = [x \geqslant a]$ then $T_{x \geqslant a}$ is marked, which implies that $\nu(x) \geqslant a$ and then $v(x) \geqslant a$, thus $min_r(x) = \lfloor v(x) \rfloor \geqslant a$. If $c = [x > a]$ then $T_{x>a}$ is marked, which implies that $\nu(x) \geqslant a + 1$ and then $v(x) \geqslant a + 1$ thus $min_r(x) = \lfloor v(x) \rfloor \geqslant a + 1 > a$. If $c = [x \leqslant a]$ then $T_{x \leqslant a}$ is marked, which

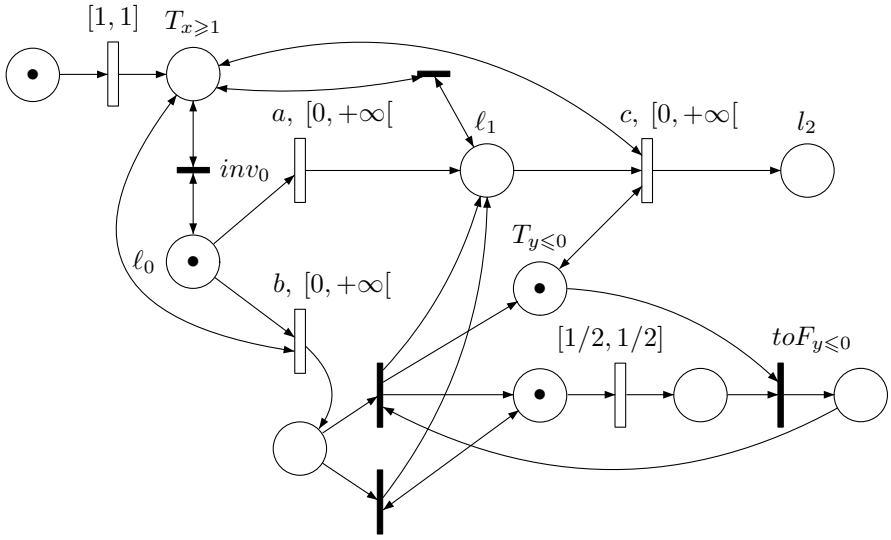


Figure 4.20. *The TPN bisimilar to* b_0

implies that $\nu(x) \leqslant a + 1/2$ and then $v(x) \leqslant a + 1/2$ thus $min_r(x) = \lfloor v(x) \rfloor \leqslant a$. If $c = [x < a]$ then $T_{x<a}$ is marked, which implies that $\nu(x) \leqslant a - 1/2$ and then $v(x) \leqslant a - 1/2$ thus $min_r(x) = \lfloor v(x) \rfloor \leqslant a - 1 < a$.

So $(\ell, v) \xrightarrow{e} (\ell', v')$ for some (ℓ', v'). By construction of $\mathcal{N}$ and definition of $\mathcal{R}$, $(\ell', v')\mathcal{R}(M', \nu')$.

Case 4: $(M, \nu) \xrightarrow{d} (M, \nu + d)$

An intermediate marking cannot let time elapse. Thus, M is a permanent marking. Consider an $x \leqslant a$ that belongs to the invariant of l. Then $a \neq 0$, otherwise, from state (M, ν), $stop_l$ must be fired and, in this case, time may not elapse. Similarly, since $stop_l^{x\leqslant a}$ is only possibly fireable from $(M, \nu + d)$, it follows that $\nu(x) + d \leqslant a$, thus $v(x) + d \leqslant a$.

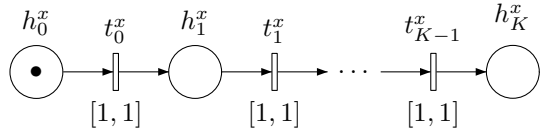
Consequently, $(\ell, v) \xrightarrow{d} (\ell, v + d)$ and obviously $(\ell, v + d)\mathcal{R}(M, \nu + d)$. ∎

We finally illustrate this construction on the timed automaton $\mathcal{B}_0$ in Figure 4.15 and its translation given below (with some simplifications related to this particular TA). For readability, immediate transitions (where interval $[0, 0]$ is represented in black and ε labels) are not shown.

First, note that the subnet associated with the constraint $y \leqslant 0$ switches the condition to false (firing of $toF_{y\leqslant 0}$) when the implicit value of y maintained in the net reaches $1/2$. Seemingly, this translation appears to be less constrained than the original condition. We explain how we prove that this translation is nevertheless sound. Let r be the region corresponding to the current configuration (ℓ, v) of the automaton simulated by the net. If the net is able to simulate a discrete step of the automaton, we prove that in the configuration (ℓ, min_r) of the automaton this step is also possible. Thus, by condition *(c)*, the step is also possible from (ℓ, v). On the other hand, if a discrete step is possible for (ℓ, v) in the automaton, we show that this step is also simulatable in the net using both conditions *(b)* and *(c)* and the following fact: $\forall x \in X, \exists(\ell_r, v'), (\ell_r, v'') \in \overline{r}$ such that $v'(x) = \lfloor v(x) \rfloor$ and $v''(x) = \lceil v(x) \rceil$. We also need to handle the invariants. First it is straightforward to observe that, due to condition *(a)*, an atomic constraint $x < c$ occuring in an invariant may be safely deleted since its effectiveness leads to the existence of a region r whose time-successor (which intersects $\overline{r}$) would not be reachable. The subnet associated with the atomic constraint $x \leqslant 1$ occuring in the invariant of ℓ_0 leads to transition inv_0 (not modifying the marking) which is fireable as soon as the simulated value of x reaches 1 and the place ℓ_0 is marked. Thus, time cannot progress except if the location is left.

4.6.6. *Second construction*

Proof for the second construction of sufficiency. When the conditions on the unlabeled timed automaton $\mathcal{A}$ are satisfied, we build a 1-TPN $\mathcal{N}$ with atomic semantics which is weakly timed bisimilar to $\mathcal{A}$. We suppose that all invariant conditions of a location are added to the guard of each ingoing transition. Recall that $K = m + 1$, where m is the maximal constant for $\mathcal{A}$. The construction of the TPN is a partial replication of both the region automaton of $\mathcal{A}$ and the class automaton, as explained later. There is first a subnet for each clock x, in which only the integral parts of x appear in the places (but with a fractional part that can reach 1).



Then we add one place C for each class $C = (\ell, Z)$ of the class automaton, with the initial class marked. Now let $e = (\ell, g, a, R, \ell')$ be a transition of $\mathcal{A}$. For each pair (v, v') of clock valuations in $\mathbb{N}^X$, with $v, v' \leqslant \overrightarrow{K}$, we build a subnet which simulates the transition $(\ell, v) \xrightarrow{e} (\ell', v')$, where we have $v'(x) = 0$ if $x \in R$ and $v'(x) = v(x)$ otherwise. Let $C_1 = (\ell, Z_1), \ldots, C_k = (\ell, Z_k)$ be the subset of classes

such that $\exists v'' \in Z_i \wedge \forall x \in X, v''(x) = v(x) \vee (v''(x) \geqslant K \wedge v(x) = K))$ for $1 \leqslant i \leqslant k$ and $C'_1, \ldots, C'_k$ be the classes obtained by applying transition e to $C_1, \ldots, C_k$ respectively. We have a transition with label e for each C_i (with $k = 2$ in the figure below), all with interval $[0, +\infty[$. Note that all reset operations for clocks in R are executed successively with instantaneous transitions. Moreover, the upper part of the net ensures that the invariant conditions of location l are satisfied (this part has been omitted for ℓ').

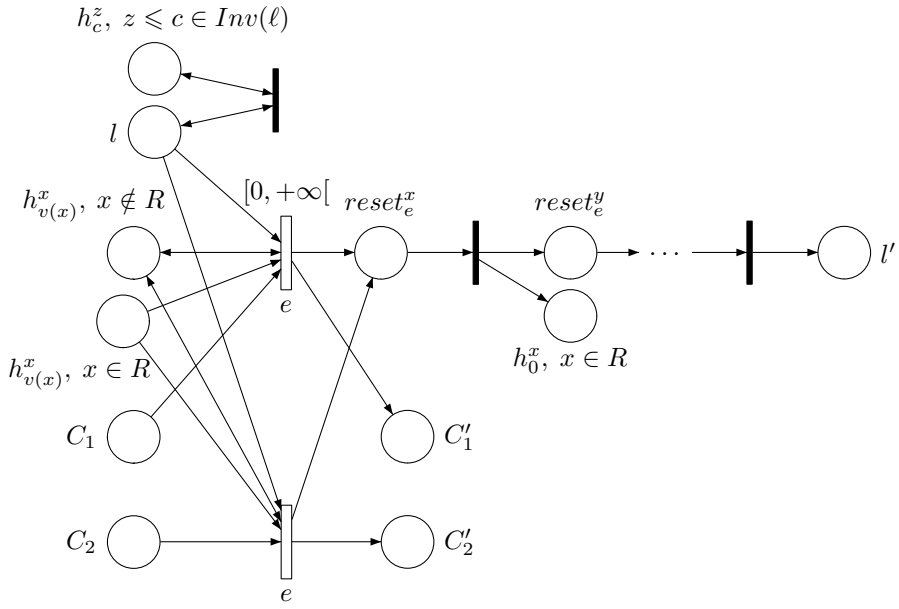


Figure 4.21. *Simulation of a transition*

Like in the previous proof, we say that a configuration (and the corresponding marking) (M, ν) of the TPN is permanent if $M(\ell) = 1$ for some l. Otherwise, it is an intermediate configuration (and marking), where $M(reset^x_e) = 1$ for some (exactly one of each) x and e, meaning that some reset operations are in progress. Here again, a permanent configuration is reached instantaneously from such an intermediate configuration, with only firing sequences completing the reset operations for transition e (interleaved with possibly transitions firings of some t^x_c).

Furthermore, for a configuration (M, ν), there is exactly one non-empty place h^x_c for each clock x. By writing c_x for the constant, such that $M(h^x_{c_x}) = 1$, we have either $c_x = K$ or $0 \leqslant \nu(t^x_{c_x}) \leqslant 1$, where $\nu(t^x_c)$ is the time elapsed since the arrival of the token in the place $h^x_{c_x}$. This means that the value of clock x is either $v(x) \geqslant K$

or $v(x) = c_x + \nu(t^x_{c_x})$ with $\lfloor v(x) \rfloor$ equal to either c_x or $c_x + 1$. In the latter case, transition $t^x_{c_x}$ can be fired instantaneously, leading to the configuration (M', ν') with one token in place $h^x_{c_x+1}$ and either $c_x + 1 = K$ or $\nu'(t^x_{c_x+1}) = 0$. We can thus reach a configuration where $c = (c_x)_{x \in X}$ is maximal.

Bisimulation relation. The relation $\mathcal{R}$ is defined as the set of pairs $((M, \nu), (\ell, v))$ such that:

– either (M, ν) is a permanent configuration with $M(\ell) = 1$, the relation between v and ν is the one described above, and there exists exactly one class $C = (\ell, Z)$ such that $M(C) = 1$ and $v \in Z$;

– or (M, ν) is an intermediate configuration leading to some permanent configuration (M', ν') such that $((M', \nu'), (\ell, v)) \in \mathcal{R}$.

We end the proof with an auxiliary lemma and the fact that $\mathcal{R}$ is a weak time bisimulation.

The following lemma, which relates regions and classes, shows how the class automaton will be used to control the firing of a transition when the minimal point c is not in the same region as v.

LEMMA 4.10.– *Let $\mathcal{A}$ be an automaton satisfying the conditions of Theorem 4.9, let $C = (\ell, Z)$ be a class of the class automaton and $(\ell, v) \in C$. Let $(\ell, v) \in r$ where r is a region with respect to to the choice $K = \infty$ (which means that there is an infinite number of regions). Then $\forall (\ell, v') \in \overline{r}, (\ell, v') \in C$. In particular, $(\ell, \lfloor v \rfloor) \in C$.*

Proof. The proof is obtained by induction on the reachability relation between regions. The case of a discrete step follows from conditions *(b)* and *(c)* of Theorem 4.9. The case of a time step follows from the choice of $K = \infty$ which implies that, given a region r, every item of $\overline{succ(r)}$ is reached by a time step from an item of $\overline{r}$. ∎

LEMMA 4.11.– *The relation $\mathcal{R}$ defined above is a weak time bisimulation.*

Proof. Assume that $(M, \nu)\mathcal{R}(\ell, v)$ and consider a move in $\mathcal{A}$.

Case 1: $(\ell, v) \xrightarrow{d} (\ell, v + d)$ (with $d \neq 0$). In this case, we must consider different subcases, according to the regions that can be reached by elapsing time. We consider only moves in which at most one different region is reached, whereas the general case would be a combination of those elementary moves. First note that since $v' = v + d$ can be reached, no invariant condition needs to be activated in $\mathcal{N}$. Moreover, if (M, ν) is an intermediate configuration, we first apply the sequence described above and reach the equivalent configuration (M_1, ν_1). Also in this case, since classes are unchanged by elapsing time, if we prove that a delay move is possible from (M_1, ν_1), we immediately obtain that the class is the same in the resulting configuration. Thus, the resulting configuration will be equivalent to $(\ell, v + d)$:

– if v belongs to a time-open region, the case where v' belongs to the same time-open region is easy, since it simply corresponds to a delay transition from (M_1, ν_1) in $\mathcal{N}$, each clock being in some h_c^x and staying inside (no token move), with $(M_1, \nu_1 + d)$ equivalent to $(\ell, v + d)$. If v' has reached an integer value, we consider a clock x with the greatest integral part, so that $v'(x) = \lfloor v(x) \rfloor + 1 = v(x) + d$ with $v(y) + d \leqslant \lfloor v(y) \rfloor + 1$ for all other clocks. In this case also, we obtain a delay move in $\mathcal{N}$ from (M_1, ν_1);

– if there are some clocks x for which $v(x)$ has an integer value, then elapsing time leads to the successor region, which is time-open. From (M_1, ν_1), it is possible to reach with instantaneous transitions a configuration (M_2, ν_2) where for all clocks with integer values, $M_2(h_c^x) = 1$ with c maximal, and (M_2, ν_2) is still equivalent to (ℓ, v). Now from (M_2, ν_2), a delay move can be applied so that $(M, \nu) \xrightarrow{*} (M_1, \nu_1) \xrightarrow{*} (M_2, \nu_2) \xrightarrow{d} (M_2, \nu_2 + d)$, with $(M_2, \nu_2 + d)\mathcal{R}(\ell, v + d)$.

Case 2: If $(\ell, v) \xrightarrow{e} (\ell', v')$ for some $e = (\ell, g, a, R, l')$, then condition (b) implies that a transition $(\ell, \lfloor v \rfloor) \xrightarrow{e} (\ell', \lfloor v' \rfloor)$ is also possible in $\mathcal{A}$. Here again we may have to apply from (M, ν) a sequence of instantaneous transitions, leading to (M_1, ν_1) where place l is marked and from there we can reach an equivalent configuration (M_2, ν_2) with $c = (c_x)_{x \in X}$ maximal. Let $C = (\ell, Z)$ be the class for which $M(C) = 1$, with $v \in Z$. From lemma 4.10, $(\ell, \lfloor v \rfloor)$ also belongs to C, and $\forall x \in X, \lfloor v \rfloor(x) = c_x \vee (\lfloor v \rfloor(x) \geqslant K \wedge c_x = K)$ so that the transition e (corresponding to this vector and this class) can be fired in $\mathcal{N}$, immediately followed according to the corresponding reset sequence, leading to (M', ν'). Since exactly one class C' is marked after e, we have $(M', \nu')\mathcal{R}(\ell', v')$ according to the definition of $\mathcal{R}$.

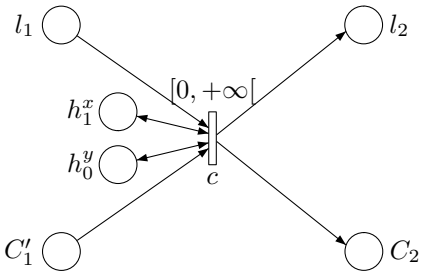
For the converse, we consider a move in $\mathcal{N}$.

Case 3: $(M, \nu) \xrightarrow{d} (M, \nu + d)$ (with $d \neq 0$). Then, neither reset transitions nor transitions of the form t_c^x can be fired in $\mathcal{N}$. Thus, the places h_c^x which contain a token are such that $\nu(t_c^x) < 1$ and $\nu(t_c^x) + d \leqslant 1$. For the state (ℓ, v), we have $M(\ell) = 1$ and $v(x) = c + \nu(t_c^x)$. The move $(\ell, v) \xrightarrow{d} (\ell, v + d)$ is possible in $\mathcal{A}$ since $(\ell, v + d)$ belongs either to the region of (ℓ, v) or to its time successor which is reachable by condition ***(a)***. Therefore, $(\ell, v) \xrightarrow{d} (\ell, v + d)$ in $\mathcal{A}$ with $(M, \nu + d)\mathcal{R}(\ell, v + d)$.

Case 4: $(M, \nu) \xrightarrow{t} (M', \nu')$. For any transition of $\mathcal{N}$, which is not associated with some transition $e = (\ell, g, a, R, l')$ in $\mathcal{A}$, no time can elapse so there is no need for a move in $\mathcal{A}$ because (M', ν') is still equivalent to (ℓ, v). Suppose now that t is associated with an edge e, so we have $M(\ell) = 1$, $M(C) = 1$ for some class $C = (\ell, Z)$ with $v \in Z$. Since t is fireable, considering the valuation $c = (c_x)_{x \in X}$ the construction implies that $\exists v'' \in Z$ such that $\forall x \in X, v''(x) = c_x \vee (v''(x) \geqslant K \wedge c_x = K)$, which implies that the segment $[v'', v] \subseteq Z$, from the convexity of Z, with $0 \leqslant v(x) - v''(x) = v(x) - c_x \leqslant 1$ for each x such that $c_x < K$. Thus, $[(\ell, v''), (\ell, v)]$ is contained in the topological closure $\overline{r}$ of some reachable region such that $min_r = c$ and $l = l_r$. Since

$(\ell, c) \xrightarrow{e} (\ell', c')$ is possible in $\mathcal{A}$, and (ℓ, v'') is strongly time bisimilar to (ℓ, c), we have $(\ell, v'') \xrightarrow{e} (\ell', v''')$. Now condition *(c)* implies that a move $(\ell, v) \xrightarrow{e} (\ell', v')$ is also possible in $\mathcal{A}$. From the definition, $(M', \nu')\mathcal{R}(\ell', v')$. ∎

For instance, for the automaton $\mathcal{B}_0$ from Figure 4.15, we have four classes: $C_0 = \{l_0, 0 \leqslant x = y \leqslant 1\}$, $C_1 = \{l_1, 0 \leqslant x = y \leqslant 1\}$, $C_1' = \{l_1, x = 1 \wedge y = 0\}$ and $C_2 = \{l_2, 0 \leqslant y = x - 1\}$. We show below the subnet corresponding to the transition c at point $(l_1, (1, 0))$ and class C_1'.



Consider the following run in $\mathcal{B}_0$: $(l_0, (0, 0)) \xrightarrow{a} (l_1, (0, 0)) \xrightarrow{1} (l_1, (1, 1))$. The simulation of this run by $\mathcal{N}$ may lead to the following configuration: l_1, h_0^x, h_0^y and C_1 are marked and t_0^x and t_0^y have been enabled for 1 time unit. Suppose that the sequence $t_0^x t_0'^x$ is fired by marking the place t_1^x, then without the input place C_1' the transition labeled c could be erroneously fired. Since C_1' is unmarked this firing is disabled.

4.6.7. *Complexity results*

This characterization leads to the following complexity results.

THEOREM 4.10.– *[Complexity results] Given a (label-free) timed automaton $\mathcal{A}$, deciding whether there is a TPN weakly timed bisimilar to $\mathcal{A}$ is **PSPACE**-complete. The reachability problem for the class $\mathcal{TA}^-$ is **PSPACE**-complete.*

Proof. The reachability problem for regions is in **PSPACE**. In order to check whether the condition (a) is false we *non-deterministically* pick a region r and a region r' which intersects $\overline{r}$ and check whether r is reachable and r' is not reachable. In order to check whether the condition (b) is false we *non-deterministically* pick a region r and a edge e and check whether r is reachable and e is fireable from r and not fireable from (l_r, min_r). In order to check whether the condition (c) is false we *non-deterministically* pick a region r, a region r' which intersects $\overline{r}$ and an edge e and check whether r is reachable and e is not fireable from r or r' and fireable

from (l_r, min_r). By Savitch's construction, we obtain a deterministic algorithm in **PSPACE**.

In order to show the **PSPACE**-hardness, we use the construction in [ACE 02] which reduces the acceptation problem for linear bounded Turing machine (LBTM) to the reachability problem for TA with restricted guards. The calculated TA (called $A_{\mathcal{M},w_0}$) satisfies the conditions (a) and (b) but does not satisfy the condition (c). However, it can be safely transformed in order to satisfy this condition by adding the invariant $t \leqslant 1$ to any state (q, i) and the invariant $t \leqslant 0$ to any state (i, θ, j). This intermediate automaton is now bisimilar to a TPN.

Then we transform the edges entering the end state by resetting t and at last we add an edge $(end, t = 0, e, \emptyset, end)$.

If the LBTM $\mathcal{M}$ does not accept the word w_0, then the state end is not reachable and $A_{\mathcal{M},w_0}$ satisfies the conditions (a),(b),(c).

If the LBTM $\mathcal{M}$ accepts the word w_0, then the state end is reachable and $A_{\mathcal{M},w_0}$ does not satisfy the condition (c) (the additional edge is fireable when entering end but not after letting the time elapse). The fact that the reachability problem for the class $\mathcal{TA}^-$ is **PSPACE**-complete was proved implicitly within the proof above. ∎

At last, we complete these results by adapting them to other models of TA. A simpler characterization holds for TA without strict (and diagonal) constraints. Despite the fact that excluding strict constraints simplifies the characterization, the complexity of the membership problem remains the same.

THEOREM 4.11.– *[TA without strict constraints] Let $\mathcal{A}$ be an unlabeled timed automaton without strict constraints, let $R(\mathcal{A})$ be its region with respect to the 1-grid, then $\mathcal{A}$ is weakly timed bisimilar to a time Petri net iff: $\forall r \in R(\mathcal{A})$, $\forall e$ an edge of $\mathcal{A}$, $\forall(\ell_r, v) \in r$, $(\ell_r, min_r) \xrightarrow{e}\Rightarrow (\ell_r, v) \xrightarrow{e}$. Furthermore, deciding whether there is a TPN weakly time bisimilar to $\mathcal{A}$ is **PSPACE**-complete.*

Proof. It is straightforward to show that conditions (a) and (b) are satisfied by an automaton without strict constraints. Similarly the condition (c) is easily deduced from the current condition when the automaton does not include strict constraints.

The **PSPACE** membership is obviously true. We note that although the net of [ACE 02] contains contraints $x_i > 1$, they can be safely changed to $x_i \geqslant 2$. Thus, the **PSPACE** hardness follows. ∎

4.7. Conclusion

In this chapter, we have investigated different questions relative to the expressiveness of TPNs. At first, we have shown that TAs and bounded TPNs (enlarged with

strict constraints) are equivalent with respect to the timed language equivalence. We have also provided a more general and efficient construction of a TPN equivalent to a TA than the previous ones. Then we have focused on the weak time bisimilarity equivalence and we have developed our main contribution: a characterization of TAs time bisimulatable by a TPN. From this characterization, we have proved that deciding whether a TA admits a time bisimilar TPN is a **PSPACE**-complete problem. Furthermore, the reachability problem is still **PSPACE**-complete for this subclass of TAs. Finally, we have proved that for bounded TPNs the different semantics lead to equivalent models with respect to the time bisimilarity but that this is no longer true with strict constraints.

We are now looking for similar (multiple) characterizations for TPNs enlarged with strict constraints since in this context the choice of the semantics is relevant. We will also try to apply the same techniques to compare the different models of Petri nets with time.

4.8. Bibliography

[ABD 01] ABDULLA P. A., NYLÉN A., "Timed Petri Nets and BQOs", *ICATPN'01*, vol. 2075 of *LNCS*, Springer-Verlag, p. 53–72, June 2001.

[ACE 02] ACETO L., LAROUSSINIE F., "Is Your Model Checker on Time? On the Complexity of Model Checking for Timed Modal Logics", *Journal of Logic and Algebraic Programming*, vol. 52-53, p. 7-51, Elsevier Science Publishers, August 2002.

[ALU 94] ALUR R., DILL D., "A theory of timed automata", *Theoretical Computer Science B*, vol. 126, p. 183–235, 1994.

[ALU 99] ALUR R., FIX L., HENZINGER T. A., "Event-Clock Automata: A Determinizable Class of Timed Automata", *Theoretical Computer Science*, vol. 211, p. 253–273, 1999.

[AUR 00] AURA T., LILIUS J., "A Causal Semantics for Time Petri Nets", *Theoretical Computer Science*, vol. 243, num. 1–2, p. 409–447, 2000.

[BER 91] BERTHOMIEU B., DIAZ M., "Modeling and Verification of Time Dependant Systems Using Time Petri Nets", *IEEE Transactions on Software Engineering*, vol. 17, num. 3, p. 259–273, March 1991.

[CAS 04] CASSEZ F., ROUX O. H., "Structural Translation of Time Petri Nets into Timed Automata", *Proceedings of Workshop on Automated Verification of Critical Systems (AVoCS'04)*, Electronic Notes in Computer Science, Elsevier, 2004.

[DIA 94] DIAZ M., SENAC P., "Time stream Petri nets: a model for timed multimedia information", *Proceedings of 15th International Conference on Application and Theory of Petri Nets (ICATPN'94)*, vol. 815 of *LNCS*, Zaragoza, Spain, Springer-Verlag, p. 219–238, 1994.

[DIL 89] DILL D. L., "Timing assumptions and verification of finite state concurrent systems", *Proceedings Workshop on Automatic Verification Methods for Finite State Systems*, vol. 407 of *LNCS*, Springer-Verlag, 1989.

[FRU 00] DE FRUTOS ESCRIG D., VALERO RUIZ V., MARROQUÍN ALONSO O., "Decidability of properties of timed-arc Petri nets", *ICATPN'00*, vol. 1825 of *LNCS*, Aarhus, Denmark, Springer-Verlag, p. 187–206, June 2000.

[HAA 02] HAAR S., SIMONOT-LION F., KAISER L., TOUSSAINT J., "Equivalence of Timed State Machines and safe Time Petri Nets", *Proceedings of WODES 2002*, Zaragoza, Spain, p. 119–126, 2002.

[KHA 96] KHANSA W., DENAT J., COLLART-DUTILLEUL S., "P-Time Petri Nets for manufacturing systems", *WODES'96*, p. 94–102, 1996.

[LIM 03] LIME D., ROUX O. H., "State class Timed Automaton of a Time Petri Net", *Proceedings of PNPM'03*, IEEE Computer Society, September 2003.

[MER 74] MERLIN P. M., A study of the recoverability of computing systems, PhD thesis, University of California, Irvine, CA, 1974.

[PEZ 99] PEZZÉ M., YOUNG M., "Time Petri Nets: A Primer Introduction", Tutorial presented at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications, Zaragoza, Spain, September 1999.

[RAM 74] RAMCHANDANI C., Analysis of asynchronous concurrent systems by timed Petri nets, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974.

[SIF 80] SIFAKIS J., "Performance Evaluation of Systems using Nets", BRAUER W., Ed., *Net Theory and Applications, Advanced Course on General Net Theory of Processes and Systems*, vol. 84 of *LNCS*, Springer-Verlag, p. 307–319, 1980.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 5

A "Maximum Node Clustering" Problem

5.1. Introduction

We consider the following problem: given an undirected graph $G(V, E)$, with $V = \{1, 2, \ldots, n\}$, $E \subseteq \{\{i, j\} : i, j \in V\}$, non-negative node profits π_i, $i \in V$, edge weights w_{ij}, $\{i, j\} \in E$, and a given capacity $B > 0$, determine a subset $S^* \subseteq V$ such that the total profit $\pi(S^*) = \sum_{i \in S^*} \pi_i$ is maximum and:

$$w(S^*) = \sum_{\{i,j\} \in \delta(S^*)} w_{ij} \leqslant B \qquad [5.1]$$

where $\delta(S^*)$ is the set of all edges having at least one endpoint in S^*. In what follows it is also assumed that $w(\{i\}) \leqslant B,\ \forall\, i \in V$, otherwise node i could not be part of any feasible solution and could be discarded in advance. We call such problem Maximum Node Clustering (MNC). This problem generalizes the $0/1$ Knapsack problem (see the reduction of Figure 5.1 in section 5.3) and is strictly related to the Dense K-Subgraph problem [FEI 01, BIL 05].

The problem models also a class of knapsack-like problems with quadratic capacity constraint, arising in some telecommunications network design problems [CAR 04, YAM 05]. In telecommunications networks with hierarchical architecture, terminal nodes — *terminals* in what follows — representing origins and destinations

Chapter written by Giuliana CARELLO, Federico DELLA CROCE, Andrea GROSSO and Marco LOCATELLI.

of traffic demands are connected to hub nodes in charge of aggregating small flows into larger ones and routing them on intra-hub backbone links. Usually hubs have a limit, B, on the amount of traffic they can handle — i.e. the amount of traffic they can aggregate/disaggregate and route. The amount of traffic to be faced by a given hub is the sum of the traffic related to terminals connected to it. Consider a set of terminals V, a subset $S \subseteq V$ connected to the considered hub and a traffic matrix $t_{ij}, (i,j) \in V \times V$. The amount of traffic to be faced by the hub is:

$$\sum_{i \in S} \sum_{j \in V} t_{ij} + \sum_{i \in S} \sum_{j \in V \setminus S} t_{ji} = \sum_{i \in S} \left[\sum_{j \in V} (t_{ij} + t_{ji}) - \sum_{j \in S} t_{ji} \right]$$

In certain environments we can specify meaningful "profits" coming from connecting a terminal i to the hub; this leads to a knapsack-like quadratic model:

$$\max \sum_{i \in V} \pi_i x_i \qquad [5.2]$$

subject to:

$$\sum_{i \in V} \left[\sum_{j \in V} (t_{ij} + t_{ji})\, x_i - \sum_{j \in V} t_{ji} x_i x_j \right] \leqslant B \qquad [5.3]$$

$$x_i \in \{0,1\},\ \forall i \in V \qquad [5.4]$$

where $x_i = 1$ iff the terminal i is connected to the considered hub.

The profits assigned to the terminals may be different. With all equal profits we aim to maximize the number of connected terminals. In another situation, suppose that a terminal i^* is considered and that the goal is to minimize the total amount of traffic that it sends on the intra-hubs links — this helps in reducing network costs, since a flow depending cost is usually given between every pair of hubs. Thus, the aim is to minimize $\sum_{j \in V \setminus S} (t_{i^*j} + t_{ji^*})$. Since $\sum_{j \in V} (t_{i^*j} + t_{ji^*})$ is a constant, this is equivalent to maximize $\sum_{j \in S} (t_{i^*j} + t_{ji^*})$. This leads to a quadratic knapsack problem in which the profit of an item j is given by $t_{i^*j} + t_{ji^*}$. Finally, [5.2] to [5.4] is the pricing problem in a column generation approach for a class of Bin Packing Problems with quadratic formulation of the capacity constraint [CAR 04]. In this case, completely

general profits are defined by the simplex multipliers at each iteration. Model [5.2]–[5.4] defines a special case of MNC, where terminals are mapped on graph nodes and weights $w_{ij} = t_{ij} + t_{ji}$ are associated with edges $\{i, j\}$: equation [5.1] enforces constraint [5.3].

In this chapter, we first show that the problem is strongly **NP**-complete (even for bipartite graphs) and that approximation ratios arbitrarily close to 2 are attainable in polynomial time. Then, we consider the special case where G is a tree (which remains **NP**-complete due to a straightforward reduction from the $0/1$ Knapsack problem) and show that it is solvable in pseudo-polynomial time by a dynamic programming approach. For this latter case we also develop an FPTAS based on the dynamic programming recursion and an exact exponential algorithm that runs in time $\mathcal{O}(2^{\frac{2}{3}|V|}\text{poly}(|V|))$.

5.2. Approximation algorithm for the general problem

We first recall the following problems which are instrumental for the results.

KNAPSACK (or $0/1$-knapsack problem). Given items $N = \{1, 2, \ldots, n\}$, item profits $p_1, \ldots, p_n$ and weights $W_1, \ldots, W_n$ and $b \geqslant 0$ determine $S \subseteq N$ such that $W(S) = \sum_{i \in S} W_i \leqslant b$ and $\sum_{i \in S} p_i$ is maximum (decision version: $\sum_{i \in S} p_i \geqslant K$ for a given K).

DENSE K-SUBGRAPH (DKS). Given a graph $G'(V', E')$, with edge weights W_{ij}, $K > 0$ and $Q > 0$, determine an $S' \subseteq V'$ such that $|S'| = K$ and:

$$\sum_{\{i,j\} \in E(S')} W_{ij} \geqslant Q \qquad [5.5]$$

where $E'(S') = \{\{i, j\} \in E' : i, j \in S'\}$.

The KNAPSACK problem is **NP**-complete in the ordinary sense [GAR 79], while the DKS problem is **NP**-complete in the strong sense even for unweighted bipartite graphs [COR 84].

The decision version of MNC calls for finding an $S \subseteq V$ such that $w(S) \leqslant B$ and $\pi(S) \geqslant R$ for a given R. The following proposition establishes the complexity of MNC even for the special case of bipartite graphs.

PROPOSITION 5.1.– *MNC is strongly* **NP***-complete.*

Proof. Feasibility of any $S \subseteq V$ can be checked in $\mathcal{O}(|E|)$ time, hence MNC $\in$ **NP**. A reduction DKS $\propto$ MNC is immediate: set $V \equiv V'$, $E \equiv E'$, $w_{ij} \equiv W_{ij}$ for all

$\{i,j\} \in E$, $\pi_i = 1$ for all $i \in V$, $B = \sum_{\{i,j\} \in E} W_{ij} - Q$, $R = |V| - K$. An $S \subseteq V$ exists such that $\pi(S) = |S| = R$ and $w(S) \leqslant B$ iff $S' = V \setminus S$ is a K-nodes subset satisfying [5.5]. ∎

We now discuss approximate algorithms for MNC on arbitrary graphs.

PROPOSITION 5.2.– *An approximate solution S_A such that:*

$$\frac{\pi(S^*)}{\pi(S_A)} \leqslant 2 + \varepsilon$$

is computable in polynomial time (in n and $1/\varepsilon$) for any $\varepsilon > 0$ and this bound is tight.

Proof. Let I be an instance of MNC with a graph $G(V,E)$, profits π_i, $i \in V$, weights w_{ij}, $\{i,j\} \in E$ and capacity B. We define n instances of KNAPSACK I_k, $k \in V$, as follows:

$$(I_k) : \begin{cases} N = \{1,2,\ldots,n\} \setminus \{k\} = V \setminus \{k\} \\ p_i = \pi_i, i \in N \\ W_i = \sum_{\substack{j:\{i,j\} \in E \\ j \neq k}} w_{ij}, i \in N \\ b = B - \sum_{j:\{k,j\} \in E} w_{kj} \end{cases}$$

As already noticed, we assume $W_i \leqslant B$ for all i (if not, node i cannot be part of the solution and can be eliminated). A (non-polynomial, at the moment) approximation algorithm is defined as follows:

1. $S_A := \emptyset$;
2. **for all** $k = 1,\ldots,n$ **do**
3. Solve I_k, let S_k^* be the optimal solution;
4. **if** $\pi_k + \pi(S_k^*) > \pi(S_A)$ **then** $S_A := S_k^* \cup \{k\}$;
5. **end for**;
6. **return** S_A;

Note that all $S_k^* \cup \{k\}$ are feasible for I.

From now on we consider an index k such that $k \in S^*$ and $\pi_k \geqslant \pi_i$ for all $i \in S^*$, and denote $W_k = \sum_{j:\{k,j\} \in E} w_{kj}$. We first prove that:

$$\pi(S^*) \leqslant 2[\pi_k + \pi(S_k^*)] \qquad [5.6]$$

(note that $\pi(S_A) \geqslant \pi_k + \pi(S_k^*)$).

Let $\tilde{I}_k$ be an instance of KNAPSACK defined like I_k, but restricted only to the (unknown) items appearing in S^* and let $\tilde{S}^*$ be its optimal solution. Clearly, $\pi(S_k^*) \geqslant \pi(\tilde{S}^*)$.

We first observe that, for S^* to be feasible for I, we must have:

$$W(S^* \setminus \{k\}) = \sum_{i \in S^* \setminus \{k\}} W_i \leqslant 2(B - W_k) \qquad [5.7]$$

since each w_{ij} is counted at most twice in $W(S_k^*)$. Consider a $t \in S^* \setminus (\tilde{S}^* \cup \{k\})$. If such a t does not exist, then $\tilde{S}^* \cup \{k\} \equiv S^*$ and the algorithm finds the optimal solution; otherwise, according to the optimality of $\tilde{S}^*$ we have:

$$W(\tilde{S}^*) + W_t > B - W_k \qquad [5.8]$$

By combining [5.7] and [5.8], we get:

$$W[S^* \setminus (\tilde{S}^* \cup \{k, t\})] \leqslant B - W_k$$

i.e. $S^* \setminus (\tilde{S}^* \cup \{k, t\})$ is feasible for $\tilde{I}_k$. Then, by optimality of $\tilde{S}^*$:

$$\pi[S^* \setminus (\tilde{S}^* \cup \{k, t\})] \leqslant \pi(\tilde{S}^*)$$

Finally, $\pi(S^*)$ can be written as:

$$\begin{aligned} \pi(S^*) &= \pi_k + \pi_t + \pi[S^* \setminus (\tilde{S}^* \cup \{k, t\})] + \pi(\tilde{S}^*) \\ &\leqslant 2[\pi_k + \pi(\tilde{S}^*)] \\ &\leqslant 2[\pi_k + \pi(S_k^*)] \end{aligned}$$

thus establishing equation [5.6].

In order to get a polynomial algorithm, we simply require that on line 3 of the algorithm the instances I_k are approximately solved by means of an FPTAS, with relative error bounded by $\frac{\varepsilon}{2}$. Then:

$$\frac{\pi(S^*)}{\pi(S_A)} = \frac{\pi(S^*)}{\pi_k + \pi(S_k^*)} \cdot \frac{\pi_k + \pi(S_k^*)}{\pi(S_A)} \leqslant 2\left(1 + \frac{\varepsilon}{2}\right) = 2 + \varepsilon$$

For the bound tightness, consider an instance I of MNC on a complete graph where $V = \{1,2,3,4\}$, $E = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$, $\pi_i = 1$ for all $i \in V$, $w_{ij} = \frac{1}{6}B$ for all $\{i,j\} \in E$. Each I_k has capacity $\frac{1}{2}B$, and optimal value 1, hence we get a heuristic solution with value 2, while $S^* = \{1,2,3,4\}$, $\pi(S^*) = 4$. ∎

REMARK 5.1.– If the underlying graph is bipartite, for example $G(V_1 \cup V_2, E)$, an even simpler algorithm giving an approximation ratio arbitrarily close to 2 is the following (the proof is straightforward). Solve two instances I_{V_1}, I_{V_2} of KNAPSACK where $N = V_1$ and $N = V_2$ respectively, and:

$$p_i = \pi_i$$

$$W_i = \sum_{j:\{i,j\}\in E} w_{ij}, \quad \text{for each } i \in N$$

$$b = B$$

then pick the best between the two solutions.

5.3. The tree case

In what follows we consider instances of MNC where the underlying graph is a tree $T(V,E)$ (Tree-MNC, TMNC). TMNC is easily seen to be **NP**-complete: Figure 5.1 sketches a simple reduction KNAPSACK $\propto$ TMNC where T is a star.

Without loss of generality, we introduce the following notation and assumptions, in use throughout this section. The nodes in the tree $T(V,E)$ are numbered $V = \{1,2,\ldots,n\}$; we assume that nodes are numbered so that:

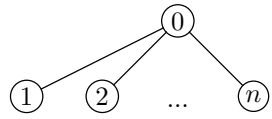
– each node $k \in V$ is connected to a set of *consecutively numbered* "children" nodes $\Gamma_k = \{j_k, j_k+1, \ldots, m_k\}$; j_k is the leftmost child and m_k is the rightmost child of node k;

– if $k \in V \setminus \{1\}$, $k' < k$ is the unique "parent" node of k; node 1 is considered the root of the tree.

The set of leaves of T is denoted by U. If $k \in V$ is a node, T_k is the subtree of T rooted at node k and for any $S \subset V$ we will denote by $S \cap T_k$ the set of all nodes of T_k contained in S. Finally, we will denote $\Pi = \sum_{i\in V} \pi_i$, $\pi_{\max} = \max\{\pi_i : i \in V\}$.

5.3.1. *Dynamic programming*

We develop a dynamic programming recursion for TMNC, which is in the spirit of those used for KNAPSACK [MAR 90]. Particularly, we note that the recursion parameter could be: (I) the maximum total weight allowed for a partial solution, or (II) the total profit z associated with a partial solution. We analyze in detail approach (II), which leads to an algorithm instrumental for further approximation results.



$$\begin{cases} N = \{1, 2, \ldots, n\} \\ W_1, \ldots, W_n \\ p_1, \ldots, p_n \\ b \\ K \end{cases} \Longrightarrow \begin{cases} V = \{0, 1, 2, \ldots, n\} \\ E = \{(0, i) \colon i = 1, 2, \ldots, n\} \\ w_{0i} = W_i, \ i = 1, 2, \ldots, n \\ \pi_0 = 0, \ \pi_i = p_i, \ i = 1, 2, \ldots, n \\ B = b \\ R = K \end{cases}$$

Figure 5.1. KNAPSACK $\propto$ *TMNC*

The goal of the recursion is to build subsets $S \subseteq V$, each one having a certain assigned profit $\pi(S) \in \{0, \ldots, \Pi\}$ and whose weight $w(S)$ is minimum.

In our approach, weight is attributed to *nodes* instead of directly to edges; the crucial issue, which affects the design of the recursion, is that the weight of each edge must be properly “charged” on one of its end nodes. Specifically, if both end nodes of an edge $\{i, j\}$ belong to S, we charge w_{ij} on the end node with the smallest index. If a node k is to be brought into S, and its parent $k' \in S$, we make the weight calculated for S increase by a quantity:

$$a_k = \sum_{j \in \Gamma_k} w_{kj}$$

in this case, we consider the weight of $\{k', k\}$ already “charged” on node k'. If $k' \notin S$ and k is brought into the solution, we make the calculated $w(S)$ increase by:

$$b_k = w_{k'k} + a_k$$

as $w_{k'k}$ additional units are to be charged on k. In both cases, $w_{k'k}$ is counted exactly once in $w(S)$.

We formally state the recursion for TMNC by using four functions $\Phi_k(z)$, $\Psi_k(z)$, $\phi_j(z)$ and $\psi_j(z)$. First we define functions $\Phi_k(z)$ for $k \in V$ and $\Psi_k(z)$ for $k \in V \setminus \{1\}$. Let S be a subset of nodes with assigned profit and minimum weight. Then:

(i) $\Phi_k(z)$ is defined as the minimum possible weight for a subset $S \cap T_k$ with $\pi(S \cap T_k) = z$, assuming the parent node $k' \in S$, or $k = 1$;

(ii) $\Psi_k(z)$ is defined as the minimum possible weight for a subset $S \cap T_k$ with $\pi(S \cap T_k) = z$, assuming the parent node $k' \notin S$.

We also need additional functions $\phi_j(z)$ and $\psi_j(z)$, defined for node indices $j \in \Gamma_k$, $k \in V$. Recall that the children nodes $j_k, \ldots, m_k$ are numbered consecutively. Then:

(iii) $\phi_j(z)$ is defined as the minimum possible weight for a subset $S \cap (T_j \cup \ldots \cup T_{m_k})$ with $\pi[S \cap (T_j \cup \ldots \cup T_{m_k})] = z$, assuming the common parent node $k \in S$;

(iv) $\psi_j(z)$ is defined as the minimum possible weight for a subset $S \cap (T_j \cup \ldots \cup T_{m_k})$ with $\pi[S \cap (T_j \cup \ldots \cup T_{m_k})] = z$, assuming the common parent $k \notin S$.

The functions described above can be written recursively as:

$$\Phi_k(z) = \min\{\phi_{j_k}(z - \pi_k) + a_k, \psi_{j_k}(z)\} \qquad k \in V \setminus U \qquad [5.9]$$

$$\Psi_k(z) = \min\{\phi_{j_k}(z - \pi_k) + b_k, \psi_{j_k}(z)\} \qquad k \in V \setminus U, k \neq 1 \qquad [5.10]$$

$$\phi_j(z) = \min_{0 \leqslant u \leqslant z} \{\Phi_j(u) + \phi_{j+1}(z - u)\} \qquad j \in \Gamma_k \setminus \{m_k\}, k \in V \qquad [5.11]$$

$$\phi_{m_k}(z) = \Phi_{m_k}(z) \qquad k \in V \setminus U \qquad [5.12]$$

$$\psi_j(z) = \min_{0 \leqslant u \leqslant z} \{\Psi_j(u) + \psi_{j+1}(z - u)\} \qquad j \in \Gamma_k \setminus \{m_k\}, k \in V \qquad [5.13]$$

$$\psi_{m_k}(z) = \Psi_{m_k}(z) \qquad k \in V \setminus U \qquad [5.14]$$

with initial conditions on leaves $k \in U$:

$$\Phi_k(z) = \begin{cases} a_k = 0 & \text{if } z = \pi_k \\ 0 & \text{if } z = 0 \\ \infty & \text{otherwise} \end{cases} \qquad [5.15]$$

$$\Psi_k(z) = \begin{cases} b_k = w_{k'k} & \text{if } z = \pi_k \\ 0 & \text{if } z = 0 \\ \infty & \text{otherwise} \end{cases} \qquad [5.16]$$

The maximum profit for the instance at hand is given by:

$$\pi(S^*) = \max\{z : \Phi_1(z) \leqslant B\} \qquad [5.17]$$

the optimal solution S^* can be recovered by backtracking.

PROPOSITION 5.3.– *Recursion [5.9]–[5.16] optimally solves TMNC in* $\mathcal{O}(n^3\pi_{\max}^2)$.

Proof. For the correctness of the recursion, we only need to verify that equations [5.9]–[5.16] correctly define functions $\Phi_k(z)$, $\Psi_k(z)$, $\phi_j(z)$, $\psi_j(z)$ accordingly with (i)–(iv). As above, let $S \subseteq V$ be a subset with assigned profit and minimum weight.

For a node $k \in U$, [5.15] and [5.16] trivially give the correct value for $\Phi_k(z)$ and $\Psi_k(z)$. Also, for a rightmost child node $m_k \in \Gamma_k$, $\phi_{m_k}(z) = \Phi_{m_k}(z)$ and $\psi_{m_k}(z) = \Psi_{m_k}(z)$, accordingly with equations [5.12] and [5.14], are easily seen to hold.

If $k \notin U$, it has children nodes $j_k, \ldots, m_k$; the minimum possible weight for $S \cap T_k$, with $\pi(S \cap T_k) = z$ and $k' \in S$ is either, using definitions (i)–(iv):

$\phi_{j_k}(z - \pi_k) + a_k$, if $k \in S \cap T_k$, or
$\psi_{j_k}(z)$, if $k \notin S \cap T_k$;

this establishes equation [5.9]. Similarly for equation [5.10]: the minimum weight for $S \cap T_k$, under the hypotheses $\pi(S \cap T_k) = z$ and $k' \notin S$, is either $\phi_{j_k}(z - \pi_k) + b_k$ if $k \in S \cap T_k$, or $\psi_{j_k}(z)$ if $k \notin S \cap T_k$.

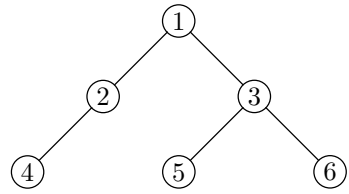
Equation [5.11] deals with a forest $\{T_j, \ldots, T_{m_k}\}$, where $j, \ldots, m_k \in \Gamma_k$ are consecutive children (left to right) of a node k. The minimum weight for $S' = S \cap (T_j \cup \ldots \cup T_{m_k})$ is obtained as follows: since $T_j, \ldots, T_{m_k}$ are separate components of the forest, $w(S') = w(S \cap T_j) + \ldots + w(S \cap T_{m_k})$. Suppose that $\pi(S \cap T_j) = u$; then the minimum weight for S', with $\pi(S \cap T_j) = u$, $\pi(S') = z$ and $k \in S$ is

$$\Phi_j(u) + \phi_{j+1}(z - u)$$

and minimizing over $u \in \{0, \ldots, z\}$ we get equation [5.11]. We proceed similarly for equation [5.13].

Finally, note that each $\Phi_k(z)$ (respectively, $\Psi_k(z)$) and $\phi_j(z)$ (resp. $\psi_j(z)$) only depends on the Φ, Ψ, ϕ and ψ values calculated for all z on nodes with higher index. Hence all the left-hand sides of equations [5.9]–[5.16] can be consistently calculated by scanning the nodes in reverse order $n, n-1, \ldots, 1$.

For the time bound: for each z, there are no more than n values for $\Phi_k(z)$ and $\Psi_k(z)$ respectively to calculate, each computable in constant time by [5.9] or [5.11]; for each $k \in V$ there are exactly $|\Gamma_k|$ pairs of values $\phi_j(z)$ and $\psi_j(z)$ to calculate, each requiring time $\mathcal{O}(z)$. Hence, for a fixed z a time proportional to $n + z \cdot \sum_{k \in V} |\Gamma_k| = \mathcal{O}(n + nz) = \mathcal{O}(nz)$ is required. There are Π relevant values for z, hence the total time is proportional to $n \cdot \sum_{z=1}^{\Pi} z = \mathcal{O}\big(n\Pi^2\big) = \mathcal{O}\big(n^3\pi_{\max}^2\big)$. ∎



	$z=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\Phi_k(z)$	$k=1$	0	2	1	3	5	4	6	5	4	6	**5**	7	9	9	9
	$k=2$	0	∞	∞	3	∞	3	∞	∞	3	∞	∞	∞	∞	∞	∞
	$k=3$	0	2	1	3	3	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=4$	0	∞	∞	**0**	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=5$	0	∞	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=6$	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$z=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\Psi_k(z)$	$k=2$	0	∞	∞	3	∞	4	∞	∞	**4**	∞	∞	∞	∞	∞	∞
	$k=3$	0	2	**1**	3	5	5	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=4$	0	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=5$	0	∞	**1**	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$k=6$	**0**	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$z=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\phi_j(z)$	$j=2$	0	2	1	3	3	3	5	4	3	5	4	6	6	6	∞
	$j=3$	0	2	1	3	3	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$j=4$	0	∞	∞	**0**	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$j=5$	0	0	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$j=6$	0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$z=$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$\psi_j(z)$	$j=2$	0	2	1	3	5	4	6	5	4	6	**5**	7	9	9	∞
	$j=3$	0	2	**1**	3	5	5	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$j=4$	0	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$j=5$	0	2	**1**	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	$j=6$	**0**	2	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

Figure 5.2. *Example of recursion for TMNC (data in the text). The states involved in backtracking are in boldface*

Consider the example illustrated in Figure 5.2, with $V = \{1, \ldots, 6\}$, $\Gamma_1 = \{2, 3\}$, $\Gamma_2 = \{4\}$, $\Gamma_3 = \{5, 6\}$, $B = 5$ and:

$$w_{12} = 1, \quad w_{13} = 2, \quad w_{24} = 3, \quad w_{35} = 1, \quad w_{36} = 2$$
$$\pi_1 = 1, \quad \pi_2 = 5, \quad \pi_3 = 2, \quad \pi_4 = 3, \quad \pi_5 = 2, \quad \pi_6 = 1$$

We have $a_1, \ldots, a_6 = 3, 3, 3, 0, 0, 0$, and $b_2, \ldots, b_6 = 4, 5, 3, 1, 2$. By processing the nodes according to the sequence $(6, 5, 4, 3, 2, 1)$, we calculate the values:

$$\Phi_6(z), \Psi_6(z), \phi_6(z), \psi_6(z), \Phi_5(z), \ldots, \Phi_2(z), \Psi_2(z), \phi_2(z), \psi_2(z), \Phi_1(z)$$

for all $z \in \{0, \ldots, \Pi = 14\}$ respectively. This gives the tables enclosed in Figure 5.2. According to [5.17] the optimal solution has profit $\pi(S^*) = 10$. By backtracking, we

get:

$$
\begin{array}{lll}
\Phi_1(10) = \psi_2(10) = 5 & \rightarrow 1 \notin S^*, \psi_2(10) & = \Psi_2(8) + \psi_3(2) \\
 & & = 4 + 1 = 5 \\
\Psi_2(8) = \phi_4(3) + b_2 = 0 + 4 = 4 & \rightarrow 2 \in S^*, \phi_4(3) & = \Phi_4(3) \\
\Phi_4(3) = a_4 = 0 & \rightarrow 4 \in S^*, \psi_3(2) & = \Psi_3(2) = 1 \\
\Psi_3(2) = \psi_5(2) = 1 & \rightarrow 3 \notin S^*, \psi_5(2) & = \Psi_5(2) + \psi_6(0) \\
 & & = 1 + 0 = 1 \\
\Psi_5(2) = b_5 = 1 & \rightarrow 5 \in S^*, \psi_6(0) & = \Psi(0) \\
\Psi_6(0) = 0 & \rightarrow 6 \notin S^* &
\end{array}
$$

Hence $S^* = \{2, 4, 5\}$.

REMARK 5.2.– We could design the recursion with the total weight as parameter; then $\Phi_k(z)$ would be the maximum profit attainable by a partial solution $S \cap T_k$ having $w(S \cap T_k) \leqslant z$, assuming $k' \in S$ or $k = 1$. The other definitions should be modified similarly. Along the same lines as above, we can prove that such recursion solves TMNC in time $\mathcal{O}\big(nB^2\big)$.

REMARK 5.3.– If node profits and/or edge weights are unitary ($\pi_{\max} = 1$ and/or $B < \frac{n(n-1)}{2}$), the complexity becomes polynomial. This induces, as a byproduct, that DKS on trees is polynomially solvable [COR 84]. We show this for DKS with unitary weights, i.e., the so-called *Densest* K-Subgraph problem in [BIL 05], but through dicothomic search also the case with arbitrary weights, called *Heaviest* K-Subgraph problem in [BIL 05] is polynomially solvable. Indeed, given an instance on a tree T of DKS, it is sufficient to optimally solve the corresponding TMNC instances for all values of $B \in \{1, \ldots, |E|\}$ — call TMNC(B) such instances — and check the corresponding solution values. This can be done in polynomial time as $|E| \leqslant \frac{n(n-1)}{2}$. The value B^* such that its optimal value $\mathrm{OPT}[\mathrm{TMNC}(B^*)] = n - K$ and $\mathrm{OPT}[\mathrm{TMNC}(B^* - 1)] = n - K - 1$, gives the optimal solution value of DKS.

5.3.2. *A fully polynomial time approximation scheme*

Given the dynamic programming recursion [5.9]–[5.16], an FPTAS for MNC on trees is defined as follows. Let I be an instance of TMNC with a tree $T(V, E)$, profits π_i, $i \in V$, weights w_{ij}, $\{i, j\} \in E$ and B; let ε be the desired bound on the relative error, $\pi_{\max} = \max\{\pi_i : i \in V\}$, as previously defined, and $q = \lceil 1/\varepsilon \rceil$, $\bar{M} = \frac{\pi_{\max}}{n(1+q)}$. We construct a scaled instance I' with the same tree, capacity $B' = B$ and weights $w'_{ij} = w_{ij}$, and scaled profits $\pi'_i = \lfloor \pi_i / \bar{M} \rfloor$. Each solution feasible for I' is also feasible for I. Then I' is solved by the $\mathcal{O}\left(n^3 \pi^2_{\max}\right)$ dynamic programming recursion, and the optimal solution S' for I' is returned as heuristic solution for I.

PROPOSITION 5.4.– $\frac{\pi(S^*)}{\pi(S')} \leqslant 1+\varepsilon$, *and* S' *can be calculated in time* $\mathcal{O}\left(\frac{n^5}{\varepsilon^2}\right)$.

Proof. The proof exactly mimics that given for KNAPSACK [GAR 79]. For S^* and S', we have:

$$\sum_{i\in S'} \pi_i \geqslant \sum_{i\in S'} \bar{M}\pi'_i \geqslant \sum_{i\in S^*} \bar{M}\pi'_i \geqslant \sum_{i\in S^*} (\pi_i - \bar{M}) \geqslant \sum_{i\in S^*} \pi_i - n\bar{M}$$

hence, $\pi(S') \geqslant \pi(S^*) - n\bar{M}$. Then:

$$\begin{aligned} \frac{\pi(S^*)}{\pi(S')} &\leqslant 1 + \frac{n\bar{M}}{\pi(S')} \leqslant 1 + \frac{n\bar{M}}{\pi(S^*) - n\bar{M}} \leqslant 1 + \frac{n\bar{M}}{\pi_{\max} - n\bar{M}} = 1 + \frac{1}{q} \\ &\leqslant 1+\varepsilon \end{aligned}$$

The time bound follows from $\pi'_{\max} = \lfloor \pi_{\max}/\bar{M} \rfloor = \lfloor n(1+q) \rfloor$, hence the recursion takes time:

$$\mathcal{O}\left(n^3(\pi'_{\max})^2\right) = \mathcal{O}(n^5 q^2) = \mathcal{O}\left(\frac{n^5}{\varepsilon^2}\right)$$

that completes the proof. ∎

5.4. Exponential algorithms for special cases

In this section we consider exact algorithms for (special cases of) MNC having running time bounded from above by functions like $2^{p(n)}\mathrm{poly}(n)$, where $\mathrm{poly}(n)$ is any polynomial and $p(n)$ grows no faster than $\mathcal{O}(n)$. The interest for the formal analysis of such super-polynomial algorithms has been growing in the last years. Accordingly with [WŒG 03] we adopt the modified big-oh notation by writing $f(n) = \mathcal{O}^*(2^{p(n)})$ if $f(n) = \mathcal{O}(2^{p(n)}\mathrm{poly}(n))$.

The tree case

Consider TMNC formulated on a tree $T(V, E)$ as in section 5.3. We present an algorithm structured in three steps.

Step 1. Problem decomposition. Given the tree T, removing a node $k \in V$ and the set of its incident edges splits T into a forest:

$$F = \{T_1(V_1, E_1), T_2(V_2, E_2), \ldots, T_m(V_m, E_m)\}$$

(m depending on k).

The set of subtrees F can be partitioned into two subforests $F_P(P, E_P)$, $F_Q(Q, E_Q)$.

We assign subtrees to F_P or F_Q in order to get P and Q as balanced as possible, i.e. with number of nodes $|P|$ as close as possible to $n/2$; for each fixed node k, this can be done in $\mathcal{O}(n^2)$ time by dynamic programming. We repeat the latter for all $k \in V$ in order to get the most balanced partition.

Step 1 can be implemented in $\mathcal{O}(n^3)$ time.

Step 2. Enumeration. From now on, consider the "best" $k \in V$ found in Step 1 and the associated P and Q as balanced as possible and assume $|P| \geqslant |Q|$. We enumerate all subsets of P and Q. This step has exponential complexity, bounded from above by $\mathcal{O}^*(2^{|P|})$.

Step 3. Merge. Step 2 gives two lists with $2^{|P|}$ and $2^{|Q|}$ partial solutions. We compose pairs of partial solutions as follows.

> *Step 3a.* First we consider solutions where node k is taken; each partial solution S is given weight:
>
> $$W_S = \sum_{\{i,j\} \in \delta(S) \setminus \delta(\{k\})} w_{ij}$$
>
> and profit $p_S = \pi(S)$. We then choose $\tilde{S}_P \subseteq P$, $\tilde{S}_Q \subseteq Q$ such that $p_{\tilde{S}_P} + p_{\tilde{S}_Q} + \pi_k$ is maximum and $W_{\tilde{S}_P} + W_{\tilde{S}_Q} \leqslant B - w(\{k\})$. By the techniques illustrated in [WŒG 03] and [HOR 74] for the knapsack problem, this can be done in time $\mathcal{O}^*(2^{|P|})$.
>
> *Step 3b.* Then we consider solutions where node k is not taken; we give to each S weight:

$$W_S = w(S) = \sum_{\{i,j\} \in \delta(S)} w_{ij}$$

and profit $p_S = \pi(S)$; we then apply the same techniques to choose $\hat{S}_P \subseteq P$, $\hat{S}_Q \subseteq Q$ with maximum $p_{\hat{S}_P} + p_{\hat{S}_Q}$ and $W_{\hat{S}_P} + W_{\hat{S}_Q} \leqslant B$.

The optimal solution for TMNC is that with the highest profit between:

$$\{k\} \cup \tilde{S}_P \cup \tilde{S}_Q \quad \text{and} \quad \hat{S}_P \cup \hat{S}_Q$$

PROPOSITION 5.5.– *Steps 1-3 solve TMNC in time* $\mathcal{O}^*(2^{\frac{2}{3}n}) \approx \mathcal{O}^*(1.5874^n)$.

Proof. The correctness of the algorithm is trivial to establish. For the time bound, it is sufficient to prove that there exists $k \in V$ for which in Step 1 we can always find a partition $V = P \cup Q$ such that: $\frac{1}{3}n \leqslant |P|, |Q| \leqslant \frac{2}{3}n$.

First recall that for some $k \in V$, T can be partitioned in a forest where each component T_i has $|V_i| \leqslant \frac{n}{2}$: indeed for a given $k \in V$, at most one among $|V_1|, |V_2|, \ldots, |V_m|$ can be $> \frac{n}{2}$. W.l.o.g. assume $|V_1| > \frac{n}{2}$; then let $l \in V_1$ with $(k,l) \in E$ and split the tree T on the new node l. Components $T_2, \ldots, T_m$ will merge into a single component with no more than $\frac{n}{2}$ nodes, and T_1 will split in one or more components with strictly less than $|V_1|$ nodes. We can repeat this until all $|V_i| \leqslant \frac{n}{2}$ in the resulting forest.

Now assume $\frac{n}{2} \geqslant |V_1| \geqslant |V_2| \geqslant \ldots \geqslant |V_m|$. If $|V_1| \geqslant \frac{n}{3}$ we set $P = V_1$, $Q = V_2 \cup \ldots \cup V_m$ and the result is proved, since $|V_1| \leqslant \frac{n}{2} < \frac{2}{3}n$ and $|Q| \leqslant \frac{2}{3}n$. Otherwise, let s be the smallest index such that:

$$\sum_{i=1}^{s} |V_i| \geqslant \frac{n}{3}$$

and set $P = V_1 \cup \ldots \cup V_s$, $Q = V_{s+1} \cup \ldots \cup V_m$. As $|V_s| \leqslant |V_1| < \frac{n}{3}$, $\frac{n}{3} \leqslant |P| \leqslant \frac{2}{3}n$ holds, and the same for Q. ∎

The $\mathcal{O}^*(2^{\frac{2}{3}n})$ time bound is the best we can hope for this algorithm; Figure 5.3 sketches a tree with $n = 3p+1$ nodes where splitting the tree on any node cannot give a partition better than $|P| = \lfloor \frac{2}{3}n \rfloor$, $Q = \lfloor \frac{1}{3}n \rfloor$.

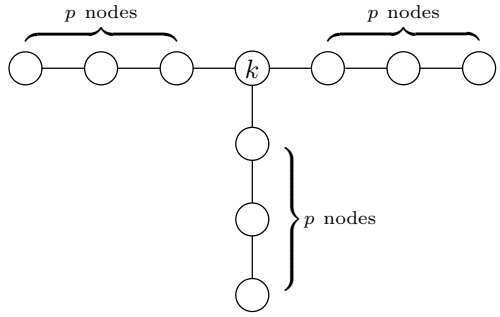


Figure 5.3. *Example of "bad"tree. Here, $n = 3p + 1$; decomposition on node k can give $|P| = \lfloor \frac{2}{3}n \rfloor$, $|Q| = \lfloor \frac{1}{3}n \rfloor$, but no partition is more "balanced"*

Bipartite graphs

Consider MNC formulated on a bipartite graph $G(V_1 \cup V_2, E)$. It is quite straightforward to show that MNC on a bipartite graph can be solved within $\mathcal{O}^*(2^{\frac{3}{4}n})$ running time. Assume $|V_1| \leqslant |V_2|$; enumerating the subsets of V_1 takes $\mathcal{O}^*(2^{|V_1|}) \leqslant \mathcal{O}^*(2^{\frac{n}{2}})$ time. Then we consider each $S_1 \subseteq V_1$ with $w(S_1) \leqslant B$, complete it with the best possible $S_2 \subseteq V_2$, and pick the best $S_1 \cup S_2$. For each S_1, finding the best S_2 amounts to solving an instance I_{S_1} of KNAPSACK where:

$$\begin{aligned} N &= \{i : i \in V_2\} \\ b &= B - w(S_1) \\ p_i &= \pi_i, \quad i \in V_2 \\ W_i &= \sum_{\substack{j:\{i,j\}\in E \\ j\in V_1 \setminus S_1}} w_{ij} \quad i \in V_2 \end{aligned}$$

From [HOR 74] it is known that each I_{S_1} can be solved in time $\mathcal{O}^*(2^{\frac{|V_2|}{2}})$. The overall procedure takes time $\mathcal{O}^*(2^{|V_1|} \cdot 2^{\frac{|V_2|}{2}})$ where the worst case scenario occurs for $|V_1| = |V_2| = \frac{n}{2}$, i.e., $\mathcal{O}^*(2^{\frac{3}{4}n}) \approx \mathcal{O}^*(1.6818^n)$.

5.5. Bibliography

[BIL 05] BILLIONNET A., ROUPIN F., "A deterministic approximation algorithm for the densest k-subgraph problem", *International Journal of Operational Research*, To appear. A draft of this paper appears as Technical Report 486 of CEDRIC, CNAM, Paris.

[CAR 04] CARELLO G., "Hub location problems in telecommunication networks"', Ph.D. Thesis, Politecnico di Torino, available at `http://www.orgroup.polito.it/members/carello/phdthesis.pdf`, 2004.

[COR 84] CORNEIL D. G., PERL Y., "Clustering and domination in perfect graphs", *Discrete Applied Mathematics*, vol. 9, p. 27–39, 1984.

[FEI 01] FEIGE U., KORTSARZ G., PELEG D., "The dense k-subgraph problem", *Algorithmica*, vol. 29, p. 410–421, 2001.

[GAR 79] GAREY M., JOHNSON D. S., *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[HOR 74] HOROWITZ E., SAHNI S., "Computing partitions with applications to the knapsack problem", *Journal of the ACM*, vol. 21, p. 277–292, 1974.

[MAR 90] MARTELLO S., TOTH P., *Knapsack Problems - Algorithms and Computer Implementations*, Wiley, New York, 1990.

[WŒG 03] WŒGINGER G. J., "Exact algorithms for NP-hard problems: a survey", JUENGER M., REINELT G., RINALDI G., Eds., *Combinatorial Optimization - Eureka! You Shrink!*, vol. 2570 of *Lecture Notes in Computer Science*, p. 185–207, Springer, 2003.

[YAM 05] YAMAN H., *Concentrator Location in Telecommunication Networks*, Springer, New York, 2005.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 6

The Patrolling Problem: Theoretical and Experimental Results

6.1. Introduction

A group of agents can be used to perform patrolling tasks in a variety of domains ranging from computer network administration to computer wargame simulations. The multi-agent patrolling problem has recently received growing attention from the multi-agent community, due to the wide range of potential applications. Many algorithms based on reactive and cognitive architectures have been developed, giving encouraging results. However, no theoretical analysis of this problem has been conducted. In this chapter, various classes of patrolling strategies are proposed and compared. The most important class of strategies presented here is the class of *cyclic strategies*, which is shown to have many nice theoretical properties. Also, various experiments show that this class yields better results than all previous approaches to the patrolling problem.

To patrol is literally the act of walking or traveling around an area, at regular intervals, in order to protect or supervise it. This task is by nature a multi-agent task and there are a wide variety of problems that can be reformulated as particular patrolling tasks. As a concrete example, during the development of the Artificial Intelligent component of an interactive computer wargame, we may face the problem of coordinating a group of units to patrol a given rough terrain in order to detect the presence of "enemies". The quality of the strategies used for patrolling may be evaluated using different measures. Informally, a good strategy is one that minimizes the time lag between two passages to the same place and for all places. Beyond simulators and computer

Chapter written by Yann CHEVALEYRE.

games, performing this patrolling task efficiently can be useful for various application domains where distributed surveillance, inspection or control are required. An example of such a task is the identification of objects or people in dangerous situations that should be rescued by robots [RES 07].

Recently, many different architectures of multi-agent systems have been proposed and evaluated on the patrolling problem [MAC 00], giving encouraging results. In particular, it was shown in [MAC 00] that very simple strategies implemented through reactive agents with nearly no communication ability could achieve impressive results. Also, some authors of these papers suggested that an approach based on partitioning the territory such that each agent patrols in its own region could also work well.

This chapter proposes a theoretical analysis of the patrolling problem addressing the following issues: do the existing algorithms generate optimal strategies? Are there efficient algorithms generating near-optimal strategies? Are patrolling strategies based on partitioning the territory good? What if the agents all follow the same circuit one behind another? To answer these questions, strategies are proposed in this paper which are close to strategies humans could build to patrol over a territory. They need a certain amount of communication/synchronization between agents. Which multi-agent architecture should be chosen and how to implement the algorithms will not be our purpose here (see [SEM 04]). Our primary purpose is to understand more deeply the patrolling problem.

To carry out this study, we will use the formal definition of the patrolling problem introduced in [MAC 00], which represents territories as graphs. A first class of strategies referred to as "cyclic strategies" is introduced in section 6.3, and it is shown that there exists an $O(n^3)$ algorithm generating such a strategy close to optimality. In section 6.4, strategies based on partitioning will be analyzed and compared to the previous ones. In section 6.5, the results obtained in [MAC 00] will be compared to ours. Finally, section 6.6 draws some conclusions and indicates directions for future work.

6.2. The patrolling task

Many tasks (such as rescuing, tracking, detecting, etc.) can involve some sort of patrolling, which may exhibit slightly different characteristics according to the domain and circumstances. It is then necessary for our study to have a more precise definition of patrolling.

In [MAC 00], it was shown that in many applications of the patrolling problem, the territory could be represented by an undirected graph. Given such a graph, the patrolling task refers to continuously visiting all the graph nodes so as to minimize the

time lag between two visits. The edges may have different associated lengths (weights) corresponding to the real distance between the nodes.

From now on, the graph representing the territory will be referred to as $G(V, E)$, where $V = \{1 \dots n\}$ is the set of nodes and $E \subseteq V^2$ the set of edges of G. To each edge (i, j) will correspond a weight c_{ij} representing the distance between nodes. Note that the graph will be assumed to be metric[1]. The time taken by an agent to move across an edge (i, j) will be exactly c_{ij}. At time 0, r agents will be positioned on the nodes of G. When the patrolling task starts, agents will move simultaneously around the nodes and edges of the graph according to a predetermined strategy.

DEFINITION 6.1.– *The strategy of an agent is a function $\pi : \mathbb{N} \rightarrow V$ such that $\pi(j)$ is the jth node visited by the agent. A multi-agent strategy $\Pi = \{\pi_1 ... \pi_r\}$is simply defined as a set of r single-agent strategies.*

Knowing the strategy of agent k, we can now predict at what time a given node will be visited. For example, agent k will visit node $\pi_k(0)$ at time 0. It will also visit node $\pi_k(j)$ at a time equal to the weight of the path $\pi_k(0) \dots \pi_k(j)$, which is $\sum_{i=0}^{j-1} c_{\pi_k(i)\pi_k(i+1)}$. For the sake of clarity, the weight of a path $s_0 ... s_m$ will be from now on noted by $c(s_0 ... s_m)$. Also, the weight of a set of edges E' will be noted by $c(E')$.

Our main goal is to find good patrolling strategies. We thus need an evaluation criterion. We will use *idleness* criteria introduced in [MAC 00].

DEFINITION 6.2.– *Let r agents patrol a graph G according to a multi-agent strategy Π. The idleness of a node i at time t is the amount of time elapsed since that node has received the visit of an agent. The idleness of all nodes at the beginning of the patrolling task is set to 0. The worst idleness is the biggest value of the idleness which occurred during the entire patrolling process for all nodes. It is noted by $WI_\Pi(G)$ or just WI_Π when there is no ambiguity.*

Figure 6.1 illustrates the calculation of the idleness and worst idleness for a single agent and a very simple graph. Notice that strategies are not necessarily finite over time and that agents can patrol forever on the graph. Of course, if strategies are tested in a simulator, the patrolling task will have to be stopped at once. Thus, the worst idleness measured during a simulation will be an approximation of the true value.

In this chapter, we chose the worst idleness criterion among the various criteria defined in [MAC 00] mainly for simplicity reasons: compared to the average idleness

1. Graphs in which the triangular inequality is not violated, i.e., given three nodes i, j, k connected by edges, we have $c_{ij} + c_{jk} \geqslant c_{ik}$

criterion, which is the other main criterion defined in [MAC 00], the theoretical analysis will be easier than with our criterion. In addition, the former is upper bounded by the latter. Thus, minimizing worst idleness will also lead to a small average idleness. More formally, the problem can be stated the following way:

DEFINITION 6.3.– *Given a connected graph $G = (V, E)$ and a number r of agents, the patrolling problem consists of finding a multi-agent strategy $\Pi = \{\pi_1 \dots \pi_r\}$ which minimizes $WI_\Pi(G)$.*

6.3. Previous work

Most work on the patrolling problem as formulated above has been done by Machado *et al.* in [MAC 00]. In their article, they proposed several multi-agent architectures varying parameters such as agent type (reactive vs. cognitive), agent communication (allowed vs. forbidden), coordination scheme (central and explicit vs. emergent), agent perception (local vs. global), decision-making (random selection vs. goal-oriented selection). For each agent, the choice of the next node to visit is basically influenced by two factors: (1) node idleness, which can be shared (generated by all agents) vs. individual (corresponding to a single agent visits); (2) field of vision, which can be local (agent's neighborhood) or global (entire graph). The experiments they

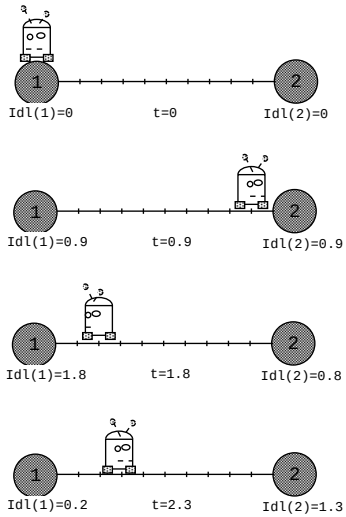


Figure 6.1. *Agent patrolling on a graph made of two nodes and an edge of weight 1. Its strategy is $\pi = 1, 2, 1, 2, 1, \dots$ The idleness of the nodes is shown at various moments. Here, $WI_\pi = 2$*

conducted showed two interesting facts: first, agents moving randomly achieved very bad results. Second, agents with absolutely no communication ability whose strategies consisted of moving towards the node with the highest idleness performed nearly as well as the most complex algorithm they implemented. In the experiments section, we will compare our approach to the two most efficient strategies of [MAC 00]: (a) the conscientious reactive strategy, in which the next node an agent visits is that with the highest individual idleness from its local neighborhood; (b) the cognitive coordinated strategy, in which the next node an agent chooses to visit is that with the highest idleness from the whole graph, according to the suggestions given by a central coordinator agent. To reach the chosen node, agents move through the shortest path leading to this node. The coordinator is responsible for avoiding that more than one agent chooses the same next node. More recently, a new approach based on reinforcement learning (RL) techniques was developped in [SAN 04]. In their paper, Santana *et al.* showed that a simple Q-learning algorithm could be used to train agents to solve the patrolling problem efficiently. The results of their algorithm will also be compared to ours in the experiments section.

The strategies proposed in this paper are very different from those proposed in previous work. They have not been developped out of a specific architecture, or a communication scheme between agents. In fact, the implementation issues of our strategies will not be tackled here. However, important work has been done concerning the implementation of the cyclic strategies in a multi-robot environment [SEM 04].

The next section proposes a new class of strategies which we called “cyclic strategies”.

6.4. The cyclic strategies

In this section, we will first show how cycles and closed paths can be used to create efficient single-agent patrolling strategies. Then, an extension to the multi-agent case will be proposed, and the resulting strategies will be shown to be near-optimal.

6.4.1. *Patrolling with a single-agent*

Consider a single agent patrolling over an area. The simplest strategy which comes to mind would be to find a cycle covering all the area, and then to make the agent travel around this cycle over and over. Applied to our case in which areas are represented by nodes in a graph, the notion of cycle is too restrictive. In fact, in the graph-theory terminology, a cycle is a path starting and ending on the same node and covering each edge at most once. However, for some graphs such as that in Figure 6.2, there is not a cycle covering all nodes. Instead of using cycles, we will have to use closed

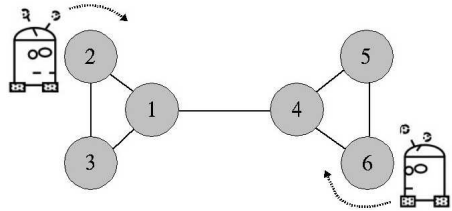


Figure 6.2. *Example of multi-agent cyclic-based strategy. The strategies of these agents are* $\pi_1 = 2, 1, 4, 5, 6, 4, 1, 3, 2, 1, 4 \ldots$ *as well as* $\pi_2 = 6, 4, 1, 3, 2, 1, 4, 5, 6, \ldots$

paths, which are paths starting and ending on the same node and covering edges possibly more than once. A closed path is usually represented by a list of nodes, beginning and ending with the same node. In Figure 6.2, for example, the closed path $s = 1, 3, 2, 1, 4, 5, 6, 4, 1$ covers all nodes, and turning over s indefinitely seems to be a good strategy for a single-patrolling problem. Single-agent strategies consisting of traveling along a closed path indefinitely will be called *single-agent cyclic strategies.* The strategy of the agent in Figure 6.1 is a single-agent cyclic strategy based on the closed path $1, 2, 1$. The bottom agent on the right of Figure 6.3 is also following a cyclic strategy based on the closed path $4, 5, 6, 5, 4$.

Before extending this idea to the multi-agent case, let us concentrate on two questions: 1) which closed path should be chosen in a single-agent patrolling problem? 2) are strategies based on closed paths optimal with a single-agent? To answer the first question, let us note that the time taken for a single agent patrolling around a closed path to visit a node twice is at most equal to the length of this closed path. Therefore, with a single agent patrolling around a closed path s, the worst idleness will be equal to the length of s. Finding the smallest closed path covering all nodes will thus result in the best possible strategy *among all single-agent cyclic strategies.* Let us show how this problem relates to the well known *Traveling Salesman Problem (TSP).*

The traveling salesman problem is a combinatorial optimization problem which was originally formulated as follows: given a set of cities on a map, find the shortest cycle such that each city is visited *exactly once*. This problem was soon extended to metric graphs and became: given a metric graph $G(V, E)$ with edges weighted according to c_{ij}, find the shortest closed path such that each node is visited *at most once* [REI 94]. Christofides [CHR 76] proposed an algorithm which generates in $O(n^3)$ a cycle (or a closed path for the metric graphs case) whose length is less than $\frac{3}{2}$ times the shortest cycle (or closed path). From now on, S_{TSP} will denote the closed path being the optimal solution to the TSP, whereas S_{Chr} will denote the closed path obtained by the algorithm of Christofides. The following holds (for which we can find a formal proof in [CHE 05]).

THEOREM 6.1.– *For a single agent, the optimal strategy in terms of worst idleness is the cyclic-based strategy based on S_{TSP}.*

We already knew that the cyclic strategy based on S_{TSP} was the best possible strategy *among all single-agent cyclic strategies*. Note that this theorem states that it is also the best strategy *among all possible single-agent strategies*. Note that this result will only be needed to demonstrate Theorem 6.2. An immediate corrolary of this theorem is that the worst idleness of a single-agent cyclic strategy based on S_{Chr} will be at most $\frac{3}{2}$ times the worst idleness of the optimal strategy. In addition, the S_{Chr} closed path is calculable in $O(n^3)$, whereas the calculation of S_{TSP} is NP-complete. In practice, there exists many efficient algorithms which can efficiently build closed paths very close to S_{TSP} using heuristic algorithms.

6.4.2. *Extending to multi-agent case*

One way to extend single-agent cyclic strategies to the multi-agent case is to arrange agents on the same closed path such that when they start moving through that path all in the same direction, they keep an approximately constant gap between them. This leads to the following definition:

DEFINITION 6.4.– *Let $S = s_0...s_m$ be a closed path visiting all nodes of a graph G. The strategy $\Pi = \{\pi_1...\pi_r\}$ is a multi-agent cyclic strategy based on S if and only if there are $d_1...d_r \in \mathbb{N}$ such that $\pi_i(k) = s_{(k+d_i) \bmod m}$. The set of all multi-agent cyclic strategies will be referred to as $\mathbf{\Pi_{cyclic}}$.*

Figure 6.2 illustrates this with a 2-agent cyclic strategy based on the closed path $2, 1, 4, 5, 6, 4, 1, 3, 2$, such that $d_1 = 0$ and $d_2 = 4$.

How does the worst idleness evolve when the number of agents grows? The following lemma shows that if r agents follow a multi-agent cyclic strategy Π and if $d_1...d_r$ are well chosen, then the worst idleness will be approximatively r times lower than the worst idleness obtained by a single agent patrolling around the same closed path.

LEMMA 6.1.– *Let $S = s_0 \dots s_m$ be a closed path covering each node of G such that there exists a node $x \in V$ covered exactly once by S. Let $l = c(S)$ be the length of the closed path. There exists a multi-agent cyclic strategy $\Pi = \{\pi_1 \dots \pi_r\}$ based on this closed path such that $\frac{l}{r} - \max\{c_{ij} \mid (i,j) \in E_S\} \leqslant WI_\Pi \leqslant \frac{l}{r} + \max\{c_{ij} \mid (i,j) \in E_S\}$.*

Here, E_S refers to the set of edges present in S. Note that l is also the worst idleness of the single-agent cyclic strategy based on S.

Proof. Let $S = s_0 ... s_m$, be the closed path covering G. Consider two agents moving around S such that at time 0, agent 1 is positioned on node s_0 and agent 2 is on node s_d. Let $l(i,j) = \sum_{k=i}^{j-1} c_{s_k s_{k+1}}$. Thus, $l(0,d)$ is the time taken by agent 1 starting at s_0 to reach node s_d. Note that at any time t, the node visited by agent 2 will also be visited by agent 1 at time $t + l(0,d)$. Of course, a node visited at time t by agent 1 will be visited by agent 2 at time $t + l(d,m)$. Therefore, we have $WI_{\{\pi_1,\pi_2\}} \leqslant \max\{l(0,d), l(d,m)\}$. In addition, if agent 2 visits node x at time t_x, because node x is present only once in S, the next visit of the agent to node x will occur exactly at $t_x + l(0,d)$. Thus, $WI_{\{\pi_1,\pi_2\}} = \max\{l(0,d), l(d,m)\}$. ∎

Let us generalize this to r agents. We get $WI_\Pi = \max\{l(0,d_1), l(d_1,d_2), \ldots, l(d_{r-1},m)\}$. We will now have to choose the values of d_k such that the worst idleness is as low as possible. By setting each d_k to the greatest integer verifying $l(0,d_k) \leqslant k \times \frac{l(0,m)}{r}$, we get $d_0 = 0$ and $d_r = m$. We can now write $WI_\Pi = \max_{k=0..r-1} l(d_k, d_{k+1})$. Let us now calculate upper and lower bounds to $l(d_k,d_{k+1}) = l(0,d_{k+1}) - l(0,d_k)$ by showing that $k \times \frac{l(0,m)}{r} - \max\{c_{ij} \mid (i,j) \in E_S\} \leqslant l(0,d_k) \leqslant k \times \frac{l(0,m)}{r}$, thus $(k+1) \times \frac{l(0,m)}{r} - \max\{c_{ij} \mid (i,j) \in E_S\} \leqslant l(0,d_{k+1}) \leqslant (k+1) \times \frac{l(0,m)}{r}$. Combining these two equations, we get for all k: $\frac{l}{r} - \max\{c_{ij} \mid (i,j) \in E_S\} \leqslant l(d_k,d_{k+1}) \leqslant \frac{l}{r} + \max\{c_{ij} \mid (i,j) \in E_S\}$. Because $WI_\Pi = \max_{k=0..m-1} l(d_k,d_{k+1})$, the lemma follows.

6.4.3. *Optimality of cyclic strategies*

We have shown previously that cyclic strategies based on S_{TSP} were optimal for single agents. We will now show a similar result for the multi-agent case. From now on, *opt* will refer to the worst idleness of the optimal strategy.

THEOREM 6.2.– *Let G=(V,E) be a connected metric graph and let r be the agents patrolling on it. Let Π_{Chr} be the multi-agent cyclic strategy based on S_{Chr}. We have $WI_{\Pi_{Chr}} \leqslant 3 \times opt + 4 \times \max_{ij}\{c_{ij}\}$.*

Note that if all edges of G have the same length, then $opt \geqslant \max_{ij}\{c_{ij}\}$ and therefore, $WI_{\Pi_{Chr}} \leqslant 7 \times opt$. To prove this theorem, we first need to demonstrate the following lemma.

LEMMA 6.2.– *For any multi-agent strategy Π, there exists a multi-agent strategy $\Pi' = \{\pi'_1, ..., \pi'_r\}$ such that, for each $k \in 1..r$, each strategy π'_k consists of moving through a path $s^k_1 \ldots s^k_{m_k}$ of m_k nodes forwards and backwards indefinitely and such that $WI_{\Pi'} \leqslant 2 \max_k c(s^k_0 \ldots s^k_{m_k}) \leqslant 2 \times WI_\Pi$.*

Proof. Consider the multi-agent strategy $\Pi = \{\pi_1 ... \pi_k\}$. Let $s^k_1 \ldots s^k_{m_k}$ be the list of nodes visited by agent k according to strategy π_k between time $t = 0$ and time $t =$

WI_Π. Clearly, $\cup_{k=1..r}\{s_1^k,\ldots,s_{m_k}^k\} = V$. Let $\pi'_k = s_1^k \ldots s_{m_k}^k, s_{m_k-1}^k, \ldots, s_1^k, \ldots$ be the strategy consisting of moving forward and backwards through $s_1^k \ldots s_{m_k}^k$. Time taken by agent k using strategy π'_k to visit a node twice is at most $2 \times c(s_1^k \ldots s_{m_k}^k)$. Thus, $WI_{\Pi'}(G) \leqslant 2 \max_k c(s_1^k \ldots s_{m_k}^k) \leqslant 2 \times WI_\Pi(G)$. ∎

Now the theorem can be proven. The preceding lemma will be used on the optimal strategy to build a set of paths, which will be combined to form a closed path leading to a near-optimal strategy.

Proof of the theorem. Consider an optimal strategy Π. We can tell with Lemma 6.2 that there exists a strategy $\Pi' = \{\pi'_1, ..., \pi'_r\}$ in which each strategy π'_k consists of moving through a path $s_1^k \ldots s_{m_k}^k$ and such that $\frac{WI_{\Pi'}}{2} \leqslant \max_k\{c(s_0^k ... s_{m_k}^k)\} \leqslant opt$. Let us first show that there exists a closed path S covering all nodes and such that $\frac{c(S)}{2r} \leqslant \max_k\{c(s_0^k ... s_{m_k}^k)\} + \max_{ij}\{c_{ij}\}$. ∎

Let U be the set of edges present in these paths; because these paths cover all nodes of G, by adding at most $r-1$ edges to U, we obtain a set U' of edges such that the graph (V, U') is connected. Clearly, there exists a tree $T \subseteq U$ covering all nodes. In addition, $c(T) \leqslant c(U') \leqslant \sum_{k=1}^{r} c(s_0^k ... s_{m_k}^k) + (r-1) \times \max_{ij}\{c_{ij}\}$. There exists a closed path S covering all nodes obtained by exploring the tree T in a depth-first manner (thus twice for each edge), such that $\frac{c(S)}{2r} = \frac{c(T)}{r} \leqslant \max_k\{c(s_0^k \ldots s_{m_k}^k)\} + \max_{ij}\{c_{ij}\}$. Let us recall that the closed path S_{Chr} obtained by Christofides algorithm is at most $\frac{3}{2}$ times longer than the shortest closed path covering G. Thus, we have: $c(S_{Chr}) \leqslant \frac{3}{2}c(S) \leqslant 3r \times \max_k\{c(s_0^k \ldots c_{m_k}^k)\} + 3r \times \max_{ij}\{c_{ij}\} \leqslant 3r \times opt + 3r \times \max_{ij}\{c_{ij}\}$.

With Lemma 6.1, we can generate from S_{Chr} a multi-agent cyclic-based strategy Π_{Chr} such that $WI_{\Pi_{Chr}} \leqslant \frac{c(S_{Chr})}{r} + \max_{ij}\{c_{ij}\} \leqslant 3.opt + 4. \max_{ij}\{c_{ij}\}$.

It is clear that because of the $\max_{ij}\{c_{ij}\}$ in the theorem, cyclic strategies will not be suited for graphs containing long edges. For this reason, let us study different strategies, which we will call *partition-based strategies.*

6.5. Partition-based strategies

Another very intuitive way to make r agents patrol over a territory would be to make a partition of this territory into r disjoint regions and to have each agent patrolling inside a single region.

From now on $P = \{P_1 ... P_r\}$ will denote a partition of V. Thus, $P_1 \cup \ldots \cup P_r = V$ and $P_i \cap P_j = \emptyset$. Also, $\{G_1 \ldots G_r\}$ will refer to the subgraphs induced by the partition. Thus, $G_i = (V \cap P_i, E \cap (P_i \times P_i))$.

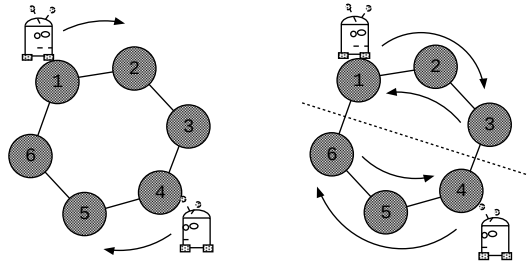


Figure 6.3. *On the left: cyclic strategy* $\Pi_{cyc} = \{\pi_{cyc1}, \pi_{cyc2}\}$ *and to the right: partition based strategy* $\Pi_P = \{\pi_{P_1}, \pi_{P_2}\}$ *with two agents. We have* $\pi_{cyc1} = 1, 2, 3, 4, 5, 6, 1, 2, 3 \ldots$ *and* $\pi_{cyc2} = 4, 5, 6, 1, 2, 3, 4, 5, 6, 1 \ldots$. *Also,* $\pi_{P_1} = 1, 2, 3, 2, 1, 2, \ldots$ *and* $\pi_{P_2} = 4, 5, 6, 5, 4, 5, \ldots$. *Thus,* $WI_{\Pi_{cyc}} = 3$ *and* $WI_{\Pi_P} = 4$

DEFINITION 6.5.– *A multi-agent strategy* $\Pi = \{\pi_1 \ldots \pi_r\}$ *is said to be based on a partition* P *iff each agent* k *following strategy* π_k *visits the nodes of a single region of* P. *The class of all strategies based on partition* P *is referred to as* $\mathbf{\Pi_P}$.

The previous definition does not specify what moves agents should make in their own region. Given a partition P, how should agents behave inside their region? The following lemma brings an answer.

LEMMA 6.3.– *For* $k \in 1..r$, *let* π_k *be the single-agent cyclic strategy based on the TSP of* G_k. *Then,* $\Pi = \{\pi_1 ... \pi_r\}$ *is the optimal strategy based on partition* P.

Proof. For any strategy $\Pi = \{\pi_1 ... \pi_r\}$ based on partition P, each node of G will not be visited by more than one agent. Thus, $WI_\Pi(G) = \max_k\{WI_{\pi_k}(G_k)\}$. Therefore, a set of r optimal single-agent strategies in $P_1 ... P_r$ is an optimal multi-agent partition-based strategy. Theorem 6.1 stated that single-agent cyclic strategies based on TSP were optimal. Thus, by combining such single-agent strategies, we will obtain an optimal partition-based strategy. ∎

Figure 6.3 illustrates how both strategies perform on a circular graph with two agents. In this example, the cyclic strategy wins. Consider now Figure 6.2. On this figure, if the distance between node 1 and 4 was huge, the cyclic strategy would be disastrous, as both agents would spend most time crossing the edge $(1, 4)$. However, a strategy based on the partition $\{\{1, 2, 3\}, \{4, 5, 6\}\}$ would not have this problem. Thus, it seems that on graphs having "long corridors" connecting two subgraphs as on Figure 6.2, partition-based strategies could be better.

The following theorem compares the values of the worst idleness of the optimal cyclic strategy and the optimal partition-based strategy. Here $opt_{\Pi_{Cycle}}$ and opt_{Π_P}

refer to the worst idleness of the optimal cyclic strategy and of the strategies based on partition P. The complete proof of this theorem can be found in [CHE 05].

THEOREM 6.3.– $opt_{\mathbf{\Pi_{cycle}}} \leqslant opt_{\mathbf{\Pi_P}} + 3 \times \max_{ij}\{c_{ij}\}$.

Sketch of proof. The previous lemma showed that the optimal partition-based strategy was composed of r single-agent cyclic strategies based on TSP. Let S^k_{TSP} denote the closed path being the solution to the TSP on subgraph G_k. By joining together these closed paths, it is possible to build a closed path S covering all nodes such that $c(S) \leqslant \sum_{k=1}^{r} c(S^k_{TSP}) + 2r \times \max_{ij}\{c_{ij}\}$. Therefore, we have $c(S) \leqslant \sum_{k=1}^{r} c(S^k_{TSP}) + 2r \times max_{ij}\{c_{ij}\} \leqslant r \times max_{k\in 1..r}\{c(S^k_{TSP})\} + 2r \times max_{ij}\{c_{ij}\} \leqslant r \times opt_{\mathbf{\Pi_P}} + 2r \times max_{ij}\{c_{ij}\}$. From closed path S, we can build a multi-agent cyclic strategy Π_S using Lemma 6.1. $WI_{\Pi_S} \leqslant \frac{c(S)}{r} + \max_{ij}\{c_{ij}\} \leqslant opt_{\mathbf{\Pi_P}} + 3 \times \max_{ij}\{c_{ij}\}$. Thus, given any partition P, there exists a cyclic strategy Π_S verifying the previous equation. ∎

To conclude this section, we can say that cyclic strategies are to be preferred when graphs do not have long edges connecting far regions. Otherwise, building a partition of the graph and making agents follow cyclic strategies based on the TSP of the regions is a good solution. The following corollary is a "computable" version of the main theorem.

COROLLARY 6.1.– Let $P = \{P_1, ..., P_r\}$ be a partition of V. It is possible to calculate in $O(n^3)$ a cyclic strategy Π_{Chr} such that $WI_{\Pi_{Chr}} \leqslant \frac{3}{2} opt_{\mathbf{\Pi_P}} + 4 \times \max_{ij}\{c_{ij}\}$.

6.6. Experiments

Six different graphs (Figure 6.4) were proposed in [DEL 04] as a benchmark for the patrolling problem. To evaluate the cyclic strategy on these graphs, the TSP of each graph was calculated using the open-source software "Concorde"[2] which contains efficient heuristic algorithms which are in practice much faster than the $O(n^3)$ Christofides algorithm [CHR 76], and often come much closer to the optimum. Then, the simulator described in [MAC 00] was used to measure the idleness of agents patrolling around the obtained closed paths.

The graphs in Figures 6.5 and 6.6 show the performance of the strategies described in [MAC 00] and in [SAN 04] on the six graphs for 5 and 15 agents per graph. On each graph, the cyclic strategy obtains the best results. When the number of agents increases largely above 15, all strategies become equivalent.

2. Available at `www.math.princeton.edu/tsp/concorde.html`.

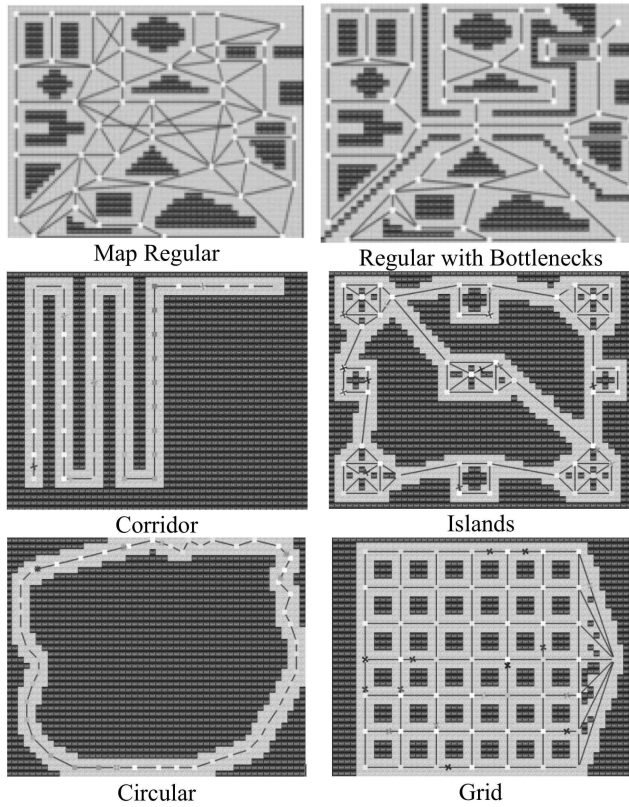


Figure 6.4. *Graphs used during experiments*

6.7. Conclusion

We have shown various theoretical results for the patrolling problem. First, we have shown that the single-agent patrolling problem could be solved with a TSP approach. Then, we defined the class of cyclic strategies, based on an extension of this approach to more than one agent. An approximation result was obtained for this class, showing that in $O(n^3)$, a strategy close to the optimum could be obtained.

The strategies based on a partitioning of the graph were also studied. A surprising result was obtained: except when $\max_{ij}\{c_{ij}\}$ is big, the cyclic strategies based on TSP were shown to be better than any partition-based strategy. However, when graphs

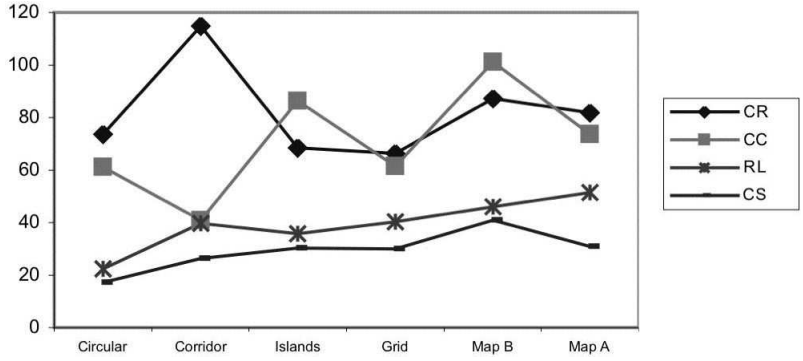


Figure 6.5. *Idleness on different graphs with 5 agents. The strategies compared here are the Conscientious Reactive (CR), the Cognitive Coordinated (CC), the reinforcement learning (RL) and the cyclic strategy (CS)*

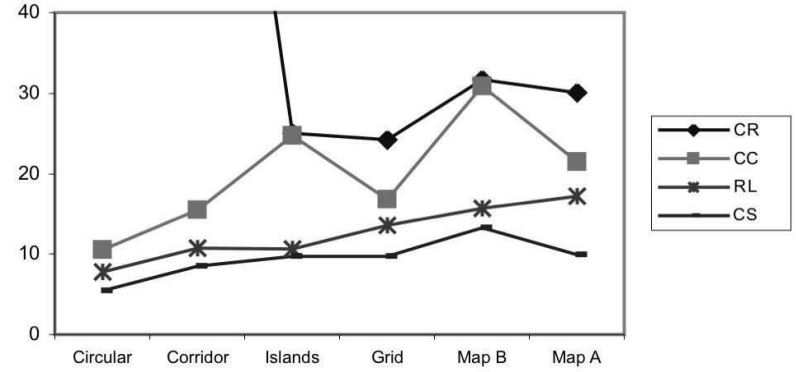


Figure 6.6. *Idleness on different graphs with 15 agents*

have long "tunnels" separating regions, the cyclic strategies are not well suited. Finally, some experiments were conducted to compare the state-of-the-art patrolling algorithms to the cyclic strategy based on TSP. The results are encouraging, but would need to be pursued on more graphs.

Many other interesting theoretical results have been obtained, in particular concerning other kinds of strategies offering a good compromise between cyclic strategies and partition-based strategies.

The cyclic strategy based on TSP has already been implemented in a multi-robot patrolling problem with a noisy environment [SEM 04]. Many modifications had to be brought to the algorithm in order to synchronize the robots. It was shown that the cyclic strategy still performs better when the noise level is low.

6.8. Bibliography

[CHE 05] CHEVALEYRE Y., Le problème de la patrouille, Annales du LAMSADE 4-5, Paris-Dauphine University, 2005.

[CHR 76] CHRISTOFIDES N., Worst-case analysis of a new heuristic for the traveling salesman problem, Report num. CS-93-13, Carnegie Mellon University, Graduate School of Industrial Administration, 1976.

[DEL 04] DE LUNA ALMEIDA A., RAMALHO G., SANTANA H., TEDESCO P., MENEZES T., CORRUBLE V., CHEVALEYRE Y., "Recent advances on multi-agent patrolling", *17th Brazilian Symposium on Artificial Intelligence*, Sao Luis, Brazil, p. 474–483, 2004.

[MAC 00] MACHADO A., RAMALHO G., ZUCKER J., DROGOUL A., "Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures", *Third International Workshop on Multi-Agent Based Simulation*, p. 155–170, 2000.

[REI 94] REINELT G., *The Traveling Salesman: Computational Solutions for TSP Applications*, vol. 840 of *Lectures Notes in Computer Science*, Springer, 1994.

[RES 07] ROBOCUP RESCUE, "RoboCup Rescue Homepage", 2007, `http://www.rescuesystem.org/robocuprescue/`.

[SAN 04] SANTANA H., CORRUBLE V., RATITCH B., "Multi-Agent Patrolling with Reinforcement Learning", *Third International Joint Conference on Autonomous Agents and Multiagent Systems*, vol. 3, p. 1122–1129, 2004.

[SEM 04] SEMPE F., Self-organizing groups of robots: application to patrolling tasks in noisy environments, PhD thesis, University of Paris 6, 2004.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 7

Restricted Classes of Utility Functions for Simple Negotiation Schemes: Sufficiency, Necessity and Maximality

7.1. Introduction

The problem of allocating a set of indivisible resources to a set of self-interested agents has recently received much attention from the Artificial Intelligence community [CHE 06, KRA 01, ROS 94] and has important applications ranging from electronic commerce and industrial procurement [GIO 04], over the joint exploitation of Earth observation satellites [LEM 99], to Grid computing [GAL 05]. Much work in this area has focused on combinatorial auctions [CRA 06]. In this case, the allocation procedure is *centralized* and the so-called winner determination problem is the problem of determining the allocation of resources maximizing the auctioneer's revenue.

A different perspective is taken when we assume that the allocation process is truly *distributed*, in the sense that agents autonomously negotiate over the bundles of resources they hold and that allocations emerge over time, as a consequence of a sequence of local deals agreed upon by the agents in the system. This assumption is justified in many applications where no central authority can be relied upon to decide on the allocation of resources. This may, for instance, be the case in view of calculational limitations of any potential center, or in view of its trustworthiness. In this case, the system designer will typically seek to set up the system in such way that

Chapter written by Yann CHEVALEYRE, Ulle ENDRISS and Nicolas MAUDET.

it nevertheless guarantees certain desirable properties, but without directly interfering in the negotiation process itself. Similar considerations have led to research areas such as distributed constraint optimization [FAL 05] or distributed mechanism design [PAR 04, PET 06]. In this chapter we further analyze a framework for distributed negotiation over indivisible resources which was recently investigated by a number of authors [SAN 98, END 06, DUN 05b].

We assume a set of negotiating agents populating the system, and we model their preferences (over different bundles of resources) by means of utility functions. In order to pursue their own interests, agents agree on deals benefitting themselves but without planning ahead (i.e. they are both rational and myopic [SAN 98]), thereby modifying the allocation of resources. Negotiation is thus a *local* process; groups of individual agents come together to exchange resources in view of their individual interests, without regard for the system as a whole. From a global point of view, on the other hand, a system designer may seek to ensure that negotiation converges towards an allocation that is "optimal" according to a suitable metric. This would typically be some form of aggregation of the individual agent preferences, i.e. we can employ well-known formal tools from welfare economics and social choice theory to characterize optimal allocations. Standard examples are the notions of Pareto optimality and social welfare [ARR 02, MOU 88]. In this paper, we are going to be interested in negotiating allocations of resources that maximize *utilitarian social welfare*, i.e. the sum of individual agent utilities. The work described here is complementary to the large body of literature on mechanism design and game-theoretical models of negotiation in multi-agent systems (see e.g. [KRA 01, ROS 94, PAR 04]). Rather than analyzing the incentives of individual agents in detail, here we simply assume that agents are rational in the sense of never accepting a disadvantageous deal, but we do not assume that they will necessarily attempt to negotiate the very best outcome for themselves in every single step of a negotiation process. Our interests concentrate on the interplay of the local and the global level: assuming that agents only negotiate mutually beneficial deals, how does this affect social welfare for the system as a whole?

Section 7.2 introduces the negotiation framework used in this paper. We are going to distinguish two variants of the framework. In the first one, agents may enhance deals with monetary side payments (*framework with money*) to make them more attractive to agents who would otherwise have no incentive to agree to a particular deal. In the second variant of the negotiation framework, agents are assumed not to have the option of using side payments (*framework without money*). As we shall recall in section 7.3, it is known that very complex multi-lateral deals are potentially required to reach an optimal allocation [SAN 98, END 06]. When deals are restricted (e.g. to a limited number of resources), it is only possible to guarantee an optimal outcome by also restricting the problem space to agents whose preferences have certain properties. These properties are modeled as restrictions to the class of utility functions that agents may use to model their preferences over alternative bundles of resources. Section 7.4 defines two such restrictions. The first is the well-known class of *modular*

utility functions. The second is a further restriction on modular functions, where the utility an agent may assign to any one resource is limited to a choice between three given values (representing positive preference, negative preference and indifference). Each choice of values for positive and negative preference gives rise to a different class of restricted utility functions (the value for indifference is 0 in all cases).

In this chapter, we study the conditions under which negotiation conducted by means of the simplest deals, involving one item at a time (or *1-deal negotiation* for short) still allows us to reach an optimal allocation. Section 7.5 generalizes a result from other works and shows that modeling preferences with modular utility functions is a *sufficient* condition. This applies to the framework with money, and a similar result is proved for negotiation without money. However, modularity is not a *necessary* condition. This is demonstrated in section 7.6 by means of a counterexample. We also show that there can be *no* condition on utility functions that would be both necessary and sufficient for optimal allocations to be negotiable by means of rational 1-deals. These results pertain to conditions on single utility functions, to be met by the utility functions of all the agents in the system. Section 7.6 concludes by showing that describing such a necessary and sufficient condition over the *profile* of utility functions of an entire agent society would be possible in theory, but that it would be computationally intractable to evaluate any such condition in practice. This result is obtained as a by-product of a **coNP**-completeness proof for the problem of checking whether, for a given profile of utility functions, any sequence of rational 1-deals is bound to result in an allocation with maximal social welfare, for a whole range of representation languages for encoding utility functions.

The main contribution of this chapter,[1] as far as the framework with money is concerned, is to show that the class of modular utility functions is *maximal*, in the sense that no class strictly including the modular utility functions would still be sufficient for 1-deal negotiation. The proof detailed in section 7.7 shows that, given any non-modular utility function, it is always possible to construct a modular utility function for a second agent and select a scenario where the optimal allocation cannot be reached by rational 1-deals. This is followed up by a similar result for the framework without money, where each of the aforementioned restrictions on the class of modular utility functions turns out to be both sufficient and maximal for 1-deal negotiation without side payments. Finally, section 7.8 concludes the chapter.

7.2. Myopic negotiation over indivisible resources

In this section, we introduce the decentralized negotiation framework used throughout this paper and report a number of known technical results [END 06].

1. Part of the work reported here has previously been presented at IJCAI-2005 [CHE 05] and we would like to thank the IJCAI reviewers and several colleagues for their helpful comments.

7.2.1. *Negotiation problems and deals*

In this framework, a finite set of agents negotiate over a finite set of indivisible resources. A resource *allocation* is a partitioning of the resources amongst the agents (i.e., every resource has to be allocated to one and only one agent). As an example, the allocation A defined by $A(i) = \{r_1\}$ and $A(j) = \{r_2, r_3\}$ would allocate resource r_1 to agent i, while resources r_2 and r_3 would be owned by agent j. We are going to model the preferences of agents by means of *utility functions* mapping bundles of resources to real numbers. The parameters of a negotiation problem are summarized in the following definition:

DEFINITION 7.1.– *A negotiation problem is a tuple $\mathcal{P} = \langle \mathcal{A}, \mathcal{R}, \mathcal{U}, A_0 \rangle$, where:*

– $\mathcal{A} = \{1, \ldots, n\}$ is a finite set of agents ($n \geqslant 2$);

– $\mathcal{R}$ is a finite set of indivisible resources;

– $\mathcal{U} = \langle u_1, \ldots, u_n \rangle$ is a profile of utility functions, such that for all $i \in \mathcal{A}$, u_i is a mapping from $2^{\mathcal{R}}$ to $\mathbb{R}$;

– $A_0 : \mathcal{A} \to 2^{\mathcal{R}}$ is an (initial) allocation.

Observe that the value of a utility function only depends on the resources held by the agent in question, i.e. agents are not concerned with resources held by other agents or any other external factors. We are frequently going to use the abbreviation $u_i(A)$ for $u_i(A(i))$, representing the utility value assigned by agent i to the bundle it holds in allocation A.

Agents may agree on a *deal* to exchange some of the resources they possess. Such a deal transforms the current allocation of resources A into a new allocation A'; we can define a deal as a pair $\delta = (A, A')$ of allocations (with $A \neq A'$).

We should stress that this is a *multi-lateral* negotiation framework. A single deal may involve the displacement of any number of resources between any number of agents. An actual implementation of this abstract framework may, however, not allow the same level of generality. Sandholm [SAN 98] has proposed a typology of different types of deals, such as *swap deals* involving an exchange of single resources between two agents or *cluster deals* involving the transfer of a set of items from one agent to another. The simplest type of deals are those involving only a single resource (and thereby only two agents).

DEFINITION 7.2.– *A 1-deal is a deal $\delta = (A, A')$ resulting in the reallocation of exactly one resource.*

The above is a condition on the *structure* of a deal. Other conditions relate to the *acceptability* of a deal to a given agent. We assume that agents are *rational* in the sense of aiming at maximizing their individual welfare (utility). Furthermore, agents are

assumed to be *myopic*. This means that agents will not accept deals that would reduce their level of welfare, not even temporarily, because they are either not sufficiently able to plan ahead or not willing to take the associated risk (see also Sandholm's work [SAN 98] for a justification of such an agent model in the context of multi-agent resource allocation).

7.2.2. *Negotiating with money*

In the first variant of this framework, we will allow agents to enhance deals with *monetary side payments*, in order to compensate other agents for a possible loss in utility. This can be modeled using a *payment function* $p : \mathcal{A} \rightarrow \mathbb{R}$. Such a function has to satisfy the side constraint $\sum_{i \in \mathcal{A}} p(i) = 0$, i.e. the overall amount of money in the system remains constant. If $p(i) > 0$, then agent i *pays* the amount of $p(i)$, while $p(i) < 0$ means that it *receives* the amount of $-p(i)$. The following rationality criterion will define the acceptability of deals:

DEFINITION 7.3.– *A deal $\delta = (A, A')$ is individually rational iff there exists a payment function p such that $u_i(A') - u_i(A) > p(i)$ for all $i \in \mathcal{A}$, except possibly $p(i) = 0$ for agents i with $A(i) = A'(i)$.*

From a system designer's perspective, we are interested in assessing the well-being of the whole society, or *social welfare* [ARR 02, MOU 88], which is often defined as the sum of utilities of all the agents.

DEFINITION 7.4.– *The social welfare $sw(A)$ of an allocation A is defined as follows:*

$$sw(A) \quad = \quad \sum_{i \in \mathcal{A}} u_i(A)$$

This is the *utilitarian* definition of social welfare. While this is the definition usually adopted in the multi-agent systems [SAN 99, WOO 02], we should stress that also several of the other notions of social welfare developed in the social sciences (e.g. egalitarian social welfare [ARR 02, MOU 88]) do have potential applications in the context of multi-agent resource allocation [CHE 06].

We now recall two important known results. The first one makes the connection between the local decisions of agents and the global behavior of the system explicit [END 06].

LEMMA 7.1.– *A deal $\delta = (A, A')$ is individually rational iff $sw(A) < sw(A')$.*

In other words, side payments can be arranged in such a way that a given deal is beneficial for all the agents involved if and only if that deal increases social welfare. The second result is the fundamental convergence theorem for this negotiation framework [SAN 98]:

THEOREM 7.1.– *Any sequence of individually rational deals will eventually result in an allocation of resources with maximal social welfare.*

The main significance of this latter result, beyond the equivalence of rational deals and social welfare-increasing deals stated in Lemma 7.1, is that *any* sequence of deals satisfying the rationality criterion will eventually converge to an allocation that is socially optimal. There is no need for agents to consider anything but their individual interests. Every single deal is bound to increase social welfare and there are no local optima.

7.2.3. *Negotiating without money*

What happens if we do *not* allow agents to enhance deals with monetary side payments (or *explicit utility transfers*), in order to compensate other agents for a possible loss in utility? In this context, we shall assume that agents are *cooperatively rational* in the sense of accepting deals that may not result in a *strict* increase in personal welfare, with the further condition that at least one agent will strictly benefit from the deal.

DEFINITION 7.5.– *A deal $\delta = (A, A')$ is called cooperatively rational iff $u_i(A) \leqslant u_i(A')$ for all $i \in \mathcal{A}$ and there exists an agent $j \in \mathcal{A}$ such that $u_j(A) < u_j(A')$.*

Note that we have $sw(A) < sw(A')$ for any deal $\delta = (A, A')$ that is cooperatively rational, but *not* vice versa. Clearly, in this general setting, it is not possible to guarantee that agents will eventually reach an allocation with maximal social welfare, in the sense previously defined. A result similar to Theorem 7.1 can be exhibited, though, if we consider the Pareto optimality criterion instead (an allocation is Pareto optimal when no other allocation is strictly better for one agent without being worse for any other agent): *any* sequence of deals satisfying the cooperative rationality criterion will eventually converge to a Pareto optimal allocation [END 06].

7.3. Convergence for restricted classes of utility functions

While Theorem 7.1 (like the corresponding result for negotiation without money) shows that, in principle, it is always possible to negotiate an allocation of resources that is optimal from a social point of view, deals involving any number of agents and

resources may be required to do so [SAN 98, END 06]. In particular, the most basic type of deal, which involves moving a single resource from one agent to another and which is the type of deal implemented in most systems realising a kind of *Contract-Net* protocol [SMI 80], is certainly *not* sufficient for negotiation between agents that are not only rational but also myopic.

This has first been shown by Sandholm [SAN 98] and is best explained by means of an example. Let $\mathcal{A} = \{1, 2, 3\}$ and $\mathcal{R} = \{r_1, r_2, r_3\}$. Suppose the utility functions of these agents are defined as follows (over singleton sets):

$$\begin{array}{lll} u_1(\{r_1\}) = 5 & u_1(\{r_2\}) = 4 & u_1(\{r_3\}) = 0 \\ u_2(\{r_1\}) = 0 & u_2(\{r_2\}) = 5 & u_2(\{r_3\}) = 4 \\ u_3(\{r_1\}) = 4 & u_3(\{r_2\}) = 0 & u_3(\{r_3\}) = 5 \end{array}$$

Furthermore, for any bundle R not listed above, suppose $u_i(R) = 0$ for all $i \in \mathcal{A}$. Let A_0 with $A_0(1) = \{r_2\}$, $A_0(2) = \{r_3\}$ and $A_0(3) = \{r_1\}$ be the initial allocation, *i.e.* $sw(A_0) = 12$. The optimal allocation would be A^* with $A^*(1) = \{r_1\}$, $A^*(2) = \{r_2\}$ and $A^*(3) = \{r_3\}$, which yields a social welfare of 15. All other allocations have lower social welfare than both A_0 and A^*. Hence, starting from A_0, the deal $\delta = (A_0, A^*)$ would be the only deal increasing social welfare. According to Lemma 7.1, δ would also be the only deal that is individually rational. This deal, however, involves all three resources and affects all three agents. In particular, δ is not a 1-deal. Hence, if we choose to restrict ourselves to rational deals, then 1-deals are not sufficient to negotiate allocations of resources with maximal social welfare. Of course, for some particular negotiation problems, rational 1-deals *will* be sufficient. The difficulty lies in recognising the problems where this is so.

The structural complexity of deals required to be able to guarantee socially optimal outcomes partly stems from the generality of the framework. In particular, so far we have made no assumptions on the structure of utility functions used by the agents to model their preferences. By introducing restrictions on the class of admissible utility functions, it may be possible to ensure convergence to an allocation with maximal social welfare by means of simpler deals. In this paper, we are interested in characterizing those classes of utility functions that allow successful 1-deal negotiation. To this end, we first define what it means for a class of utility functions to guarantee convergence to an optimum for a restricted class of deals:

DEFINITION 7.6.– *Let Φ be a class of deals and let $\mathcal{F}$ be a class of utility functions. Then $\mathcal{F}$ is said to guarantee Φ-convergence iff any sequence of deals belonging to Φ will eventually result in an allocation of resources with maximal social welfare whenever the utility functions of all the agents belong to $\mathcal{F}$.*

Theorem 7.1, for instance, says that the full class of utility functions guarantees convergence for the class of individually rational deals (without any structural restrictions). We are going to be interested in two concrete classes of deals:

– If Φ is the class of individually rational 1-deals and $\mathcal{F}$ is a class of utility functions that guarantees Φ-convergence, then we also say that *$\mathcal{F}$ allows 1-deal negotiation with money*.

– If Φ is the class of cooperatively rational 1-deals and $\mathcal{F}$ is a class of utility functions that guarantees Φ-convergence, then we also say that *$\mathcal{F}$ allows 1-deal negotiation without money*.

For instance, a recent result shows that the class of *additive* utility functions permits 1-deal negotiation with money [END 06].[2] In the next section, we are going to introduce a number of similar restrictions on utility functions, for which we are later going to investigate under what circumstances they may guarantee convergence.

7.4. Modular utility functions and variants

In this section, we are going to introduce several classes of restricted utility functions. We are first going to define the class of *modular* utility functions. This is an important (see e.g. [ROS 94]), albeit simple, class of functions that can be used in negotiation domains where there are no synergies (either complementaries or substitutables) between different resources.

DEFINITION 7.7.– *A utility function u is modular iff the following holds for all bundles $R_1, R_2 \subseteq \mathcal{R}$:*

$$u(R_1 \cup R_2) \quad = \quad u(R_1) + u(R_2) - u(R_1 \cap R_2) \qquad [7.1]$$

The class of modular functions includes the aforementioned additive functions. This may be seen as follows. Let R be any non-empty bundle of resources and let $r \in R$. Then equation [7.1] implies $u(R) = u(R \setminus \{r\}) + [u(\{r\}) - u(\{\,\})]$. If we apply this step recursively for every resource in R, then we end up with the following equation:

$$u(R) \quad = \quad u(\{\,\}) + \sum_{r \in R} [u(\{r\}) - u(\{\,\})] \qquad [7.2]$$

2. A utility function is additive iff the utility assigned to a set of resources is always the sum of utilities assigned to its members.

In other words, in case $u(\{\}) = 0$, the utility assigned to a set will be the sum of utilities assigned to its members (*i.e.* u will be additive). Clearly, equation [7.2] also implies equation [7.1], *i.e.* the two characterizations of the class of modular utility functions are equivalent.

We now introduce a restriction on the class of modular utility functions, namely the classes of *modular functions with shared α, β-values* (or $\mathcal{M}_{\alpha,\beta}$ for short). Intuitively, $\mathcal{M}_{\alpha,\beta}$ classes are suitable in domains where agents can only like, dislike, or possibly be indifferent towards any given resource in the system. The key point is that agents all agree on the intensities used to indicate positive and negative preferences for each single resource. $\mathcal{M}_{\alpha,\beta}$ functions hence define a set of classes, each class being parameterized by the range of possible values assigned to each resource.

DEFINITION 7.8.– *A profile of utility functions $\langle u_1, \ldots, u_n \rangle$ is called modular with shared α, β-values iff:*

– there exists a list of coefficients $\alpha = (\alpha_{r_1}, \ldots, \alpha_{r_{|\mathcal{R}|}}) \in (\mathbb{R}^+)^{|\mathcal{R}|}$,

– there exists a list of coefficients $\beta = (\beta_{r_1}, \ldots, \beta_{r_{|\mathcal{R}|}}) \in (\mathbb{R}^-)^{|\mathcal{R}|}$, and

– for each utility function u_i, there exist two sets $S_i^+, S_i^- \subseteq \mathcal{R}$ with $S_i^+ \cap S_i^- = \{\}$, such that for all bundles $R \subseteq \mathcal{R}$, $u_i(R)$ can be calculated as follows:

$$u_i(R) \quad = \quad u_i(\{\}) + \sum_{r \in R \cap S_i^+} \alpha_r + \sum_{r \in R \cap S_i^-} \beta_r$$

Given two lists of coefficients $\alpha \in (\mathbb{R}^+)^{|\mathcal{R}|}, \beta \in (\mathbb{R}^-)^{|\mathcal{R}|}$, the class $\mathcal{M}_{\alpha,\beta}$ is defined as the largest set of modular functions with shared α, β-values.

Consider, for example, the class $\mathcal{M}_{(2,1),(-20,-7)}$ over resources $\{r_1, r_2\}$ and let u_1 and u_2 be two utility functions drawn from this class such that $u_1(\{\}) = u_2(\{\}) = 0$, $S_1^+ = \{r_1\}$, $S_1^- = \{r_2\}$, $S_2^+ = \{r_2\}$, and $S_2^- = \{\}$. Then, the values of u_1 and u_2 are as shown in the following table:

R	$u_1(R)$	$u_2(R)$
$\{\}$	0	0
$\{r_1\}$	2	0
$\{r_2\}$	-20	1
$\{r_1, r_2\}$	-18	1

Observe that the union of all classes of modular functions with shared α, β-values is the class of modular functions.

7.5. Sufficient classes of utility functions

If a class of utility functions allows 1-deal negotiation (with or without money), then all utility functions belonging to that class constitute a *sufficient* condition for convergence by means of rational 1-deals. In this section, we are going to establish two such results for modular utility functions.

7.5.1. *Framework with money*

It turns out that in domains where all utility functions are modular, it is always possible to reach a socially optimal allocation by means of a sequence of rational deals involving only a single resource each. This is a slight generalization of a result originally proved for the case of additive utilities [END 06] and our proof closely follows that original proof.

THEOREM 7.2.– *The class $\mathcal{M}$ of modular utility functions allows 1-deal negotiation with money.*

Proof. According to Lemma 7.1, any individually rational deal results in a strict increase in social welfare. Together with the fact that the number of distinct allocations is finite, this ensures that there can be no infinite sequence of rational deals (termination). It therefore suffices to show that for any allocation that does not have maximal social welfare there still exists a rational 1-deal that would be applicable.

We are going to use the alternative characterization of modular utility functions given by equation [7.2]. For any allocation A, let f_A be the function mapping each resource r to the agent i that holds r in situation A. Then, for modular domains, the formula for social welfare (see Definition 7.4) can be rewritten as follows:

$$sw(A) \quad = \quad \sum_{i \in \mathcal{A}} u_i(\{\,\}) + \sum_{r \in \mathcal{R}} u'_{f_A(r)}(\{r\})$$

with $u'_i(R) = u_i(R) - u_i(\{\,\})$. Now assume that we have reached an allocation of resources A that does not have maximal social welfare, *i.e.* there exists another allocation A' with $sw(A) < sw(A')$. Considering the above definition of social welfare and observing that $\sum_{i \in \mathcal{A}} u_i(\{\,\})$ is a constant that is independent of the current allocation, this implies that at least one resource r must satisfy the inequality $u'_{f_A(r)}(\{r\}) < u'_{f_{A'}(r)}(\{r\})$, *i.e.* the agent owning r in allocation A values that resource less than the agent owning it in allocation A'. But then the 1-deal consisting of passing r from agent $f_A(r)$ to agent $f_{A'}(r)$ would already increase social welfare and thereby be rational. ∎

Like Theorem 7.1, the above establishes an important convergence result towards a global optimum by means of decentralized negotiation between self-interested agents. In addition, provided all utility functions are modular, convergence can be guaranteed by means of a much simpler negotiation protocol, which only needs to cater for agreements on 1-deals (rather than multilateral deals over sets of resources).

7.5.2. *Framework without money*

We are now going to prove a similar result for the variant of our framework that does not allow monetary side payments.

THEOREM 7.3.– *Given two vectors $\alpha \in (\mathbb{R}^+)^{|\mathcal{R}|}$ and $\beta \in (\mathbb{R}^-)^{|\mathcal{R}|}$, the class $\mathcal{M}_{\alpha,\beta}$ of modular functions with shared α, β-values allows 1-deal negotiation without money.*

Proof. According to Definition 7.5, any cooperatively rational deal results in a strict increase in social welfare. Together with the fact that the number of distinct allocations is finite, this ensures that there can be no infinite sequence of rational deals (termination). It therefore suffices to show that for any allocation that does not have maximal social welfare there still exists a cooperatively rational 1-deal that would be applicable. If an allocation A does not have maximal social welfare then it must be the case that some agent i holds a resource r and that there is another agent j in the system such that $u_i(\{r\}) < u_j(\{r\})$. In other words, either (i) some agent i holds a resource r with $u_i(\{r\}) - u_i(\{\,\}) = 0$, and there is another agent j in the system with $u_j(\{r\}) - u_i(\{\,\}) = \alpha_r$, or (ii) some agent i holds a resource r with $u_i(\{r\}) - u_i(\{\,\}) = \beta_r$, and there is another agent j in the system such that either (ii_a) $u_j(\{r\}) - u_j(\{\,\}) = 0$, or (ii_b) $u_j(\{r\}) - u_j(\{\,\}) = \alpha_r > u_i(\{r\})$. In every case, passing r from i to j would be a cooperatively rational deal, so either negotiation has not yet terminated or we are in a situation with maximal social welfare. ∎

Theorem 7.3 generalizes a previous result stating that any sequence of cooperatively rational 1-deal will eventually result in an allocation with maximal social welfare in case all agents are using so-called 0-1 functions to model their preferences [END 06]. A utility function u_i is a 0-1 function if and only if it is additive and $u_i(\{r\}) = 0$ or $u_i(\{r\}) = 1$ for all $r \in \mathcal{R}$. In other words, the class of 0-1 functions is like the class $\mathcal{M}_{(1,1,\ldots),(0,0,\ldots)}$, except that the former does not allow non-zero utility values to be assigned to the empty set.

7.6. Necessity issues

As explained earlier, the convergence results of the previous section may be considered results on *sufficient* conditions on utility functions for convergence by means

of rational 1-deals. This naturally raises the question of whether there may be a corresponding set of *necessary* conditions on utility functions. In this section, we are going to give a negative answer to this question. We are first going to show that all utility functions belonging to the class of modular functions (which we have shown to be a sufficient condition for convergence by means of 1-deals) is not a necessary condition. Then we are going to show that, in fact, there can be *no* condition on individual utility functions that would be both necessary and sufficient in this sense. Of course, it *is* possible to formulate a necessary and sufficient condition for convergence by means of 1-deals for *profiles* of utility functions (i.e. if we are allowed to give conditions on how the utilities of different agents should relate to each other). However, as we are going to show, evaluating such a condition would be computationally intractable[3].

7.6.1. *Modularity is not necessary*

Modularity is not a necessary condition for convergence by means of individually rational 1-deals. We demonstrate this by means of the following example. Suppose that $\mathcal{R} = \{r_1, r_2\}$ and there are two agents with utility functions u_1 and u_2 defined as follows:

$$\begin{array}{llllll} u_1(\{\}) & = & 0 & u_2(\{\}) & = & 0 \\ u_1(\{r_1\}) & = & 10 & u_2(\{r_1\}) & = & 10 \\ u_1(\{r_2\}) & = & 10 & u_2(\{r_2\}) & = & 10 \\ u_1(\{r_1, r_2\}) & = & 0 & u_2(\{r_1, r_2\}) & = & 0 \end{array}$$

These functions are not modular. The situation is as follows: each agent is willing to hold a single resource and has no preference as to which resource it actually holds. The optimal allocations are those where each agent holds exactly one resource. Furthermore, as may easily be checked, individually rational 1-deals are sufficient to move to the optimal allocation for this scenario, despite the functions not being modular. Hence, modularity cannot be a necessary condition for convergence.

7.6.2. *There is no sufficient and necessary class*

It is possible to show that there can be no class of utility functions that would be both sufficient and necessary. It suffices to produce two concrete utility functions u_1 and u_2 such that: (i) both of them would guarantee convergence if all agents were to use them and (ii) there is a scenario where some agents are using u_1 and others

3. While the analysis in this section applies to the framework of negotiation with money, a similar case may be made for negotiation without money.

u_2 and convergence is not guaranteed. This is so because assuming that a necessary and sufficient class exists, (i) would imply that both u_1 and u_2 belong to that class, while (ii) would entail the contrary. We give two such functions for the case of two agents and two resources (the argument is easily augmented to the general case; simply assume $u_i(R) = 0$ for $i > 2$):

$$\begin{array}{lcllcl} u_1(\{\,\}) & = & 0 & u_2(\{\,\}) & = & 0 \\ u_1(\{r_1\}) & = & 1 & u_2(\{r_1\}) & = & 5 \\ u_1(\{r_2\}) & = & 2 & u_2(\{r_2\}) & = & 5 \\ u_1(\{r_1, r_2\}) & = & 3 & u_2(\{r_1, r_2\}) & = & 5 \end{array}$$

The function u_1 is modular, i.e. all agents using that function is a sufficient condition for guaranteed convergence to an optimal allocation by means of individually rational 1-deals (Theorem 7.2). Clearly, convergence is also guaranteed if all agents are using u_2. However, if the first agent uses u_1 and the second u_2, then the allocation A with $A(1) = \{r_1\}$ and $A(2) = \{r_2\}$ is not socially optimal and the only deal increasing social welfare (and thereby, the only rational deal) would be to swap the two resources simultaneously. Hence, no condition on all agents' utility functions can be both sufficient and necessary to guarantee convergence to an optimal allocation by means of individually rational 1-deals alone.

7.6.3. *Evaluating conditions on profiles of utility functions is intractable*

Our argument for the inexistence of any necessary and sufficient condition for guaranteed convergence has directly exploited the fact that we were looking for a *single* condition to be met by the utility functions of *all* agents. The problem could be circumvented by looking for suitable conditions on negotiation problems as a whole, where different utility functions may meet different such conditions. In other words, we could try to formulate a condition for *profiles* $\langle u_1, \ldots, u_n \rangle$ of utility functions. Clearly, such a condition *does* exist: trivially, $\langle u_1, \ldots, u_n \rangle$ meets that condition iff convergence to an optimal allocation is guaranteed for any initial allocation when agents are using the utility functions $\langle u_1, \ldots, u_n \rangle$. Of course, phrased like this, this condition is of no practical use. What we would require is a "simple" condition on utility profiles. The condition of modularity (for single utilities), for instance, clearly is such a simple condition.

To make this notion of simplicity precise, we first need to fix a *representation language* for utility functions. Then we can check whether or not it is the case that checking a particular condition for utilities represented in this language is a tractable decision problem. As we are going to argue in this section, for a wide range of representation languages, there can be no condition on utility profiles that would be simple

in this sense. We first note that checking whether a given utility profile satisfies the necessary and sufficient condition for 1-deal negotiation is equivalent to a decision problem that has been referred to as 1-convergence (1-CONV) in [CHE 06, DUN 05a]:

DEFINITION 7.9.– *Given $\langle \mathcal{A}, \mathcal{R}, \mathcal{U} \rangle$, the 1-convergence problem (1-CONV) is the problem of checking whether any sequence of individually rational 1-deals (originating from any initial allocation) will result in an allocation with maximal social welfare.*

1-CONV has been shown to be **coNP**-complete in the case where utility functions are represented using the so-called SLP form, which encodes utilities as *straight-line programs* [DUN 05a]. This means that checking whether a profile of utility functions given in SLP form satisfies the necessary and sufficient condition for 1-deal negotiation is also **coNP**-complete, hence computationally intractable. In what follows, we are going to extend this complexity result to a whole range of representation languages, which only have to meet very weak conditions for the hardness result to apply.

We are going to establish **coNP**-hardness via a reduction from the so-called welfare improvement (WI) problem, which is closely related to the problem of finding the maximum of a utility function and which is known to be **NP**-hard for a wide range of representation languages for utility functions.

DEFINITION 7.10.– *Given a utility function u in a particular representation language and a bundle $Q \subseteq \mathcal{R}$, the welfare improvement (WI) problem is the problem of checking whether there exists a bundle $Q' \in \mathcal{R}$ such that $u(Q) < u(Q')$.*

Our proof applies to utility representation languages that are r-composable:

DEFINITION 7.11.– *Let $\mathcal{REP}$ and $\mathcal{REP}'$ be two representation languages for utility functions. Let $r \in \mathcal{R}$ be a resource. Then, $\mathcal{REP}$ is said to be an r-composition over $\mathcal{REP}'$ iff $\forall u \in \mathcal{REP}'$, $\forall x \in \{u(R) \mid R \subseteq \mathcal{R}\}$, the utilities v and $\bar{v}$ defined below belong to $\mathcal{REP}$ and the number of bits needed to represent both utilities is polynomial in $|u|$ (the number of bits needed to encode u) and in the number of resources*[4].

$$v(R) = \begin{cases} u(R) - x & \text{if } r \in R \\ 0 & \text{otherwise} \end{cases} \qquad \bar{v}(R) = \begin{cases} u(R) - x & \text{if } r \notin R \\ 0 & \text{otherwise} \end{cases}$$

4. Note that encoding the utility function v (resp. $\bar{v}$) as a sequence of bits requires encoding u, r, and x. Unfortunately, x may be a huge fractional number. Thus, if we choose to encode x straightforwardly as a sequence of bits, the total number of bits may not be polynomial in $|u|$ and $|\mathcal{R}|$. To overcome this, instead of encoding x, we may encode the bundle R whose value under v is x. This way, the size of $|v|$ remains polynomial in $|u|$ and $|\mathcal{R}|$. This will be of some importance in the main proof of this section, as we will use a reduction to a decision problem in which the size of instances will also be polynomial in $|u|$ and $|\mathcal{R}|$.

For the sake of simplicity, we will say that a representation $\mathcal{REP}$ is r-*composed* iff there exists a representation $\mathcal{REP}'$ over which $\mathcal{REP}$ is an r-composition.

As many interesting representation languages are r-composed, the results presented here are quite general. Let us briefly mention some of the representation languages commonly used in multiagent resource allocation [CHE 06]. In what follows, r is any resource chosen among $\mathcal{R}$.

– The k-*additive representation* [GRA 97] for $k \geqslant 3$ is an r-composition over the $(k-1)$-additive representation. Intuitively, k-additive utilities can be represented by a multi-nomial of degree k [BOR 02]. Note that for $k \geqslant 2$, the WI problem with k-additive utilities is **NP**-hard [CHE 04].

– *Posiforms of degree* k [BOR 02] for $k \geqslant 3$ are r-compositions over posiforms of degree $k-1$. This is a generalization of the k-additive forms and thus, for $k \geqslant 2$, the WI problem here is also **NP**-hard [CHE 04].

– The *bundle form* representation [CHE 04] is also an r-composition over itself. It consists of simply enumerating all bundles of resources for which the utility is not zero. Again, the associated WI problem is known to be **NP**-hard [CHE 04] .

– The SLP (straight-line programs) representation [DUN 05b] is an r-composition over itself. Intuitively, SLPs are computer programs without loops, taking a bundle of resources as input and returning the associated utility value as output. Here, WI is also **NP**-hard [DUN 05b].

– The XOR bidding language [SAN 02, NIS 06], developed in the combinatorial auction community, is also an r-composition over itself and has an **NP**-hard WI problem.

We can now present the main theorem of this section.

THEOREM 7.4.– *Let $\mathcal{REP}$ be a representation language. If the following conditions hold, then the 1-convergence problem over $\mathcal{REP}$ is* ***coNP****-complete under Turing reduction, even with only two agents:*

– there exists a representation $\mathcal{REP}'$ over which $\mathcal{REP}$ is an r-composition;

– the WI problem with this representation $\mathcal{REP}'$ is ***NP****-hard; and*

– there exists a zero-valued utility function v_0 in $\mathcal{REP}$.

Proof. Membership of 1-CONV to **coNP** may be shown as in [DUN 05a]. In what follows, we will show **coNP**-hardness using a Turing reduction from the WI problem. To achieve this, we will show that, given an oracle solving the 1-CONV problem in a single time unit, we can build a deterministic algorithm solving the WI problem in polynomial time. The proof is constructive, i.e. the algorithm will be given below.

First, let us recall some preliminary definitions. Let $\langle u, Q \rangle$ be an instance of the WI problem over the $\mathcal{REP}'$ representation. Note that the number of bits required to

encode such an instance is bounded by $|u| + |\mathcal{R}|$. Let v_0 be the zero-valued utility function, and let the set of agents be $\mathcal{A} = \{1, 2\}$. We now introduce two utility functions:

$$v(R) = \begin{cases} u(R) - u(Q) & \text{if } r \in R \\ 0 & \text{otherwise} \end{cases} \quad \bar{v}(R) = \begin{cases} u(R) - u(Q) & \text{if } r \notin R \\ 0 & \text{otherwise} \end{cases}$$

Clearly, these two functions belong to $\mathcal{REP}$ because of the r-composition property. Based on these functions, we can now define two instances of 1-CONV as follows: let $\mathcal{I}$ (resp. $\bar{\mathcal{I}}$) be the 1-CONV instances for two agents with utilities v and v_0 (resp. $\bar{v}$ and v_0). We will show next that deciding whether these instances are 1-convergent or not will eventually give us the answer to the WI problem $\langle u, Q\rangle$.

First of all, note that if the answer to the WI problem over $\langle u, Q\rangle$ is NO (i.e. if Q is an optimal bundle), then both $\mathcal{I}$ and $\bar{\mathcal{I}}$ are 1-convergent. In fact, 0 will be in this case the highest value which the functions v and $\bar{v}$ can take and moving resource r from an agent to another in $\mathcal{I}$ or $\bar{\mathcal{I}}$ will lead to an optimal allocation if the current allocation is not already optimal. Due to the converse, we can deduce the following:

– if $\mathcal{I}$ or $\bar{\mathcal{I}}$ is not 1-convergent, then the answer to the WI problem is YES.

Consider now the case where both problems are 1-convergent. Clearly, the answer to the WI problem is YES *if and only if* there exists an allocation A' such that $v(A') > 0$ *or* there exists an allocation $\bar{A}'$ such that $\bar{v}(\bar{A}') > 0$. Because both instances are known to be 1-convergent, checking if there are allocations valued more than 0 can be done in the following way: given an allocation A (resp. $\bar{A}$) valued 0 in $\mathcal{I}$ (resp. $\bar{\mathcal{I}}$), check whether or not there exists an individually rational 1-deal. If so, the WI problem will be answered YES. For our 0-valued allocation, it suffices to choose $A = \langle\{\,\}, \mathcal{R}\rangle$ (resp. $\bar{A} = \langle\mathcal{R}, \{\,\}\rangle$). These allocations have a social welfare in $\mathcal{I}$ (resp. $\bar{\mathcal{I}}$) whose values are $v(A) + v_0(A) = \bar{v}(\bar{A}) + v_0(\bar{A}) = 0$. To summarize:

– If both $\mathcal{I}$ and $\bar{\mathcal{I}}$ are 1-convergent, then consider allocation $A = \langle\{\,\}, \mathcal{R}\rangle$ (resp. $\bar{A} = \langle\mathcal{R}, \{\,\}\rangle$). If there exists any individually rational 1-deal $\delta = (A, A')$ over $\mathcal{I}$ or $\bar{\delta} = (\bar{A}, \bar{A}')$ over $\bar{\mathcal{I}}$, then the answer to the WI problem is YES, otherwise it is NO.

This covers all possible cases. Hence, given our oracle for 1-CONV, we can always derive an answer to WI in polynomial time. Our reduction is summarized by the algorithm shown in Figure 7.1. ∎

To conclude this section, we recall that the **coNP**-completeness result for 1-CONV entails that, for many common representation languages, checking whether profiles of utilities enable 1-deal negotiation is intractable. Therefore, searching for a condition over profiles of utility functions is unrealistic. As we have shown that no condition on *single* utility functions is both sufficient and necessary, and as, on top of that, we have just argued that checking conditions on *profiles* of utility functions is intractable in

Algorithm: deciding WI with the help of an oracle for 1-CONV
Input: utility function u, bundle $Q \subseteq \mathcal{R}$
Output: YES or NO

1) if $\{v, v_0\}$ is not 1-CONV or $\{\bar{v}, v_0\}$ is not 1-CONV then return YES;

2) if there exists an individually rational 1-deal improving allocation $\langle\{\}, \mathcal{R}\rangle$ within $\langle v, v_0\rangle$ then return YES;

3) if there exists an individually rational 1-deal improving allocation $\langle\mathcal{R}, \{\}\rangle$ within $\langle\bar{v}, v_0\rangle$ then return YES;

4) return NO

Figure 7.1. *Algorithm summarizing the reduction from* 1-CONV *to WI*

most cases, the best thing we can do is to investigate whether some restricted classes of utility can be identified as being *maximal*. This is what we shall do in the next section.

7.7. Maximal classes of utility functions

We are now going to prove the main results of this chapter, namely the surprising fact that the classes of utility functions shown to allow 1-deal negotiation (with money and without money) are not only sufficient, but also *maximal* in the sense that no class of utility functions strictly including this class of functions would still be sufficient for 1-deal negotiation. The significance of this result can only be fully appreciated when considered together with the "negative" result on necessary and sufficient conditions discussed in the previous section.

Before stating the main result, we prove the following auxiliary lemma:

LEMMA 7.2.– *A utility function u is modular iff the following holds for all $R \subseteq \mathcal{R}$ and all $r_1, r_2 \in \mathcal{R}$ with $r_1, r_2 \notin R$ and $r_1 \neq r_2$:*

$$u(R \cup \{r_1, r_2\}) \quad = \quad u(R \cup \{r_1\}) + u(R \cup \{r_2\}) - u(R) \qquad [7.3]$$

Proof. To show this, let us recall elementary facts about submodular functions. A function $v : \mathcal{R} \rightarrow \mathbb{R}$ is submodular iff $\forall R_1, R_2 \subseteq \mathcal{R},\ v(R_1) + v(R_2) \geqslant v(R_1 \cup R_2) + v(R_1 \cap R_2)$. It is also known that v is submodular iff $v(R \cup \{r_1\}) + v(R \cup \{r_2\}) \geqslant v(R \cup \{r_1, r_2\}) - v(R)$ for any $R \subseteq \mathcal{R},\ r_1, r_2 \in \mathcal{R} \backslash R$, with $r_1 \neq r_2$ [NEM 88, p.662]. Because a function u is modular iff both u and $-u$ are submodular, the lemma holds. ∎

7.7.1. *Framework with money*

We are now in a position to prove our theorem on the maximality of the class of modular utility functions with respect to rational 1-deal negotiation with money:

THEOREM 7.5.– *Let $\mathcal{M}$ be the class of modular utility functions. Then for any class of utility functions $\mathcal{F}$ such that $\mathcal{M} \subset \mathcal{F}$, $\mathcal{F}$ does not allow 1-deal negotiation with money.*

Proof. First observe that for $|\mathcal{R}| \leqslant 1$, *any* utility function is modular, *i.e.* the theorem holds vacuously in these cases. Therefore, without loss of generality, from now on we assume that there are at least two distinct resources in the system.

The proof is constructive. We will show that for any non-modular utility function u_1 on m resources, it is possible to construct a modular utility function u_2 (with $u_i \equiv 0$ for all other agents i) and an initial allocation such that no optimal allocation can be reached by means of 1-deals. This implies that $\mathcal{M} \cup \{u_1\}$ does not allow 1-deals.

Because u_1 is non-modular, Lemma 7.2 can be applied in the following way: there is a bundle X and distinct resources $r_1, r_2 \notin X$ such that ϵ, defined as follows, is not equal to 0:

$$\epsilon = u_1(X \cup \{r_1\}) + u_1(X \cup \{r_2\}) - u_1(X) - u_1(X \cup \{r_1, r_2\}) \qquad [7.4]$$

From now on, $A_{12|}$, $A_{|12}$, $A_{1|2}$ and $A_{2|1}$ will refer to allocations in which r_1 and r_2 belong to one of the first two agents, resources in X are owned by 1 and resources in $Y = \mathcal{R} \backslash (X \cup \{r_1, r_2\})$ by 2, as shown in the following table.

	Agent 1	Agent 2
$A_{12\|}$	$\{r_1, r_2\} \cup X$	Y
$A_{\|12}$	X	$\{r_1, r_2\} \cup Y$
$A_{1\|2}$	$\{r_1\} \cup X$	$\{r_2\} \cup Y$
$A_{2\|1}$	$\{r_2\} \cup X$	$\{r_1\} \cup Y$

Let us build a modular utility function u_2 defined as follows: $\forall R \in \mathcal{R}$:

$$u_2(R) = \sum_{r \in \{r_1, r_2\} \cap R} \alpha_r + \sum_{r \in R \cap Y} \omega - \sum_{r \in R \cap X} \omega \qquad [7.5]$$

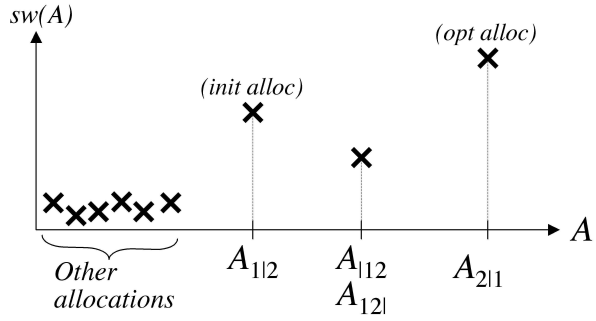


Figure 7.2. *Values of sw for the four allocations (case $\epsilon > 0$)*

with $\omega = 14 \times \max|u_1| + 1$. Let $\Omega = u_2(Y) = |Y| \times \omega$. As the rest of the proof shall reveal, the value of ω has been chosen such that the social welfare of each of these four allocations is greater than that of any other allocation. Of course, this will imply that the optimal allocation has to be among these four. The values of α_{r_1} and α_{r_2} will be chosen later. The social welfare of each of these four allocations can then be written as follows:

$$\begin{aligned} sw(A_{|12}) &= \Omega + \alpha_{r_1} + \alpha_{r_2} + u_1(X) \\ sw(A_{12|}) &= \Omega + u_1(X \cup \{r_1, r_2\}) \\ sw(A_{1|2}) &= \Omega + \alpha_{r_2} + u_1(X \cup \{r_1\}) \\ sw(A_{2|1}) &= \Omega + \alpha_{r_1} + u_1(X \cup \{r_2\}) \end{aligned}$$

It remains to be shown that depending on the value of ϵ, we can always choose an initial allocation among these four and values of α_{r_1} and α_{r_2} such that: (1) this initial allocation does not have optimal social welfare, (2) there is only one rational deal from this allocation, (3) this deal leads to the optimal allocation, but (4) this rational deal would involve more than one resource. We will have to consider two cases for equation [7.4]: the case of $\epsilon > 0$ and the case of $\epsilon < 0$.

(1st case) Suppose that $\epsilon > 0$. Let us choose $\alpha_{r_1} = u_1(X \cup \{r_1\}) - u_1(X) - \frac{\epsilon}{4}$ and $\alpha_{r_2} = u_1(X \cup \{r_1, r_2\}) - u_1(X \cup \{r_1\}) + \frac{\epsilon}{4}$.

Let us first show that the four allocations have a greater social welfare than any other. With the help of equation [7.4], observe that both $|\alpha_{r_1}|$ and $|\alpha_{r_2}|$ are less than $3 \times \max|u_1|$. Thus, all four allocations have a social welfare of at least $\Omega - |\alpha_{r_1}| -$

$|\alpha_{r_2}| - \max|u_1| \geqslant \Omega - 7 \times \max|u_1| > \Omega - \frac{\omega}{2}$. All other allocations have a social welfare lower than $\Omega - \omega + |\alpha_{r_1}| + |\alpha_{r_2}| + \max|u_1| \leqslant \Omega - \omega + 7 \times \max|u_1| < \Omega - \frac{\omega}{2}$. Thus, the social welfare of each of the four allocations is greater than that of any other allocation.

Now let us show that $A_{2|1}$ is the optimal allocation, as illustrated in Figure 7.2. More precisely, let us show that $sw(A_{|12}) < sw(A_{1|2})$, that $sw(A_{12|}) < sw(A_{1|2})$ and that $sw(A_{1|2}) < sw(A_{2|1})$. By substituting the values of α_{r_1} and α_{r_2} and using equation [7.4], the social welfare of each allocation can be written as follows:

$$\begin{aligned}
sw(A_{|12}) &= \Omega + u_1(X \cup \{r_1, r_2\}) \\
sw(A_{12|}) &= \Omega + u_1(X \cup \{r_1, r_2\}) \\
sw(A_{1|2}) &= \Omega + u_1(X \cup \{r_1, r_2\}) + \frac{\epsilon}{4} \\
sw(A_{2|1}) &= \Omega + u_1(X \cup \{r_1\}) + u_1(X \cup \{r_2\}) \\
&\quad -u_1(X) - \frac{\epsilon}{4} \\
&= \Omega + u_1(X \cup \{r_1, r_2\}) + \frac{3}{4}\epsilon
\end{aligned}$$

Here, $A_{2|1}$ is clearly the optimal allocation. If we choose $A_{1|2}$ as the initial allocation, then the only 1-deals involving resources r_1 or r_2 are $\delta(A_{1|2}, A_{12|})$ and $\delta(A_{1|2}, A_{|12})$. These deals decrease social welfare, and thus are not individually rational according to Lemma 7.1. Thus, it is not possible to reach the optimal allocation $A_{2|1}$ starting from $A_{1|2}$ using only 1-deals.

(2nd case) Suppose that $\epsilon < 0$. Let us choose $\alpha_1 = u_1(X \cup \{r_1\}) - u_1(X) - \frac{\epsilon}{4}$ and $\alpha_2 = u_1(X \cup \{r_2\}) - u_1(X) - \frac{\epsilon}{4}$.

Note that again, both $|\alpha_{r_1}|$ and $|\alpha_{r_2}|$ are less than $3 \times \max|u_1|$. Thus, by the same argument as in the first case, the four allocations all have greater social welfare than any other allocation.

The optimal allocation is now $A_{12|}$. To see this, let us show that $sw(A_{1|2}) < sw(A_{|12})$, that $sw(A_{2|1}) < sw(A_{|12})$ and that $sw(A_{|12}) < sw(A_{12|})$ as illustrated in Figure 7.3:

$$\begin{aligned}
sw(A_{|12}) &= \Omega + u_1(X \cup \{r_1\}) + u_1(X \cup \{r_2\}) \\
&\quad -u_1(X) - \frac{\epsilon}{2}
\end{aligned}$$

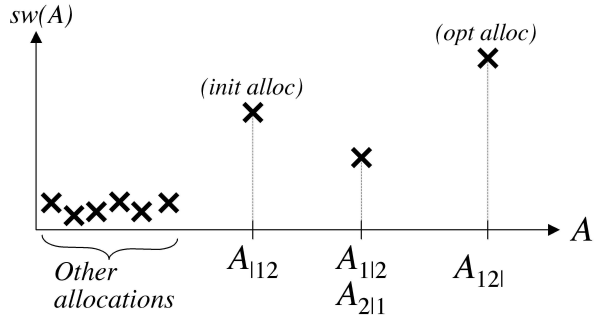


Figure 7.3. *Values of sw for the four allocations (case $\epsilon < 0$)*

$$\begin{aligned}
sw(A_{12|}) &= \Omega + u_1(X \cup \{r_1, r_2\}) \\
&= \Omega + u_1(X \cup \{r_1\}) + u_1(X \cup \{r_2\}) \\
&\quad -u_1(X) - \epsilon \\
sw(A_{1|2}) &= \Omega + u_1(X \cup \{r_1\}) + u_1(X \cup \{r_2\}) \\
&\quad -u_1(X) - \frac{\epsilon}{4} \\
sw(A_{2|1}) &= \Omega + u_1(X \cup \{r_1\}) + u_1(X \cup \{r_2\}) \\
&\quad -u_1(X) - \frac{\epsilon}{4}
\end{aligned}$$

Here, $A_{12|}$ is clearly the optimal allocation. If we choose $A_{|12}$ as the initial allocation, then the only 1-deals involving r_1 or r_2 are $\delta(A_{|12}, A_{1|2})$ and $\delta(A_{|12}, A_{2|1})$. These deals decrease social welfare, and thus are not individually rational according to Lemma 7.1. Thus, it is not possible to reach the optimal allocation $A_{12|}$ starting from $A_{|12}$ using only 1-deals. ∎

Why is this result significant? As argued earlier, while the full abstract negotiation framework introduced at the beginning of this paper would be difficult to implement, designing a system that only allows pairs of agents to agree on deals over one resource at a time is entirely feasible. As we would like to be able to guarantee socially optimal outcomes in as many cases as possible, also for such a restricted negotiation system, we would like to be able to identify the largest possible class of utility functions for which such a guarantee can be given. However, our discussion in section 7.6 has shown that there can be no class of utility functions that *exactly* characterizes the class of negotiation problems for which negotiating socially optimal allocations by means of

rational 1-deals is always possible. Still, there *are* classes of utility functions that allow 1-deal negotiation. As shown by Theorem 7.2, the class of modular functions is such a class and it is certainly a very natural class to consider. An obvious question to ask is therefore whether this class can be enlarged in any way without losing the desired convergence property.

Our proof of Theorem 7.5 settles this question by giving a negative answer: for any agent with a non-modular utility function there are modular utility functions (for the other agents) and an initial allocation such that rational 1-deals alone do not suffice to negotiate an allocation of resources with maximal social welfare. Of course, there may well be further such classes (that are both sufficient and maximal), but we have not been able to identify any such class that would also be *interesting*. An example for a class of utility functions that is sufficient but *not* interesting is the class of *pseudo-constant* functions. A utility function u_i is pseudo-constant iff there exists a $c \geqslant 0$ such that $u_i(R) = c$ for all $R \neq \{\}$ and $u_i(\{\}) = 0$. Clearly, if all agents use such a function, then individually rational 1-deals will be possible until an allocation with maximal social welfare has been reached; but this is of course a very limited model of agent preferences.

7.7.2. *Framework without money*

Can the result of the framework with money be translated into our framework without money? We are now going to show that this is the case: any class of modular utility functions with α, β-shared values is not only sufficient for 1-deal negotiation, but also *maximal* in the sense that no class of utility functions strictly including that class would still be sufficient.

THEOREM 7.6.– *Let $\mathcal{M}_{\alpha,\beta}$ be a class of modular functions with shared α, β-values. Then for any class of utility functions $\mathcal{F}$ such that $\mathcal{M}_{\alpha,\beta} \subset \mathcal{F}$, $\mathcal{F}$ does not allow 1-deal negotiation without money.*

Proof. The proof is constructive and involves two agents. We will show that for *any* function u_1 not belonging to the class of modular functions with shared α, β-values, it is possible to construct a modular utility function u_2 and to choose an initial allocation such that the optimal allocation cannot be reached by means of cooperatively rational 1-deals. This will imply that $\mathcal{M}_{\alpha,\beta} \cup \{u_1\}$ does not allow 1-deal negotiation.

First note that for any modular function with shared α, β-values, any $X \subset \mathcal{R}$ and any $r \in \mathcal{R} \backslash X$ we have $u(X \cup \{r\}) = u(X) + v$, with $v \in \{\alpha_r, 0, \beta_r\}$. Let u_1 be an arbitrary function, *not* belonging to the class $\mathcal{M}_{\alpha,\beta}$; in other words, there exist a set $X \subset \mathcal{R}$ and a resource $r_1 \in \mathcal{R} \backslash X$ such that $u(X \cup \{r_1\}) = u(X) + v$, with

$v \notin \{\alpha_{r_1}, 0, \beta_{r_1}\}$. We are going to define a second utility function u_2 in terms of X and r_1. Namely, let $u_2 \in \mathcal{M}_{\alpha,\beta}$ be defined as follows (for all $R \in \mathcal{R}$) :

$$u_2(R) = \sum_{r \in R \cap X} \beta_r + \sum_{r \in R \setminus (X \cup \{r_1\})} \alpha_r + \sum_{r \in R \cap \{r_1\}} z$$

with z to be chosen in $\{\alpha_{r_1}, \beta_{r_1}\}$.

Observe that we have chosen $u_2(\{\,\}) = 0$.

We now consider the two allocations where agent 1 owns all resources in X and agent 2 owns all resources in $\mathcal{R}\setminus(X \cup \{r_1\})$. Amongst these allocations, let $A_{r_1|}$ (resp. $A_{|r_1}$) be the one where agent 1 (resp. agent 2) owns in addition resource r_1. We are going to show that we can always choose a non-optimal initial allocation amongst these two such that no cooperatively rational 1-deal is possible. First observe that agent 1 cannot pass a single resource $r \in X$ to agent 2 (because u_2 would decrease by $|\beta_r|$) and that agent 2 cannot pass a single resource $r \in \mathcal{R}\setminus(X \cup \{r_1\})$ to agent 1, because in this case u_2 would decrease by the value α_r. Let us now consider the case of 1-deals involving resource r_1. We are going to distinguish four different cases, depending on the value of v. Note that $sw(A_{r_1|}) - sw(A_{|r_1}) = v - z$ and recall that $\alpha_{r_1} > 0$ and $\beta_{r_1} < 0$.

1) Suppose that $v > \alpha_{r_1}$. By choosing $z = \alpha_{r_1}$, we prevent agent 2 from passing r_1 on to agent 1 (because u_2 would decrease by α_{r_1}), but still we have $sw(A_{r_1|}) > sw(A_{|r_1})$ (because $v - z > 0$). Hence, $A_{|r_1}$ is a non-optimal allocation from which there are no cooperatively rational 1-deals.

2) Suppose that $v < \beta_{r_1}$. By choosing $z = \beta_{r_1}$, we prevent agent 1 from passing r_1 on to agent 2 (because u_1 would decrease by $|\beta_{r_1}|$), but still we have $sw(A_{r_1|}) < sw(A_{|r_1})$ (because $v - z < 0$). So in this case $A_{r_1|}$ is a non-optimal allocation from which there are no cooperatively rational 1-deals.

3) Suppose that $0 < v < \alpha_{r_1}$. By choosing $z = \alpha_{r_1}$, we have $sw(A_{r_1|}) < sw(A_{|r_1})$ (because $v - z < 0$), but still agent 1 cannot give r_1 to agent 2 (because u_1 would decrease by v).

4) Suppose that $\beta_{r_1} < v < 0$. By choosing $z = \beta_{r_1}$, we have $sw(A_{r_1|}) > sw(A_{|r_1})$ (because $v - z > 0$), but still agent 2 cannot give r_1 to agent 1 (because u_2 would decrease by $|v|$).

This covers all possible cases. As we have seen, for any choice of $u_1 \notin \mathcal{M}_{\alpha,\beta}$ we can construct a function $u_2 \in \mathcal{M}_{\alpha,\beta}$ and choose an initial allocation of resources such that negotiation conducted solely by means of cooperatively rational 1-deals will get stuck in a final allocation that does not have maximal social welfare. ∎

7.8. Conclusion

This paper makes a contribution to the theoretical analysis of a negotiation framework where rational but myopic agents agree on a sequence of deals regarding the reallocation of a number of indivisible resources. We have shown that the use of *modular* utility functions to model agent preferences is a *sufficient* condition, if side payments are allowed, to guarantee final allocations with maximal social welfare in the case where agents only negotiate *1-deals* (involving one resource each). When no side payments are permitted, any class of $\mathcal{M}_{\alpha,\beta}$-functions would be sufficient. We have then seen that this is, however, not a *necessary* condition for optimal outcomes and indeed, there can be no condition on (individual) utility functions that would be both necessary and sufficient in this sense. Furthermore, we have shown that, while a necessary and sufficient condition on profiles of utility functions obviously does exist, checking such a condition is intractable for most representations commonly used in multiagent resource allocation settings. We have therefore concentrated on showing that the class of modular (resp. the class of modular functions with shared α, β-values) functions is *maximal* in the framework with money (resp. without money), i.e. no strictly larger class of functions would still allow an optimal allocation to be found by means of rational 1-deals in all cases.

We consider these not only surprising results, but also useful characterizations of negotiation domains that can be handled reliably using simple negotiation protocols, catering only for *Contract-Net*-like deals over single items between pairs of agents, rather than the full range of multilateral deals foreseen in the abstract framework. Such theoretical results affect both the design of agents and of negotiation mechanisms. For instance, if a given mechanism can only handle 1-deals, then it may be inappropriate to design myopic agents with very rich preference structures to use such a mechanism.

In the introduction, we have argued for *distributed* approaches to resource allocation. Of course it would be over-simplistic to merely distinguish a centralized approach, *à la* combinatorial auction, versus a purely distributed approach. The truth is that there exists an entire spectrum of approaches between these two extremes. Very often, for instance, even if an auctioneer is available, it is desirable anyway to reduce part of the burden of the calculational task usually assigned to it. One way to do so is to delegate it to the bidders instead. This distribution of the *calculation* of the winner determination problem has been studied by some authors [NAR 05, VID 06]. What remains centralized in these approaches, however, is the fact that agents must be aware of the bids placed by other agents during the process to calculate their new bid. In other words, while the calculation of the solution is indeed distributed, the process itself remains centralized. In our approach, on the other hand, we assume that agents agree *locally* on beneficial deals, without requiring any sort of global perception of the situation. Also, while the actual allocation is modified once and for all in the case of combinatorial auctions, our setting regards negotiation as a continuously evolving

process with a sequence of local steps incrementally modifying the allocation of resources within the system.

This chapter continues recent work on convergence properties for distributed negotiation schemes [SAN 98, END 06]. Related work, notably [DUN 05b, DUN 05a], has concentrated on the computational complexity of decision problems arising in the context of such negotiation frameworks. The complexity of the most fundamental problem in resource allocation, namely social welfare optimization, has been known for some time. The underlying decision problem, i.e. checking whether a given negotiation problem admits a solution where the sum of utilities exceeds a given value, is **NP**-complete [ROT 98].[5] This can be shown, for instance, via a reduction from the well-known SET PACKING problem [GAR 79]. Dunne *et al.* [DUN 05b] have introduced the 1-PATH problem and analysed its complexity: given two allocations A and A' with $sw(A) < sw(A')$, check whether it is possible to reach A' from A by means of a sequence of rational 1-deals. Dunne *et al.* [DUN 05b] have shown that 1-PATH is **NP**-hard. Recent results have further strengthened this complexity bound and established that the problem is in fact **PSPACE**-complete [DUN 05a]. The closely related problem 1-CONV [CHE 06, DUN 05a], discussed earlier in the context of checking conditions for guaranteed convergence with respect to full utility profiles, asks whether any sequence of individually rational 1-deals would culminate in an allocation with maximal social welfare. As we have seen, 1-CONV is **coNP**-complete. This has been a known result for the case where utility functions are represented as straight-line programs [DUN 05a], which we have extended here to a wider class of preference representation languages.

7.9. Bibliography

[ARR 02] ARROW K. J., SEN A. K., SUZUMURA K., Eds., *Handbook of Social Choice and Welfare*, vol. 1, North-Holland, 2002.

[BOR 02] BOROS E., HAMMER P. L., "Pseudo-Boolean Optimization", *Discrete Applied Mathematics*, vol. 123, num. 1–3, p. 155–225, 2002.

[CHE 04] CHEVALEYRE Y., ENDRISS U., ESTIVIE S., MAUDET N., "Multiagent Resource Allocation with k-additive Utility Functions", *Proc. DIMACS-LAMSADE Workshop on Computer Science and Decision Theory*, Annales du LAMSADE 3, 2004.

[CHE 05] CHEVALEYRE Y., ENDRISS U., MAUDET N., "On Maximal Classes of Utility Functions for Efficient one-to-one Negotiation", *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI-2005)*, Morgan Kaufmann Publishers, 2005.

5. As explained in section 7.6.3, complexity results apply to a particular decision problem with respect to a particular representation language for the input (here, the utility functions of the agents). However, in practice, it is the case that most of the results mentioned here have been established for several of the most common representations, so we omit giving such details here.

[CHE 06] CHEVALEYRE Y., DUNNE P. E., ENDRISS U., LANG J., LEMAÎTRE M., MAUDET N., PADGET J., PHELPS S., RODRÍGUEZ-AGUILAR J. A., SOUSA P., "Issues in Multiagent Resource Allocation", *Informatica*, vol. 30, p. 3–31, 2006.

[CRA 06] CRAMTON P., SHOHAM Y., STEINBERG R., Eds., *Combinatorial Auctions*, MIT Press, 2006.

[DUN 05a] DUNNE P. E., CHEVALEYRE Y., Negotiation can be as hard as planning: Deciding reachability properties of distributed negotiation schemes, Report num. ULCS-05-009, Department of Computer Science, University of Liverpool, 2005.

[DUN 05b] DUNNE P. E., WOOLDRIDGE M., LAURENCE M., "The Complexity of Contract Negotiation", *Artificial Intelligence*, vol. 164, num. 1–2, p. 23–46, 2005.

[END 06] ENDRISS U., MAUDET N., SADRI F., TONI F., "Negotiating Socially Optimal Allocations of Resources", *Journal of Artificial Intelligence Research*, vol. 25, p. 315–348, 2006.

[FAL 05] FALTINGS B., YOKOO M., "Introduction: Special Issue on Distributed Constraint Satisfaction", *Artificial Intelligence*, vol. 161, num. 1–2, p. 1–5, 2005.

[GAL 05] GALSTYAN A., CZAJKOWSKI K., LERMAN K., "Resource Allocation in the Grid with Learning Agents", *Journal of Grid Computing*, vol. 3, num. 1–2, p. 91–100, 2005.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and Intractability: A Guide to the Theory of **NP**-completeness*, W. H. Freeman and Co., 1979.

[GIO 04] GIOVANNUCCI A., RODRÍGUEZ-AGUILAR J. A., REYES A., NORIA F. X., CERQUIDES J., "Towards Automated Procurement via Agent-Aware Negotiation Support", *Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*, ACM Press, 2004.

[GRA 97] GRABISCH M., "k-order Additive Discrete Fuzzy Measures and their Representation", *Fuzzy Sets and Systems*, vol. 92, p. 167–189, 1997.

[KRA 01] KRAUS S., *Strategic Negotiation in Multiagent Environments*, MIT Press, 2001.

[LEM 99] LEMAÎTRE M., VERFAILLIE G., BATAILLE N., "Exploiting a Common Property Resource under a Fairness Constraint: A Case Study", *Proc. 16th International Joint Conference on Artificial Intelligence (IJCAI-1999)*, Morgan Kaufmann Publishers, 1999.

[MOU 88] MOULIN H., *Axioms of Cooperative Decision Making*, Cambridge University Press, 1988.

[NAR 05] NARUMANCHI M. V., VIDAL J. M., "Algorithms for Distributed Winner Determination in Combinatorial Auctions", *Proc. Agent-Mediated Electronic Commerce VII*, 2005.

[NEM 88] NEMHAUSER G. L., WOLSEY L. A., *Integer and Combinatorial Optimization*, John Wiley and Sons, 1988.

[NIS 06] NISAN N., "Bidding Languages for Combinatorial Auctions", CRAMTON P. et al., Eds., *Combinatorial Auctions*, MIT Press, 2006.

[PAR 04] PARKES D. C., SHNEIDMAN J., "Distributed Implementations of Vickrey-Clarke-Groves Mechanisms", *Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*, ACM Press, 2004.

[PET 06] PETCU A., FALTINGS B., PARKES D., "MDPOP: Faithful Distributed Implementation of Efficient Social Choice Problems", *Proc. 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2006)*, ACM Press, 2006.

[ROS 94] ROSENSCHEIN J. S., ZLOTKIN G., *Rules of Encounter*, MIT Press, 1994.

[ROT 98] ROTHKOPF M. H., PEKEČ A., HARSTAD R. M., "Computationally Manageable Combinational Auctions", *Management Science*, vol. 44, num. 8, p. 1131–1147, 1998.

[SAN 98] SANDHOLM T. W., "Contract Types for Satisficing Task Allocation: I Theoretical Results", *Proc. AAAI Spring Symposium: Satisficing Models*, 1998.

[SAN 99] SANDHOLM T. W., "Distributed Rational Decision Making", WEISS G., Ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, p. 201–258, MIT Press, 1999.

[SAN 02] SANDHOLM T. W., "Algorithm for Optimal Winner Determination in Combinatorial Auctions", *Artificial Intelligence*, vol. 135, num. 1–2, p. 1–54, 2002.

[SMI 80] SMITH R. G., "The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver", *IEEE Transactions on Computers*, vol. C-29, num. 12, p. 1104–1113, 1980.

[VID 06] VIDAL J. M., "Multiagent Coordination Using a Distributed Combinatorial Auction", *Proc. AAAI Workshop on Auction Mechanisms for Robot Coordination*, 2006.

[WOO 02] WOOLDRIDGE M., *An Introduction to MultiAgent Systems*, John Wiley and Sons, 2002.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 8

Worst-case Complexity of Exact Algorithms for **NP**-hard Problems

There has recently been a growing interest in the analysis of the worst-case complexity of many **NP**-hard problems. Unless $\mathbf{P} \neq \mathbf{NP}$, solving such problems requires super-polynomial time. Each problem in **NP** can be solved by a naive algorithm that exhaustively searches the solution space. However, for most of the problems, more refined algorithms with better, but still exponential-time complexity, are known.

Development of exact algorithms is mainly of theoretical interest but the existence of fast exponential procedures may also have practical importance. Today's computers are able to handle very efficiently moderate size instances of **NP**-hard problems. However, even though we can afford to run an exponential-time algorithm, polynomial-space complexity is a must.

Satisfiability, minimum graph coloring, maximum independent set and minimum vertex covering, minimum dominating set, etc., are among the problems that have received much attention in the context of exact algorithms. In this paper we study two very well-known hard combinatorial problems: the maximum cut (MAX-CUT) and the minimum dominating set restricted to graphs of maximum degree 3 (MIN 3-DOMINATING SET). For the former, we mainly focus ourselves on sparse graphs, i.e., on graphs having bounded maximum degree. We first use a technique based upon enumeration of cuts in a properly chosen subgraph of the input graph and then an extension of them in an optimal way to produce a cut for the whole instance. By

Chapter written by Federico DELLA CROCE, Bruno ESCOFFIER, Marcin KAMIŃSKI and Vangelis Th. PASCHOS.

means of this method we produce an exact algorithm for weighted MAX-CUT with improved upper complexity bound in the case of sparse graphs (section 8.1). Next, in section 8.2, we restrict ourselves to unweighted MAX-CUT in graphs of maximum degree 3. In this case we devise a search tree based exact algorithm. We exploit some simple and intuitive dominance conditions that efficiently prune the search-tree thus producing a fairly competitive upper complexity bound for the case settled. Finally, in section 8.3, we refine search tree's pruning by introducing a counting procedure, based upon the introduction of weights for the fixed data, which allows us to measure in a more precise way the progress made by the algorithm when it fixes them. We apply this method to MIN 3-DOMINATING SET.

Let $T(\cdot)$ be a super-polynomial and $p(\cdot)$ be a polynomial, both on integers. In what follows, using notations in [WŒG 03], for an integer n, we express running-time bounds of the form $p(n) \cdot T(n)$ as $O^*(T(n))$, the asterisk meaning that we ignore polynomial factors. We denote by $T(n)$ the worst-case time required to exactly solve the considered combinatorial optimization problem with n variables. We recall (see, for instance, [EPP 01]) that, if it is possible to bound above $T(n)$ by a recurrence expression of the type $T(n) \leqslant \sum T(n - r_i) + O(p(n))$, we have $\sum T(n-r_i)+O(p(n)) = O^*(\alpha(r_1, r_2, \ldots)^n)$ where $\alpha(r_1, r_2, \ldots)$ is the largest zero of the function $f(x) = 1 - \sum x^{-r_i}$.

8.1. MAX-CUT

Given an arbitrary graph with non-negative weights assigned to its edges, the MAX-CUT problem consists of finding a partition of vertices into two subsets such that the sum of the weights of all the edges that have endpoints in two different parts of the partition is maximized. In the unweighted case (i.e., all weights are positive and equal) the problem is often referred to as UNWEIGHTED MAX-CUT.

In what follows, we consider weighted, undirected, loopless graphs without multiple edges. In a graph $G = (V, E, w)$, V is the vertex set of cardinality $|V| = n$, E is the edge set of cardinality $|E| = m$ and $w : E \to \mathbb{R}^+ \cup \{0\}$ is a weight function that assigns a non-negative number w_{ij} to each edge ij of G.

The number of edges incident to a vertex in a graph is called the *degree* of the vertex. The maximum degree of all the vertices of a graph is called the *maximum degree* of the graph and denoted by Δ. The *average degree* of a graph is the sum of degrees of all vertices of the graph divided by the number of its vertices. The average degree is denoted by d; notice that $d = 2m/n$. Given a subset U of vertices of V, the subgraph induced by the vertices in U is denoted by $G[U]$.

A *cut* $C = (V_0, V_1)$ in a graph is a partition of its vertex set V into two disjoint subsets V_0 and V_1. The weight $w(C)$ of cut C is the sum of weights of all the edges

that have their endpoints in two different parts of the cut. Notice that the characteristic vector of one of the parts, for example V_0, uniquely determines the partition.

UNWEIGHTED MAX-CUT was one of the first problems whose **NP**-hardness was established. However, there are classes of graphs such as planar graphs, graphs with large girth [POL 95], cographs and graphs with bounded treewidth [BOD 00] that admit polynomial-time solution of this problem.

On the other hand, UNWEIGHTED MAX-CUT (and therefore MAX-CUT) remains **NP**-hard even if restricted to such classes as chordal, split, or 3-colorable graphs [BOD 00]. As shown in [YAN 78], the problem is **NP**-hard also for the class of graphs with bounded maximum degree Δ, if $\Delta \geqslant 3$ (for $\Delta \leqslant 2$ the problem becomes trivial).

The worst-case complexity of the maximum cut problem has been studied in a few papers, some of them dealing with the weighted and some with the unweighted case. The fastest algorithm for UNWEIGHTED MAX-CUT in arbitrary graphs was proposed by Williams in [WIL 04]. In fact, the algorithm calculates the number of solutions to an instance of the 2-CSP problem and employs interesting, non-standard techniques (2-CSP is the constraint satisfaction problem with two variables per clause). Used as a UNWEIGHTED MAX-CUT solver, the algorithm runs in time $O^*(2^{\omega n/3})$ but, unfortunately, requires exponential space of $O^*(2^{\omega n/3})$, where $\omega < 2.376$ is the matrix multiplication exponent (the product of two $k \times k$ matrices can be calculated in time $O(k^\omega)$). Whether there exists a polynomial-space algorithm that calculates UNWEIGHTED MAX-CUT and runs faster than the naive one of time complexity $O^*(2^n)$ is an open question listed in [WŒG 04].

More algorithms have been developed for sparse graphs. The upper bounds on their running times are given as linear functions of the number m of edges in the input graph. It makes them faster than the algorithms whose running time is bounded by a linear function of the number of vertices (like [WIL 04] or the naive algorithm) only if m is linearly bounded by n.

In [GRA 03] an algorithm solving UNWEIGHTED MAX-CUT (via MAX-2-SAT) in time $O^*(2^{m/3})$ was proposed by Gramm *et al.* The bound was then improved to $O^*(2^{m/4})$ by Fedin and Kulikov in [FED 06]. Their algorithm solves the maximum cut problem in a graph with integer weights on its edges. In a paper by Scott and Sorkin [SCO 03] (see also [SCO 04]), a faster algorithm for MAX-CUT, running in time $O^*(2^{\min\{(m-n)/2, m/5\}})$, was described. A recent paper by Kneis and Rossmanith [KNE 05b] offers an UNWEIGHTED MAX-CUT algorithm with running time $O^*(2^{m/5.217})$. All of those algorithms use polynomial space.

In this chapter, we develop a technique that seems to be a new approach to the MAX-CUT problem. The method consists of enumerating cuts in a subgraph H of G

and then extending them in an optimal way to cuts in G. The technique is applied to graphs with bounded maximum degree and to general graphs. In both cases, we obtain an exponential-time algorithm that uses polynomial space.

For some classes of graphs our algorithms offer the best running time known. In particular, we obtain the fastest known algorithm solving the MAX-CUT problem in the class of graphs with bounded maximum degree Δ, if $\Delta = 8, 9$. We also provide a polynomial-space MAX-CUT algorithm and a polynomial-space UNWEIGHTED MAX-CUT algorithm, that are the fastest known in the class of graphs with bounded maximum degree Δ, for $\Delta \geqslant 8$.

For weighted graphs with bounded maximum degree Δ, we present an algorithmic scheme that calculates a maximum cut. For fixed Δ, the algorithm runs in time $O^*(2^{(1-(2/\Delta))n})$ and polynomial space. For $\Delta \geqslant 8$, our algorithm is faster than the MAX-CUT algorithm from [SCO 04] and the UNWEIGHTED MAX-CUT algorithm from [KNE 05b]. It is slower than the exponential-space UNWEIGHTED MAX-CUT algorithm from [WIL 04] for $\Delta \geqslant 10$.

For general weighted graphs, we obtain an algorithm that calculates a maximum cut and runs in time $2^{mn/(m+n)}$. Our algorithm is faster than the MAX-CUT algorithm from [SCO 04] for $m > 4n$ and faster than the UNWEIGHTED MAX-CUT algorithm from [KNE 05b] for $m > 4.217n$. It is slower than the UNWEIGHTED MAX-CUT exponential-space algorithm in [WIL 04] for $m > \omega n/(3-\omega) > 3.808n$.

8.1.1. *Extending a partial partition of vertices*

In what follows, we will think of a partition as an assignment of $0-1$ values to the vertices of the graph. Let x_i be a Boolean variable which takes value 0, if $v_i \in V_0$, and 1, if $v_i \in V_1$. The weight of a cut in a graph $G = (V, E, w)$ can be expressed as a pseudo-Boolean function,

$$w(C) = \sum_{ij \in E} w_{ij} \left(x_i \overline{x_j} + \overline{x_i} x_j\right) = \sum_{i \in V} w_i x_i - 2 \sum_{ij \in E} w_{ij} x_i x_j, \qquad [8.1]$$

where $w_i = \sum_{\{i,j\} \in E} w_{ij}$. A *maximum cut* in a graph G is a cut of maximum weight.

Given a graph G as an input, the MAX-CUT problem consists of calculating a cut in G that maximizes [8.1].

Notice that it is enough to consider only connected graphs since if the graph is not connected, the MAX-CUT problem can be solved for each of its connected components separately.

It is easy to see that if the weights are restricted to be non-negative real numbers, the MAX-CUT problem can be solved in polynomial time for the class of bipartite graphs.

In this section we consider a modification of the MAX-CUT problem. Suppose that some of the vertices have already been partitioned into two subsets and now the problem is to find an optimal cut in the graph with respect to that pre-partition. We prove that if the graph induced by the vertices that have not yet been partitioned is bipartite, then the problem of finding an optimal extension of the partial partition can be solved in polynomial time. The algorithms presented in the following sections are based on this result.

Let $U \subset V$ be a subset of vertices of G such that the subgraph $G' = G[U']$ induced by the vertices in $U' = V \setminus U$ is bipartite. Also, let (U_0, U_1) be a partition of U into two subsets. Consider the problem of finding a partition (V_0, V_1) of V with $U_0 \subset V_0$ and $U_1 \subset V_1$ that maximizes [8.1].

The vertices in U have already been assigned to some parts of the cut, thus variables x_i, for $i \in U$, have their values fixed. There are four possible types of edges in the cut: edges with both endpoints in U, from U_0 to U', from U_1 to U' and with both endpoints in U'. The problem of finding an optimal extension of the pre-partition is now equivalent to maximizing the following pseudo-Boolean function:

$$\sum_{\substack{i \in U_0 \\ j \in U_1}} w_{ij} + \sum_{\substack{i \in U_0 \\ j \in U'}} w_{ij} x_j + \sum_{\substack{i \in U_1 \\ j \in U'}} w_{ij} \overline{x_j} + \sum_{\substack{i \in U' \\ j \in U'}} w_{ij} \left(x_i \overline{x_j} + \overline{x_i} x_j \right)$$

where all sums are taken over edges $ij \in E$ of the graph G. Setting:

$$c_j = \sum_{i \in U_0} w_{ij} - \sum_{i \in U_1} w_{ij} + \sum_{i \in U'} w_{ij}$$

where all sums are again taken over edges $ij \in E$, and omitting the constant term, the problem is equivalent to finding a maximum of the function:

$$\sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} x_i x_j \qquad [8.2]$$

where E' is the edge set of the bipartite graph G'. In other words, the problem of finding an optimal extension of the pre-partition can be stated as the following integer quadratic program:

$$\begin{array}{ll} \max & \displaystyle\sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} x_i x_j \\ \text{s.t.} & x_i \in \{0, 1\} \end{array} \qquad [8.3]$$

The standard linearization technique applied to [8.3] by introducing $y_{ij} = x_i x_j$, yields the following integer linear program:

$$\begin{array}{lrcl} \max & \displaystyle\sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} y_{ij} & & \\ \text{s.t.} & y_{ij} & \geqslant & x_i + x_j - 1 \\ & x_i & \in & \{0,1\} \\ & y_{ij} & \in & \{0,1\} \end{array} \qquad [8.4]$$

It is easy to see that [8.3] and [8.4] are equivalent. They have the same optimal value and there is an easy correspondence between their optimal solutions, namely $y_{ij} = x_i x_j$.

Having modeled the original quadratic problem [8.3] as an integer linear program, let us study the continuous relaxation of [8.4]:

$$\begin{array}{lrcl} \max & \displaystyle\sum_{j \in U'} c_j x_j - 2 \sum_{ij \in E'} w_{ij} y_{ij} & & \\ \text{s.t.} & y_{ij} & \geqslant & x_i + x_j - 1 \\ & x_i & \geqslant & 0 \\ & x_j & \leqslant & 1 \\ & y_{ij} & \geqslant & 0 \\ & y_{ij} & \leqslant & 1 \end{array} \qquad [8.5]$$

LEMMA 8.1.– *The constraint matrix of the linear program [8.5] is totally unimodular, i.e., the determinant of every square submatrix of it equals* 0 *or* ± 1.

Proof. Let A be the constraint matrix of [8.5]. It has $|U'| + |E'|$ columns and $2|U'| + 3|E'|$ rows and all its entries are either 0 or ± 1. Let B be an edge-vertex incidence matrix of G', with rows corresponding to edges and columns corresponding to vertices. Notice that B is a submatrix of A. Moreover, any submatrix of A that has two non-zero entries in every row and every column has to be a submatrix of B.

Take any square $k \times k$ submatrix of A. We will prove the lemma by induction on k. Clearly, the result holds for $k = 1$.

Now assume that all $(k-1) \times (k-1)$ submatrices of A are totally unimodular and consider a matrix M which is a $k \times k$ submatrix of A.

If all entries of any row or column of M are 0, then $\det(M) = 0$ and M is totally unimodular. If any row or column of M has a single non-zero element (± 1), then using

the expansion method for calculating determinants and the induction hypothesis, it is easy to see that $\det(M)$ is either 0 or ± 1, and A is totally unimodular.

Suppose that each row and each column of M has at least two non-zero entries. Hence, M must be a submatrix of B but, since B is an incidence matrix of a bipartite graph, so is M. It is possible to partition the columns of M into two parts, according to the partition of vertices of a bipartite graph. The sum of the columns in each part yields a unit vector (each edge of the bipartite subgraph has one endpoint in each part) and that implies linear dependence of M, therefore $\det(M) = 0$ and M is totally unimodular. ∎

THEOREM 8.1.– *Let $U \subset V$ be such that the subgraph $G' = G[U']$ induced by the vertices in $U' = V \setminus U$ is bipartite and (U_0, U_1) be a partition of U into two subsets, then the problem of finding a partition (V_0, V_1) of V with $U_0 \subset V_0$ and $U_1 \subset V_1$ that maximizes [8.1] is polynomial-time solvable.*

Proof. The problem of finding a partition (V_0, V_1) of V with $U_0 \subset V_0$ and $U_1 \subset V_1$ that maximizes [8.1] can be modeled as the integer quadratic program [8.3] which is equivalent to [8.4]. Total unimodularity of the constraint matrix of [8.5] (according to Lemma 8.1) implies the existence of an optimal $0-1$ solution of [8.5] and such a solution can be found in polynomial time (see for example [SCH 86]). Since the relaxation [8.5] of [8.4] has an optimal $0-1$ solution, [8.3] can be solved in polynomial time. ∎

Before we proceed to the next section, let us briefly describe the algorithmic technique we are going to apply. Given an induced bipartite subgraph $G[B]$ of G, we can enumerate all partitions of $V \setminus B$ and find an optimal extension of each in polynomial time (according to Theorem 8.1). The complexity of such a technique is $O^*(2^{|V \setminus B|})$ and it strongly depends on the size of the bipartite subgraph that has to be constructed.

8.1.2. *An algorithm for graphs with bounded maximum degree*

In this section we present and analyze an algorithmic scheme $\mathtt{A}(\Delta)$. For a fixed integer Δ ($\Delta \geqslant 3$), the scheme yields an algorithm whose input is a weighted graph $G = (V, E, w)$ of maximum degree Δ and whose output is a maximum cut in G with respect to the weight function w.

`Step 1.` If G is isomorphic to the complete graph on $\Delta + 1$ vertices, then let B be any pair of vertices and go to `Step 3`.

`Step 2.` Δ-color G. Let B be the union of the 2 largest color classes of the coloring.

`Step 3.` Enumerate all partitions of elements of $V \setminus B$ into two subsets (all $0-1$ assignments) and for each find an optimal extension of the partial partition.

`Step 4`. Find a cut C that has the largest weight among all those checked in `Step 3`. Return the cut C.

THEOREM 8.2.– *For a fixed integer Δ ($\Delta \geqslant 3$), algorithm* `A`(Δ) *calculates* MAX-CUT *in a graph G in time $O^*(2^{(1-(2/\Delta))n})$ and polynomial space.*

Proof. Let us first note that the algorithm indeed finds a maximum cut. It is clear that the induced subgraph $G[B]$ is bipartite. Therefore, any partition of $V \setminus B$ into two subsets can be extended to an optimal partition of V in polynomial time by Theorem 8.1. Clearly, by enumerating all partitions of $V \setminus B$ and then extending each in an optimal way, a maximum cut in G is found.

The enumeration of partitions in `Step 3` is the bottleneck of the algorithm; it needs exponential time $O^*(2^{|V \setminus B|})$. Other steps can be performed in linear time. It is clear for `Steps 1` and `4`, and the linear time algorithm for `Step 2` is given in [LOV 75]. Note that the algorithm can be implemented in such a way that each step uses only polynomial space. In particular, in `Step 3` we need to store only the current best solution.

Suppose that the input graph is isomorphic to the complete graph on $\Delta+1$ vertices. The number of partitions that are enumerated in `Step 3` is 2^{n-2} but since $\Delta = O(n)$ the claimed running time follows.

Now suppose that the input graph G is not isomorphic to the complete graph on $\Delta + 1$ vertices. Then, according to Brooks' theorem G is Δ colorable [LOV 75]. Clearly, the union of two largest color classes has size at least $2n/\Delta$ and $|V \setminus B| \leqslant n(1 - (2/\Delta))$. The number of partitions that are enumerated in `Step 3` is $O^*(2^{(1-(2/\Delta))n})$ and the claimed running time follows. ∎

8.1.3. *An algorithm for general graphs*

Let us note that in the algorithm presented in the previous section, the assumption of bounded maximum degree is needed only to obtain an induced bipartite graph. Now we relax this assumption and study the complexity of the method in general graphs. Let us formalize that as Algorithm `B`. The input of `B` is a weighted graph $G = (V, E, w)$ and the output is a maximum cut in G with respect to the weight function w.

`Step 1`. Find a maximal independent set I_0 in G.

`Step 2`. Find a maximal independent set I_1 in $G[V \setminus I_0]$. Let B be the union of I_0 and I_1.

`Step 3.` Enumerate all partitions of elements of $V \setminus B$ into two subsets (all $0-1$ assignments) and for each find an optimal extension of the partial partition.

`Step 4.` Find a cut C that has the largest weight among those checked in `Step 3`. Return the cut C.

To complete the description of the algorithm, we need to provide a procedure that finds an induced bipartite subgraph in `Steps 1` and `2`.

From Turan's theorem follows that the size of a maximum independent set is at least $n/(d+1)$ and as shown in [SAK 03], there is a linear-time algorithm that constructs an independent set of at least that size. As the time complexity of `B` depends on $|B|$, we need to give a lower bound on the size of the bipartite subgraph B.

CLAIM. The set B of vertices constructed in `Step 2` of algorithm `B` has size at least $2/(d+2)$.

Proof. Let $i = |I_0|$ and m' be the number of edges of the subgraph $G[I_0 \cup I_1]$. If $i \geqslant 2n/(d+2)$, then $|B| \geqslant 2n/(d+2)$ and the claim follows. Suppose that $i < 2n/(d+2)$. The average degree d' in the graph $G[V - I_0]$ is $d' = 2(m-m')/(n-i)$. Note that $m' \geqslant n-i$, since I_0 is an independent set. Hence, $d' \leqslant 2n/(n-i) - 2$ and since $i < 2n/(d+2)$, we have $d' < d$. It follows that $|B| = i + (n-i)/(d'+1) \geqslant 2n/(d+2)$. ∎

Having established the lower bound on the size of B, we can claim the running time of Algorithm `B`. Notice that $2/(d+2) = n/(n+m)$ and $n - |B| \leqslant mn/(m+n)$.

THEOREM 8.3.– *Algorithm* `B` *calculates* MAX-CUT *in a graph* G *with* n *vertices and* m *edges in time* $O^*(2^{mn/(m+n)})$*, and polynomial space.*

The proof of this theorem is similar to the proof of Theorem 8.2 and will be omitted.

8.2. Pruning the search tree by dominance conditions: the case of MAX-CUT-3

Among the existing exact methods for solving **NP**-hard problems, search tree algorithms and in particular branch-and-bound approaches have been widely applied. A branch-and-bound algorithm builds and explores a search tree, thus enumerating the solution space. In order to reduce the calculational burden, several techniques can be applied to prune some branches of the search tree and to avoid the enumeration of known non-optimal solutions. A usual technique is to consider dominance conditions

while branching from a node. Dominance conditions can be typically derived by comparing two nodes of the search tree, namely two partial solutions of the given problem where the two nodes share some common features.

In this section, we revisit MAX-CUT and study the application of dominance conditions within exact search tree algorithms for this problem. We show that taking into account such conditions in the study of the complexity of exact algorithms leads to generally fine combinatorial analyses deriving non-trivial upper complexity bounds for simple and intuitive search tree optimal algorithms. Another advantage of the methods described in this section is that the presented algorithms have polynomial space-complexities.

Here, we consider the restricted case where all vertices have maximum degree $d_{\max} = 3$ (hence, $m \leqslant 3n/2$).

We denote by d_j the degree of vertex v_j and by $N(v_j)$ the set of vertices (neighborhood) adjacent to v_j (in other words, $d_j = |N(v_j)|$).

The best known and most recent upper time-bound for MAX-CUT-3 is, to our knowledge, that of [KNE 05a] of $O^*(1.123^n)$ but the corresponding algorithm uses exponential space. A polynomial space exact algorithm for MAX-CUT-3 is presented in [KNE 05a] providing a time-bound of $O^*(1.221^n)$. In what follows, we propose a search tree-based algorithm with running time $O^*(1.2920^n)$ which, even though dominated by the bound in [KNE 05a], already dominates the bounds, for instance, of [FED 06] and [GRA 03] when applied to MAX-CUT-3.

8.2.1. *Dominance conditions*

We assume, without loss of generality, that vertex v_1 is assigned to V_1. Also, with respect to the worst-case analysis, we assume without loss of generality that the input-graph G is connected. The following straightforward lemma holds.

LEMMA 8.2.– *There exists at least one optimal solution of* MAX-CUT *where, for any vertex $v_j \neq v_1$ assigned to V_1 (resp., V_2), at least $(d_j + 1)/2$ vertices $v_i \in N(v_j)$, for d_j odd, and at least $d_j/2$ vertices $v_i \in N(v_j)$, for d_j even, are assigned to V_2 (resp., V_1).*

Proof. For d_j odd, if less than $(d_j + 1)/2$ vertices are assigned to V_2 (resp., V_1), then moving j to V_2 (resp., V_1), would improve the solution. On the other hand, for d_j even, if less than $d_j/2$ vertices are assigned to V_2 (resp., V_1), then moving j to V_2 (resp., V_1) would again improve (or at least not worsen) the solution. ∎

REMARK 8.1.– For d_j even and the vertices $v_i \in N(v_j)$ equally distributed between V_1 and V_2, it is immaterial to assign v_j to V_1 or V_2.

Consider solving MAX-CUT by means of a search tree approach. Suppose that at some point a branching is considered related to a vertex v_j and that all (nearly all) of its adjacent vertices v_i ($v_i \in N(v_j)$) have already been assigned. We denote by $S_j(V_k)$ the set of vertices adjacent to vertex v_j and assigned to set V_k, $j = 1, \ldots, n$, $k = 1, 2$; namely, $S_j(V_k) = \{i : v_i \in N(v_j) \cap V_k\}$. Then, using Lemma 8.2, the following lemma holds.

LEMMA 8.3.– *Consider any vertex v_j such that:*

1) all $v_i \in N(v_j)$ have already been assigned and d_j is odd;

2) all $v_i \in N(v_j)$ have already been assigned and d_j is even;

3) all but one $v_i \in N(v_j)$ have already been assigned and d_j is even.

For all of the cases above, there exists at least one optimal solution with the assignment of v_j uniquely determined as follows:

1) (Condition 1 holds) if $|S_j(V_1)| \geqslant (d_j + 1)/2$, then v_j is assigned to V_2, otherwise it is assigned to V_1;

2) (Condition 2 holds):

a) if $|S_j(V_1)| > d_j/2$, then v_j is assigned to V_2,

b) if $|S_j(V_1)| < d_j/2$, then v_j is assigned to V_1,

c) if $|S_j(V_1)| = |S_j(V_2)| = d_j/2$, it is immaterial to assign v_j to V_1 or V_2;

3) (Condition 3 holds) if $|S_j(V_1)| \geqslant d_j/2$, then v_j is assigned to V_2, otherwise it is assigned to V_1.

Proof. Due to Lemma 8.2 and Remark 8.1, the proof of items 1 and 2 above, is immediate. For item 3, let $v_k \in N(v_j)$ be the unassigned vertex. If $|S_j(V_1)| \geqslant d_j/2$, whatever the assignment of v_k, either the condition of item 2a, or the condition of item 2c hold and hence there exists at least an optimal solution with v_j assigned to V_2. Analogously, if $|S_j(V_1)| \leqslant d_j/2 - 1$, whatever the assignment of v_k, either the condition of case 2b, or the condition of case 2c hold and hence there exists at least an optimal solution with v_j assigned to V_1. ∎

REMARK 8.2.– Since the graph is connected, it is always possible to devise a search tree algorithm in which we always branch on a vertex v_j that is adjacent to at least another vertex v_i which has already been assigned to one of the sets of the partition (V_1, V_2).

Consider the following search tree algorithm for MAX-CUT, denoted by `SOLVE-MAX-CUT` where, at any node of the tree, a decision is taken on the assignment of a vertex either to V_1 or to V_2:

– select arbitrarily vertex v_1 and assign it to V_1;

– apply a search tree algorithm to assign the remaining vertices according to the following rule: select for branching the unassigned vertex whose branching induces

the minimum worst-case complexity and, in case of a tie, the unassigned vertex with minimum degree (in the case of a further tie select the vertex with smallest index).

When we consider for branching in SOLVE-MAX-CUT a given vertex v_j and $d_{\max} = 3$, either $d_j \leqslant 2$ and, due to Lemma 8.3 and Remark 8.2, v_j can be assigned without branching, or $d_j = 3$. The following dominance condition holds.

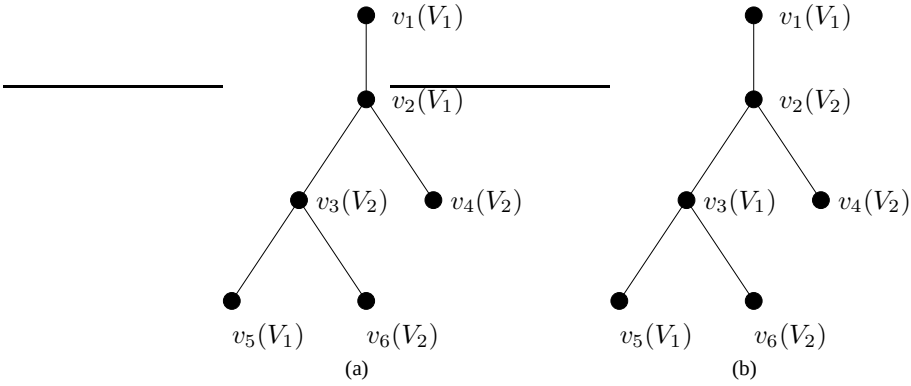


Figure 8.1. *Comparing two configurations with six vertices*

LEMMA 8.4.– *Consider six vertices $v_1, \ldots, v_6$ connected as in figure 8.1(a). Then, swapping the assignment of vertices v_2 and v_3 leads to an equivalent solution, i.e., configuration 8.1(a) can be substituted by configuration 8.1(b).*

Proof. It is immediate to see that for both configurations exactly three edges belong to the cut. ∎

8.2.2. *The worst-case upper-time bound for* MAX-CUT-*3*

We now prove the following result dealing with MAX-CUT-3.

PROPOSITION 8.1.– *Algorithm SOLVE-MAX-CUT optimally solves* MAX-CUT-*3 with worst-case time-complexity $O^*(1.2920^n)$.*

Proof. The relevant branching cases for a given vertex v_j in the application of SOLVE-MAX-CUT are those with $d_j = 3$, otherwise no branching occurs. Let v_i, v_k and v_l be the corresponding adjacent vertices. Recall that, from Remark 8.2, at least one vertex v_i has already been assigned. Also, from Lemma 8.3, no branching occurs if all adjacent vertices v_i, v_k and v_l have already been assigned. Finally, from

Lemma 8.2, no branching occurs if two of the adjacent vertices have already been assigned to the same set of the partition (V_1, V_2). Then, the following relevant cases may occur:

1) $d_j = 3$, two vertices v_i, v_k adjacent to v_j have already been assigned to different sets of the partition (V_1, V_2), while the third adjacent vertex v_l has not yet been assigned. Then, the following exhaustive subcases may hold:

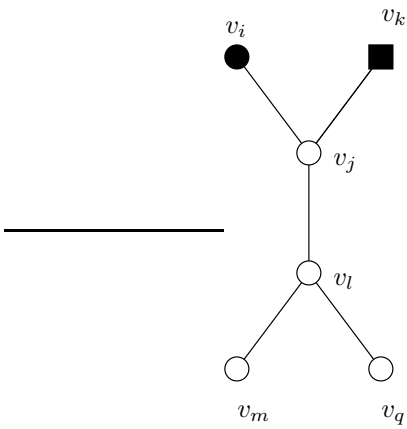


Figure 8.2. *Case 1a of Proposition 8.1*

a) $d_l = 3$, v_l is adjacent to vertex v_j and to other two vertices v_m, v_q both unassigned (see Figure 8.2, where black circles represent vertices assigned to V_1, black rectangles represent vertices assigned to V_2 and white circles represent yet unassigned vertices). Due to Lemma 8.2, if at least one of v_m, v_q is assigned to V_1 (resp., V_2), we can arbitrarily assign v_l to V_2 (resp., V_1) and v_j to V_1 (resp., V_2). Alternatively, both vertices v_m, v_q are assigned to V_2 (resp., V_1) and hence we can assign v_l to V_1 (resp., V_2) and v_j to V_2 (resp., V_1). This can be seen as a binary branching where either 2 vertices (v_j, v_l), or 4 vertices (v_j, v_l, v_m, v_q) are assigned. Then, $T(n) \leqslant T(n-2)+T(n-4)+O(p(n))$, where the terms $T(n-2)$ and $T(n-4)$ measure the time for solving the same case with $n-2$ and $n-4$ unassigned vertices, respectively. Correspondingly, we have $T(n) = O^*(1.2721^n)$.

b) $d_l = 3$, v_l is adjacent to vertex v_j and to other two vertices v_m, v_q, where at least one of them (for example v_m, where v_m may possibly coincide with v_i or v_k) has already been assigned (Figure 8.3). Then, whatever the assignment of v_q, if v_m is assigned to V_2 (resp., V_1), v_l can be assigned to V_1 (resp., V_2) and v_j to V_2 (resp., V_1) without branching.

c) $d_l = 2$ (Figure 8.4). Then, v_l is adjacent to vertex v_j and to another vertex v_m which has not yet been assigned (or else v_l would have already been assigned

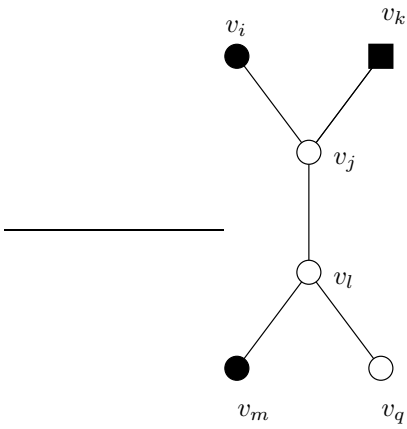


Figure 8.3. *Case 1b of Proposition 8.1*

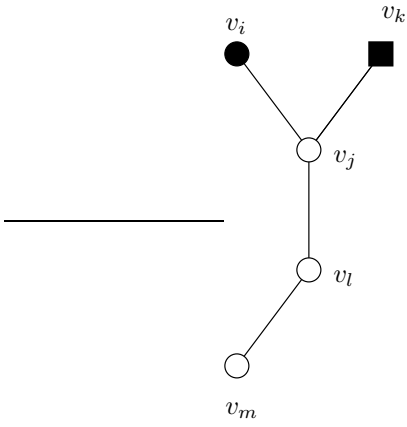


Figure 8.4. *Case 1c of Proposition 8.1*

earlier without branching). If we assign v_m to V_2 (resp., V_1), then v_l must be assigned to V_1 (resp., V_2) and v_j to V_2 (resp., V_1). In other words, we can branch on vertex v_m and, for both branches, correspondingly, fix the assignment of vertices v_j, v_l, v_m all together. This can be seen as a binary branching where, in both cases, three vertices (v_j, v_l, v_m) are assigned and hence $T(n) \leqslant 2T(n-3)+O(p(n))$, i.e., $T(n) = O^*(1.2599^n)$.

d) $d_l = 1$ (Figure 8.5). Then, v_j can be assigned to V_1 and v_l to V_2 without branching (actually, it is immaterial to assign v_j to V_1 or V_2, provided that v_l is assigned to the opposite set of the partition).

2) $d_j = 3$, a vertex v_i adjacent to v_j has already been assigned to V_1 (resp., V_2),

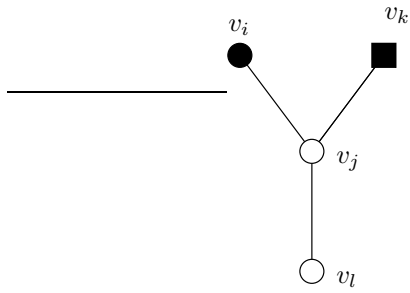


Figure 8.5. *Case 1d of Proposition 8.1*

while the other two adjacent vertices v_k and v_l have not yet been assigned. We assume, without loss of generality, that $d_k \leqslant d_l$. The following subcases must be considered:

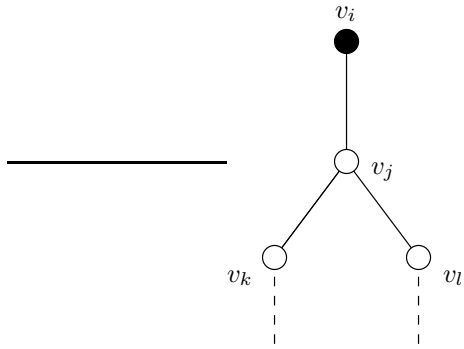


Figure 8.6. *Case 2a of Proposition 8.1*

a) $1 \leqslant d_k \leqslant d_l \leqslant 2$ (Figure 8.6). A branching on vertex v_j can be applied. Either v_j is assigned to V_1 (resp., V_2) and, correspondingly, vertices v_k and v_l must be assigned, due to Lemma 8.2, to V_2 (resp., V_1), or v_j is assigned to V_2 (resp., V_1) and, correspondingly, vertices v_k and v_l must be assigned, due to Lemma 8.2, to V_1 (resp., V_2). This can be seen as a binary branching where, in both cases, 3 vertices (v_j, v_k, v_l) are fixed and the same time-complexity $O^*(1.2599^n)$ of case 1c holds.

b) $1 \leqslant d_k \leqslant 2$, $d_l = 3$, v_k and v_l are not adjacent and one of the vertices (v_m, v_q) adjacent to vertex v_l has already been assigned while the other has not yet been assigned (Figure 8.7). We assume, without loss of generality, that v_m has already been assigned. Notice that, if v_m has been assigned to V_2 (resp., V_1), no branching occurs, due to Lemma 8.4. We assume then that v_m has already been assigned to V_1 (resp., V_2). A branching on vertex v_j can be applied. If v_j is assigned to V_1 (resp., V_2),

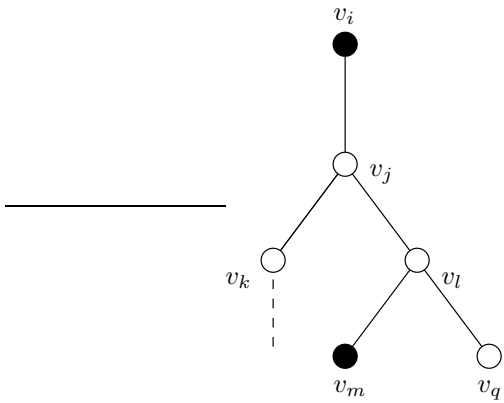


Figure 8.7. *Case 2b of Proposition 8.1*

vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1), while vertex v_q must be assigned (due to Lemma 8.4) to V_1 (resp., V_2). Otherwise, v_j is assigned to V_2 (resp., V_1) and, correspondingly, v_k can be assigned to V_1. This can be seen as a branching where, either 2 vertices (v_j, v_k), or 4 vertices (v_j, v_k, v_l, v_q) are assigned and the same time-complexity $O^*(1.2721^n)$ of case 1a holds.

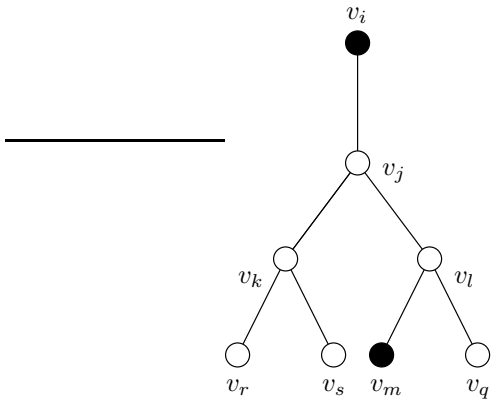


Figure 8.8. *Case 2c of Proposition 8.1*

c) $d_k = d_l = 3$, v_k and v_l are not adjacent and one of the vertices (v_m, v_q) adjacent to vertex v_l has already been assigned, while the other has not yet been assigned (Figure 8.8). Also, the vertices (v_r, v_s) adjacent to vertex v_k have not yet been assigned. We assume, without loss of generality, that v_m has already been assigned. Note that, if v_m has been assigned to V_2 (resp., V_1), no branching occurs due to Lemma 8.4. We assume, then, that v_m has already been assigned to V_1 (resp., V_2). A branching on

vertex v_j can be applied. If v_j is assigned to V_1 (resp., V_2), vertices v_k, v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1) and vertices v_q, v_r, v_s must be assigned (due to Lemma 8.4) to V_2 (resp., V_1). Otherwise, v_j is assigned to V_2 (resp., V_1). This can be seen as a branching where, either 1 vertex (v_j) is assigned, or 6 vertices ($v_j, v_k, v_l, v_q, v_r, v_s$) are assigned. Then, $T(n) \leqslant T(n-1) + T(n-6) + O(p(n))$, i.e., $T(n) = O^*(1.2852^n)$.

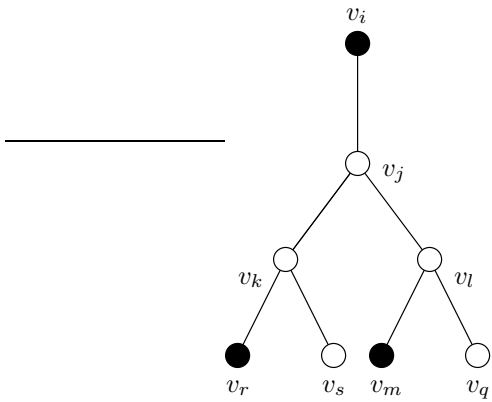


Figure 8.9. *Case 2d of Proposition 8.1*

d) $d_k = d_l = 3$, v_k and v_l are not adjacent and one of the vertices (v_m, v_q) adjacent to vertex v_l has already been assigned while the other has not yet been assigned (Figure 8.9). We assume, without loss of generality, that v_m has already been assigned. Notice that, if v_m has been assigned to V_2 (resp., V_1), no branching occurs, due to Lemma 8.4. We assume, then, that v_m has already been assigned to V_1 (resp., V_2). Also, v_k is adjacent (apart from v_j) to two vertices v_r and v_s where vertex v_r has already been assigned, while v_s has not yet been assigned. Notice that, if v_r has been assigned to V_2 (resp., V_1), no branching occurs, due to Lemma 8.4. We assume, then, that v_r has already been assigned to V_1 (resp., V_2). Finally, notice that v_m and v_r may well coincide. A composite branching can be applied. If v_j is assigned to V_1 (resp., V_2), then v_k, v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1) and v_q, v_s must be assigned (due to Lemma 8.4) to V_1 (resp., V_2). Otherwise, v_j is assigned to V_2 (resp., V_1): then, either v_q is assigned to V_1 (resp., V_2) and, correspondingly, v_l is assigned to V_2 (resp., V_1), v_k to V_1 (resp., V_2) and v_s to V_2 (resp., V_1); or v_q is assigned to V_2 (resp., V_1) and v_l is assigned to V_1 (resp., V_2). However, for this latter case a branching on vertex k can be applied where two of its adjacent vertices (v_j, v_r) have already been assigned and, in the worst case, subcase 1a holds. To conclude, this can be seen as a composite branching where either 5 vertices $(v_j, v_k, v_l, v_q, v_s)$ are assigned or 5 vertices $(v_j, v_k, v_l, v_s, v_q)$ are assigned or three vertices (v_j, v_l, v_q) are assigned and a branching of type 1a on vertex v_k with $n' = n - 3$ variables holds. Then, $T(n) \leqslant 2T(n-5) + 2T(n'-3) + O(p(n)) = 3T(n-5) + T(n-7) + O(p(n))$.

Correspondingly, we have $T(n) = O^*(\alpha^n)$, i.e., $\alpha \approx 1.2886$, implying a time complexity of $O^*(1.2920^n)$.

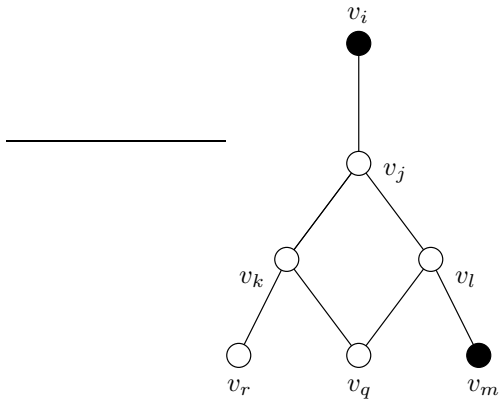


Figure 8.10. *Case 2e of Proposition 8.1*

e) $d_k = d_l = 3$, v_k and v_l are not adjacent and one of the vertices (v_m, v_q) adjacent to vertex v_l has already been assigned while the other has not yet been assigned (Figure 8.10). We assume, without loss of generality, that v_m has already been assigned. Notice that, if v_m has been assigned to V_2 (resp., V_1), no branching occurs, due to Lemma 8.4. We assume then that v_m has already been assigned to V_1 (resp., V_2). Also, v_k is adjacent (apart from v_j) to v_q and to another unassigned vertex v_r. A composite branch, first on vertex v_j and then on vertex v_q can be applied. If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1) and vertices v_q, v_r must be assigned (due to Lemma 8.4) to V_1 (resp., V_2). Otherwise, v_j is assigned to V_2 (resp., V_1) and, if v_q is assigned to V_2 (resp., V_1), then v_k and v_l must be assigned to V_1 (resp., V_2), otherwise v_q is assigned to V_1 (resp., V_2) and, consequently, v_l must be assigned to V_2 (resp., V_1), v_k to V_1 (resp., V_2) and v_r to V_2 (resp., V_1). This can be seen as a branching with three children nodes where, either 5 vertices $(v_j, v_k, v_l, v_q, v_r)$, or 4 vertices (v_j, v_k, v_l, v_q), or 5 vertices $(v_j, v_k, v_l, v_q, v_r)$ are assigned. Then, $T(n) \leqslant T(n-4) + 2T(n-5) + O(p(n))$, i.e., $T(n) = O^*(1.2672^n)$.

f) $d_k = d_l = 3$, v_k and v_l are not adjacent and one of the vertices (v_m, v_q) adjacent to vertex v_l has already been assigned while the other has not yet been assigned (Figure 8.11). We assume, without loss of generality, that v_m has already been assigned. Notice that, if v_m has been assigned to V_2 (resp., V_1), no branching occurs, due to Lemma 8.4. We assume, then, that v_m has already been assigned to V_1 (resp., V_2). Also, v_k is adjacent (apart from v_j) to v_q and to another vertex v_r that has already been assigned. Notice that, if v_r has been assigned to V_2 (resp., V_1), no branching occurs, due to Lemma 8.4. We assume, then, that v_r has already been assigned to V_1 (resp., V_2). Finally, notice that v_m and v_r may well coincide. We observe

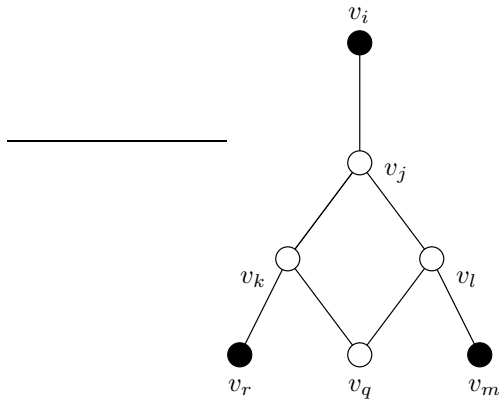


Figure 8.11. *Case 2f of Proposition 8.1*

that v_j, v_q must be assigned to the same set of the partition. Indeed, if v_j is assigned to V_1 (resp., V_2), v_q cannot be assigned to V_2 (resp., V_1) due to Lemma 8.4; on the other hand, if v_q is assigned to V_1 (resp., V_2), then v_k and v_l must be assigned to V_2 (resp., V_1) and, correspondingly, v_j cannot be assigned to V_2 (resp., V_1). Summarizing, either v_j and v_q are assigned to V_1 and, correspondingly, v_k, v_l are assigned to V_2, or v_j and v_q are assigned to V_2 and, correspondingly, v_k, v_l are assigned to V_1. This can be seen as a binary branching where, in both cases, 4 vertices (v_j, v_k, v_l, v_q) are fixed. Then, $T(n) \leqslant 2T(n-4) + O(p(n))$, i.e., $T(n) = O^*(1.1892^n)$.

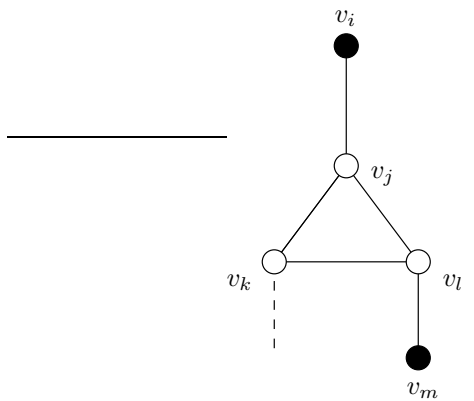


Figure 8.12. *Case 2g of Proposition 8.1*

g) $1 \leqslant d_k \leqslant d_l = 3$, v_l is adjacent to v_k and to another vertex v_m (that may eventually coincide with v_i) that has already been assigned (Figure 8.12). If v_m has been assigned to V_2 (resp., V_1), no branching occurs, as v_j must be assigned to V_2

(resp., V_1), otherwise v_k, v_l and v_m would all be assigned to V_2 (resp., V_1), thus violating Lemma 8.2. We assume, then, that v_m has already been assigned to V_1 (resp., V_2). But then a branching on v_k can be applied. If v_k is assigned to V_1 (resp., V_2), vertices v_j and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1). If v_k is assigned to V_2 (resp., V_1), then vertices v_j, v_l must be assigned to different sets of the partition (V_1, V_2) but it is immaterial to assign v_j to V_1 and v_l to V_2 or vice versa. To conclude, this can be seen as a binary branching where, in both cases, 3 vertices (v_j, v_k, v_l) are assigned and the same time-complexity $O^*(1.2599^n)$ of case 1c holds.

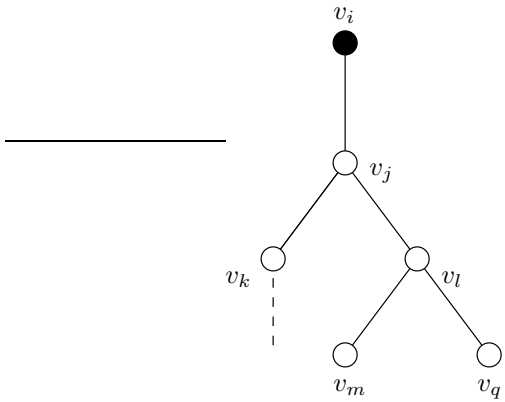


Figure 8.13. *Case 2h of Proposition 8.1*

h) $1 \leqslant d_k \leqslant 2$, $d_l = 3$, vertex v_l is not adjacent to v_k but is adjacent to vertices v_m and v_q which are both unassigned (Figure 8.13). A branching on vertex v_j can be applied. If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1), and vertices v_m and v_q must be assigned to V_1 (resp., V_2) as they cannot be assigned to different sets of the partition (due to Lemma 8.4), nor can they both be assigned to V_2 (resp., V_1) due to Lemma 8.2 applied to vertex v_l. Otherwise, v_j is assigned to V_2 (resp., V_1) and, correspondingly, v_k can be assigned to V_1 (resp., V_2). This can be seen as a branching where either 2 vertices (v_j, v_k) or 5 vertices $(v_j, v_k, v_l, v_m, v_q)$ are assigned. Then, $T(n) \leqslant T(n-2) + T(n-5) + O(p(n))$, i.e., $T(n) = O^*(1.1939^n)$.

i) $d_k = d_l = 3$, vertex v_l is not adjacent to v_k but is adjacent to vertices v_m and v_q which are both unassigned (Figure 8.14). Vertex v_k is adjacent to other two vertices v_r, v_s which are both unassigned. A branching on vertex v_j can be applied. If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1), and vertices v_m, v_q, v_r, v_s must be assigned to V_1 (resp., V_2) due to Lemma 8.4. Otherwise, v_j is assigned to V_2 (resp., V_1). This can be seen as a branching where, either 1 vertex (v_j) is assigned, or 7 vertices $(v_j, v_k, v_l, v_m, v_q, v_r, v_s)$ are

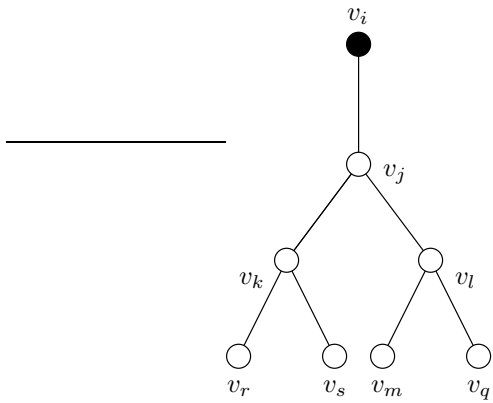


Figure 8.14. *Case 2i of Proposition 8.1*

assigned. Then, $T(n) \leqslant T(n-1)+T(n-7)+O(p(n))$, i.e., $T(n) = O^*(1.2555^n)$.

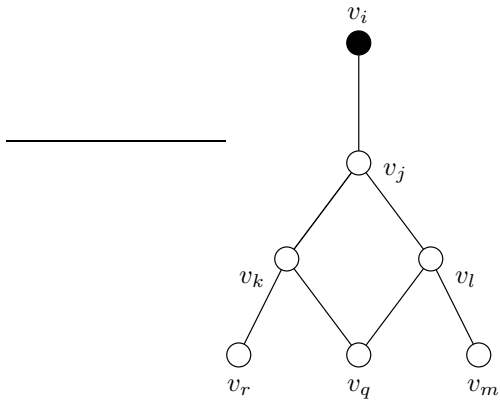


Figure 8.15. *Case 2j of Proposition 8.1*

j) $d_k = d_l = 3$, vertex v_l is not adjacent to v_k but is adjacent to vertices v_m and v_q which are both unassigned (Figure 8.15). Vertex v_k is adjacent to vertex v_q and to another vertex v_r which are also unassigned. A branching on vertex v_j can be applied. If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1), and vertices v_m, v_q, v_r must be assigned to V_1 (resp., V_2) due to Lemma 8.4. Otherwise, v_j is assigned to V_2 (resp., V_1). This can be seen as a branching where, either 1 vertex (v_j) is assigned, or 6 vertices ($v_j, v_k, v_l, v_m, v_q, v_r$) are assigned. Then, $T(n) \leqslant T(n-1)+T(n-6)+O(p(n))$ and the same time-complexity $O^*(1.2852^n)$ of case 2c holds.

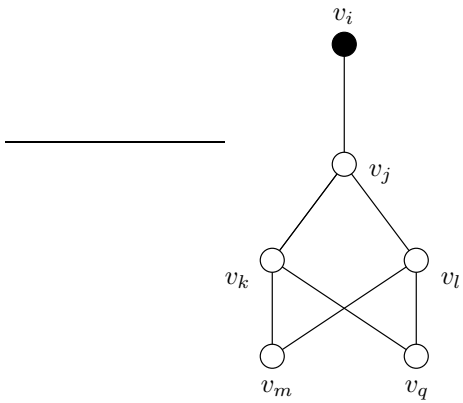


Figure 8.16. *Case 2k of Proposition 8.1*

k) $d_k = d_l = 3$, vertex v_l is not adjacent to v_k but is adjacent to vertices v_m and v_q which are both unassigned (Figure 8.16). Vertex v_k is also adjacent to vertices v_m and v_q. Notice that v_j and v_k cannot be assigned to the same set of the partition, otherwise v_l, v_m, v_q would all be assigned to the other set of the partition, thus violating Lemma 8.2. Analogously, v_j and v_l cannot be assigned to the same set of the partition. But then, if v_j is assigned to V_1 (resp., V_2), correspondingly, v_k, v_l are assigned to V_2 (resp., V_1) and v_m, v_q are assigned to V_1 (resp., V_2). Otherwise, v_j is assigned to V_2 (resp., V_1), v_k, v_l are assigned to V_1 (resp., V_2) and v_m, v_q are assigned to V_2 (resp., V_1). This can be seen as a binary branching where, in both cases, 5 vertices (v_j, v_k, v_l, v_m, v_q) are assigned. Then, $T(n) \leqslant 2T(n-5) + O(p(n))$, i.e., $T(n) = O^*(1.1487^n)$.

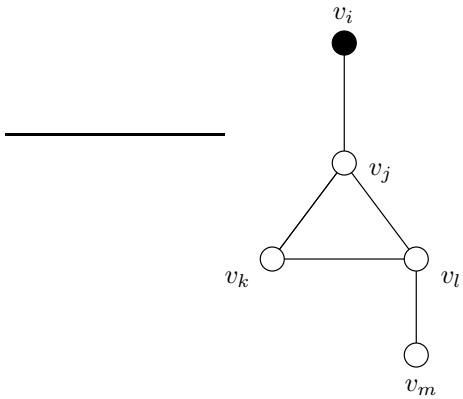


Figure 8.17. *Case 2l of Proposition 8.1*

l) $d_k = 2, d_l = 3$, v_l is adjacent to v_k and to another vertex v_m that has not yet been assigned (Figure 8.17). If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1) and, correspondingly, vertex v_m must be assigned to V_1 (resp., V_2). But an equivalent solution is obtained by simply swapping the assignment of vertices v_j, v_k. Hence, for this subcase, v_j can be assigned to V_2 (resp., V_1) without branching.

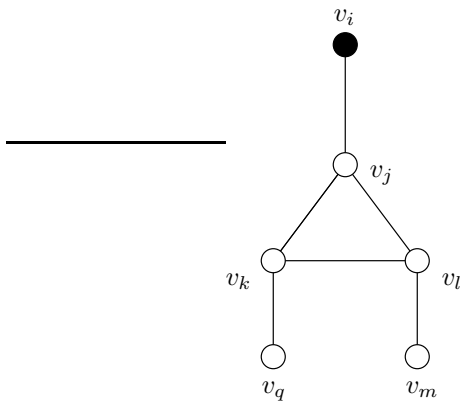


Figure 8.18. *Case 2m of Proposition 8.1*

m) $d_k = d_l = 3$, v_k is adjacent to v_l and to an unassigned vertex v_q, v_l is adjacent to v_k and to an unassigned vertex v_m, $v_m \neq v_q$ (Figure 8.18). If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1) and, correspondingly, vertices v_m and v_q must be assigned to V_1 (resp., V_2). But an equivalent solution is obtained by simply swapping the assignment of vertices v_j, v_k. Hence, for this subcase, v_j can be assigned to V_2 (resp., V_1) without branching.

n) $d_k = d_l = 3$, v_k is adjacent to v_l and both v_k and v_l are adjacent to another unassigned vertex v_m (Figure 8.19). If v_j is assigned to V_1 (resp., V_2), vertices v_k and v_l must be assigned (due to Lemma 8.2) to V_2 (resp., V_1), and, correspondingly, vertex v_m must be assigned to V_1 (resp., V_2). But an equivalent solution is obtained by simply swapping the assignment of vertices v_j, v_k. Hence, for this subcase, v_j can be assigned to V_2 (resp., V_1) without branching.

To conclude, the global worst-case complexity for MAX-CUT-3 with maximum degree three is $O^*(1.2920^n)$. ∎

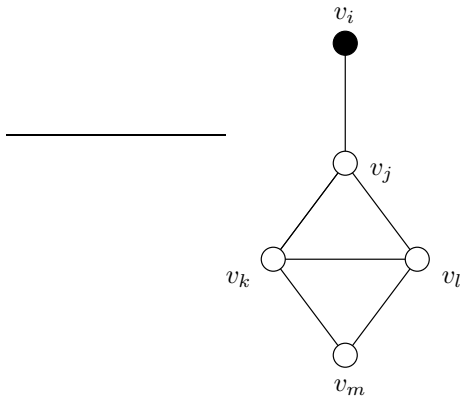


Figure 8.19. *Case 2n of Proposition 8.1*

8.3. A more careful analysis for pruning: the case of MIN 3-DOMINATING SET

We consider in this section the MIN 3-DOMINATING SET problem. Given a graph $G = [V, E]$, a dominating set of G is a subset $V' \subseteq V$ such that, for any $v \in V$, either $v \in V'$ or there exists $v' \in V'$ such that $(v, v') \in E$. In the MIN 3-DOMINATING SET problem, the input graph has degree 3, and the goal is to find a dominating set of minimum size. While the case of a graph of degree 2 is polynomial, MIN 3-DOMINATING SET is **NP**-hard [GAR 79].

Recently, a lot of exponential algorithms have been proposed for dominating problems. For the general case (no restriction on the degree), the first algorithm leading to a non-trivial worst case bound has been given in [FOM 04]. Further improvements have been obtained; the best actual bound is $O^*(1.5137^n)$ [FOM 05]. Considering the restriction to graphs of degree 3, algorithms with worst case complexity of $O^*(1.5144^n)$ [FOM 04] and $O^*(1.4143^n)$ [KNE 05a] have been devised. Very recently, Fomin and Hoie largely improved this result to $O^*(1.2010^n)$ [FOM 06].

Our purpose is to devise an exact (optimal) algorithm for MIN 3-DOMINATING SET. We propose a search tree-based algorithm with running time $O^*(1.344^n)$. This result, although dominated by that of Fomin and Hoie, improves the complexity of search-tree based algorithms for this problem (the algorithm of Fomin is based on a graph-theoretic result). Moreover, it is to our knowledge the best known bound achieved by a polynomial space algorithm.

This result is achieved thanks to a counting procedure, based upon the introduction of weights, which allows us to measure in a more precise way the progress made by the algorithm. Such an approach has already been used in [BEI 05] for MIN 3-COLORING,

in [DEL 07] for MIN 3-SET COVER and in [FOM 05] for domination and covering problems.

8.3.1. *Analysis*

Our algorithm is a search-tree based algorithm. More precisely, at each step, we will consider 5 different types of vertices (types 1, 2 and 3 correspond to vertices the status (in/out of the dominating set) of which has already been decided):

1) Already dominating vertices

2) Already dominated but non-dominating vertices

3) Non-dominating vertices not yet dominated

4) Vertices non-fixed yet, but already dominated (by a type 1 vertex)

5) Vertices non-fixed yet, but not already dominated.

At each step, the current graph is the graph induced by vertices of type 3, 4 and 5. Indeed, vertices of type 1 and 2 have already been settled. For vertices of type 3, even if we have decided that we do not take them, they are not yet dominated, so we have to keep them in the current graph.

Now, let us consider the following easy cases, where we can reduce the graph without branching:

1) If a vertex i has degree 1, we can delete it from the graph and take its neighbor j if j has type 4 or 5, and we have to take i if j has type 3.

2) An edge between two type 4 vertices can be deleted.

3) Consider Figure 8.20, where vertex 1 has type 4 or 5, vertices 2 and 3 have any degree, and at least one of these two vertices does not have type 3. Then we can assume that we do not take vertex 1. Indeed, instead we can take vertex 2 (or vertex 3).

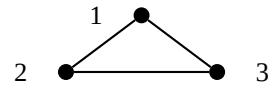


Figure 8.20. *Reducing the graph without branching: case 3*

Then at each step our algorithm either reduces the graph according to one of the three previous cases, or applies a branching on a vertex v, following the different cases given below.

For the analysis of this branching, we will distinguish the 3 following cases:

– Case 1 (section 8.3.3): the graph contains only vertices of type 5;

– Case 2 (section 8.3.4): the graph contains only vertices of type 4 and 5;

– Case 3 (section 8.3.5): the graph contains vertices of type 3 (and 4 and 5).

Our analysis is based on the following properties:

– Type 3 vertices have degree 2 at most (hence exactly 2). This is recursively verified;

– Type 4 vertices have degree 2 at most (hence exactly 2).

Dominance condition. In cases 1 and 2, if we branch on a vertex of degree 2 and if this vertex is taken in the solution, we can assume that the children and the grandchildren of this vertex are not taken in the solution. Indeed assume that we branch on a vertex i with two children j and k. If we take j or a child of j, then we can take k instead of i.

8.3.2. *Counting*

A usual way to measure the complexity of the algorithm is to count, at each step, the number of vertices (or the number of edges) of the current graph. The number of vertices (or edges) settled at each step leads to a recursive expression of the complexity that gives the final bound. This is the way the complexity is analyzed in section 8.2 for instance.

Here, we propose a more careful analysis of the complexity. The counting analysis will be done according to the edges in the current graphs. More precisely, we will consider that:

– edges that disappear from the current graph are completely settled;

– edges that remain in the current graph but that are adjacent to a type 3 vertex are partially settled. We will say that they are settled with a proportion $\alpha \in]0, 1[$ (the analysis will lead to fix $\alpha = 0.456$). As a consequence, when such an edge disappears, the gain associated will not be 1 but $1 - \alpha$.

In other words, the current graph is a graph where edges are weighted: a weight $1 - \alpha$ for edges adjacent to a type 3 vertex, a weight 1 for the other edges. This weight corresponds the amount of missing knowledge for this edge.

8.3.3. *Case n°1*

Let us begin with the simplest case, i.e., the case where the graph contains only type 5 vertices. In this case, all edges have weight 1.

Let us note that there is always a vertex of degree 2 (even possibly at the first step of the algorithm). On the other hand, if all vertices have degree 2, then we can solve the problem in polynomial time. Hence, we branch on a vertex of degree 2 adjacent to (at least) one vertex of degree 3.

More precisely, we will consider 5 different configurations: configurations 1 and 2 when the branching vertex is adjacent to a vertex of degree 2 and a vertex of degree 3, configurations 3, 4 and 5 when the branching vertex is adjacent to two vertices of degree 3.

REMARK 8.3.– When we branch on a vertex i, if a grandchild of i is adjacent only to children or grandchildren of i, the solution given by the dominance condition is not feasible. In this case, we can assume that we do not take vertex i. In this case, $T(m) \leqslant T(m-2\alpha) + O(p(m))$ (for some polynomial p).

• Configuration n°1 (Figure 8.21):

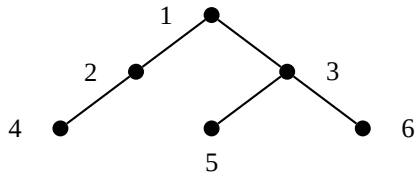


Figure 8.21. *Configuration 1 of case 1*

Thanks to Remark 8.3, we can assume that at least 3 edges (different from those depicted in the figure) are adjacent to vertices 4, 5 and 6.

If we take vertex 1, using the dominance condition, we do not take vertices from 2 to 5 and we settle $5+3\alpha$ edges: edges $(1,2),(1,3),(2,4),(3,5),(3,6)$ counted 1 and the 3 non-depicted edges in the figure, counted α since vertices 4, 5 and 6 are not yet dominated (they will be type 3 vertices).

If we do not take either vertex 1 or vertex 2, we have to take vertices 3 and 4. We settle 6 edges: the 5 edges depicted in the figure (counted 1) and the non-depicted edge adjacent to 4, which is counted 1 also.

If we do not take vertex 1 but vertex 2, we can assume that we do not take either vertex 3 or vertex 4. In this case, we settle $4+2\alpha$ edges: edges $(1,2)$, $(1,3)$ and $(2,4)$ counted 1, edges $(3,5)$ and $(3,6)$ counted α (since vertex 3 is not yet dominated) and the non-depicted edge adjacent to 4 counted 1.

Then, $T(m) \leqslant T(m-4-2\alpha) + T(m-6) + T(m-5-3\alpha) + O(p(m))$.

• Configuration n°2 (Figure 8.22):

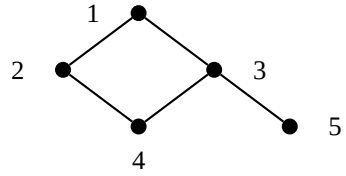


Figure 8.22. *Configuration 2 of case 1*

Thanks to Remark 8.3, we can assume that at least 2 edges (different from those depicted in the figure) are adjacent to vertices 4 and 5.

If we take vertex 1, using the dominance condition, we do not take vertices from 2 to 5 and we settle $5 + 2\alpha$ edges.

If we take neither vertex 1 nor vertex 2, we have to take vertices 3 and 4. We settle 6 edges.

If we do not take vertex 1 but vertex 2, we can assume that we take neither vertex 3 nor vertex 4. In this case, we have to take vertex 5 and we settle 7 edges.

Then, $T(m) \leqslant T(m - 5 - 2\alpha) + T(m - 6) + T(m - 7) + O(p(m))$.

• Configuration n°3 (Figure 8.23):

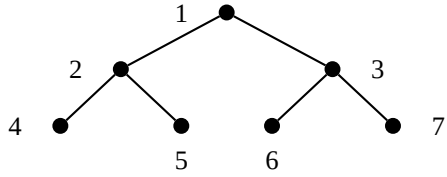


Figure 8.23. *Configuration 3 of case 1*

If we take vertex 1: using the dominance condition, none of the vertices from 2 to 7 are taken in the solution. We can assume that at least one edge links each of the vertices 4, 5, 6 and 7 to the remainder of the graph. We settle $6 + 4\alpha$ edges.

Otherwise, if we take vertex 2, we settle 4 edges.

Finally, if we take neither vertex 1 nor vertex 2, we have to take vertex 3, and we settle $4 + 2\alpha$ edges.

Then, $T(m) \leqslant T(m-4) + T(m-4-2\alpha) + T(m-6-4\alpha) + O(p(m))$.

Let us note that in the case where we settle 4 edges, the following branching will be done on a dominated vertex.

• Configuration n°4 (Figure 8.24):

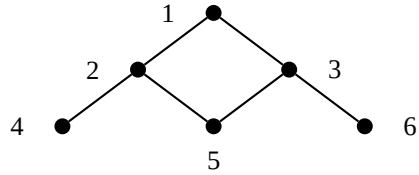


Figure 8.24. *Configuration 4 of case 1*

Thanks to Remark 8.3, we can consider that at least 3 edges link vertices 4, 5 and 6 to the remainder of the graph (one for each vertex).

If we take vertex 1: using the dominance condition, we can assume that we do not take vertices from 2 to 6. Then we have to take the last neighbor of 5 and we settle $7 + 2\alpha$ edges.

If we do not take vertex 1, we can do the same analysis as in the previous case.

Then, $T(m) \leqslant T(m-4) + T(m-4-2\alpha) + T(m-7-2\alpha) + O(p(m))$.

Let us note that in the case where we settle 4 edges, the following branching will be done on a dominated vertex.

• Configuration n°5 (Figure 8.25):

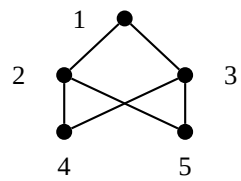


Figure 8.25. *Configuration 5 of case 1*

We can assume that we never take vertex 2. Indeed, either vertex 1 is in the solution and it is the dominance condition, or vertex 3 is in the solution and we can take vertex

4 or 5 instead of vertex 2, or one of the vertices 4 and 5 is in the solution (to dominate 3 which is non-dominated) and in this case we can take both vertices (4 and 5).

Then, $T(m) \leqslant T(m - 2\alpha) + O(p(m))$.

8.3.4. *Case n°2*

The branching is done on a type 4 vertex (of degree 2). Configuration 1 corresponds to the case where at least one neighbor of the branching vertex has degree 2. Configurations 2, 3 and 4 correspond to the case where both neighbors of the branching vertex have degree 3.

• Configuration n°1 (Figure 8.26):

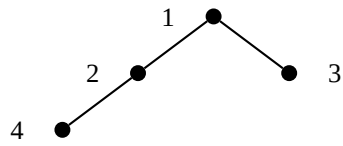


Figure 8.26. *Configuration 1 of case 2*

Vertex 3 has any degree. In this case, we can assume that we do not take vertex 2 (instead, we can take vertex 4).

Then, $T(m) \leqslant T(m - 2\alpha) + O(p(m))$ (since vertex 2 is not dominated).

• Configuration n°2 (Figure 8.27):

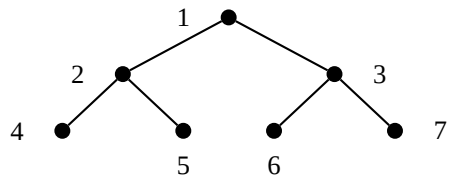


Figure 8.27. *Configuration 2 of case 2*

Assume first that at least 4 edges which are not depicted in the figure are adjacent to vertices 4, 5, 6 and 7.

– If we take vertex 1: using the dominance condition, none of the vertices from 2 to 7 is in the solution. We settle $6 + 4\alpha$ edges.

– If we do not take vertex 1: we settle 2 edges.

In all, $T(m) \leqslant T(m-2) + T(m-6-4\alpha) + O(p(m))$.

If at most 3 edges link vertices 4, 5, 6 and 7 to the remainder of the graph, then:

– either a vertex, for example 4, is linked only to his brother, 5 in this case. This is a preprocessed case of Figure 8.20;

– or two cousin vertices, for example 5 and 6, are not linked to vertices which are not depicted in Figure 8.27. In this case, at least one of the two vertices, for example 5, is not dominated (otherwise edge $(5,6)$ would have been deleted). In this case, the solution given by the dominance condition when we take vertex 1 is not feasible, hence we can assume that we do not take it. We settle 2 edges.

• Configuration n°3 (Figure 8.28):

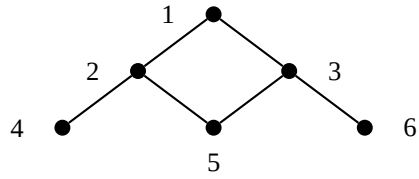


Figure 8.28. *Configuration 3 of case 2*

We can assume that we do not take vertex 1. Instead, we can take vertex 5.

Then, $T(m) \leqslant T(m-2) + O(p(m))$.

• Configuration n°4 (Figure 8.29):

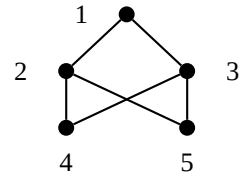


Figure 8.29. *Configuration 4 of case 2*

As in the case of a branching of a non-dominated vertex, we can assume that we do not take vertex 2.

Then, $T(m) \leqslant T(m-3\alpha) + O(p(m))$.

8.3.5. *Case n°3*

Now, we consider the most delicate case, where the graph contains type 3 vertices. Configuration n°1 corresponds to the case where there is a triangle (not considered in the preprocessed cases). Configurations 2 and 3 correspond to the (remaining) cases where there are two type 3 vertices at distance 2 from each other (let us recall that we can assume that two type 3 vertices are not adjacent). In configurations 4, 5, 6 and 7, we assume that between any two type 3 vertices the distance is at least 3.

Let us note that if a vertex is linked to three type 3 vertices, then it has type 5 and we have to take it (we settle (more than) $3 - 3\alpha$ edges). Finally, recall that all vertices of degree 3 have type 5.

On the figures, white vertices are type 3 vertices.

REMARK 8.4.– Assume that we are in the configuration of Figure 8.30. When we determine the status of vertex 1, either we take it, or we do not and we have to take vertex 3. In both cases, we settle the two edges of the graph, i.e., $2 - 2\alpha$. For $\alpha \leqslant 1/2$, $2 - 2\alpha \geqslant 1 \geqslant \alpha$.

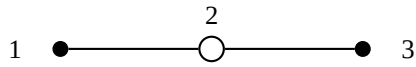


Figure 8.30. *Case of Remark 8.4*

• Configuration n°1 (Figure 8.31):

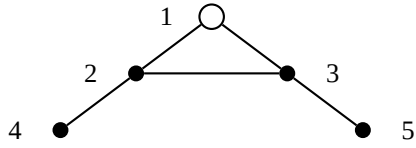


Figure 8.31. *Configuration 1 of case 3*

Vertices 2 and 3 must have type 5.

Suppose that we take vertex 2. Then, we can assume that we do not take vertex 3; therefore, we must take vertex 5. We settle $5 - 2\alpha$ edges. The other case is symmetric.

Then, $T(m) \leqslant 2T(m - 5 + 2\alpha) + O(p(m))$.

Let us note that this is valid also if vertex 4 and/or vertex 5 has (have) type 3. Indeed, in this case, thanks to Remark 8.4, when we settle vertex 2 and/or 3, we settle the $2 - 2\alpha \geqslant 1$ edges adjacent to vertex 4 and/or 5.

Finally, note that the case where vertices 4 and 5 are the same vertex is trivial (take one vertex among 2 and 3 (the choice does not matter), there is no need to branch).

• Configuration n°2 (Figure 8.32):

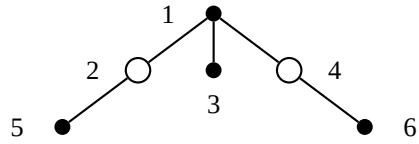


Figure 8.32. *Configuration 2 of case 3*

In this case, vertex 1 has type 5.

Either we take vertex 1 and we settle $5 - 4\alpha$ edges, or we do not take vertex 1 and we have to take vertices 3, 5 and 6. Since vertices 3, 5 and 6 have degree of at most 2, we can assume that we settle the edge adjacent to vertex 3 for instance (then, this edge is definitely settled and counts 1) (even if it is adjacent to a type 3 vertex, thanks to Remark 8.4). We settle $6 - 4\alpha$ edges (we can do even better).

Then, $T(m) \leqslant T(m - 5 + 4\alpha) + T(m - 6 + 4\alpha) + O(p(m))$.

• Configuration n°3 (Figure 8.33):

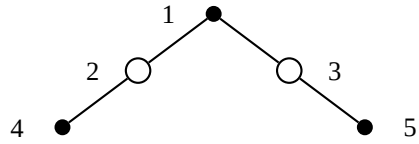


Figure 8.33. *Configuration 3 of case 3*

First note the following. If vertices 4 and 5 are not adjacent to any vertex which is not depicted in the figure (only the edge $(4, 5)$ is missing), then the graph of Figure 8.33 is isolated from the remainder of the graph. In this case, there is no need to branch, we just have to solve optimally the problem on this subgraph. The situation is analog when 4 and 5 are both linked to the same type 3 vertex.

Now, we consider that we are not in one of the two previous situations.

If vertex 1 has type 5, we have to take it. Otherwise, vertex 1 has type 4. In this case we do not take it and we have to take vertices 4 and 5. In this case, we settle $6 - 4\alpha$ edges. Indeed, thanks to Remark 8.4, we can consider that taking vertices 4 and 5 makes it possible to settle edges of total weight of at least 2, even if they are adjacent to type 3 vertices (we have already considered the two cases where the total weight would have been smaller than 2).

On the other hand, if we take vertex 1, we can assume that we do not take either 4 or 5. In this case, we settle $4 - 4\alpha + 2\alpha = 4 - 2\alpha$ edges. Indeed, settling vertices 4 and 5 makes it possible to settle 2α edges which are not depicted in the graph.

Then, $T(m) \leqslant T(m - 4 + 2\alpha) + T(m - 6 + 4\alpha) + O(p(m))$.

REMARK 8.5.– As we said previously, from now we can assume that the distance between any two type 3 vertices is at least 3.

• Configuration n°4 (Figure 8.34):

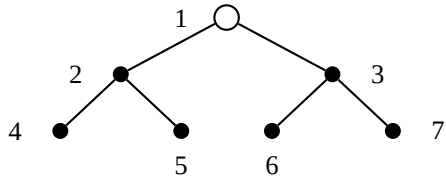


Figure 8.34. *Configuration 4 of case 3*

Vertices 4, 5, 6 and/or 7 may be the same vertex.

If we take vertex 2, then we settle $4 - 2\alpha$ edges, otherwise, we have to take vertex 3 and we settle $(2 - 2\alpha) + 2\alpha + 2 = 4$ edges.

In all, $T(m) \leqslant T(m - 4 + 2\alpha) + T(m - 4) + O(p(m))$.

• Configuration n°5 (Figure 8.35):

If we take vertex 2, then we can assume that we do not take either vertex 3 or vertex 4. Then we settle at least $(2 - 2\alpha) + 1 + 2\alpha + \alpha = 3 + \alpha$ edges.

Configuration 5.1: vertex 2 has type 5. If we do not take it, we have to take vertices 3 and 4. We settle (at least) $2 - 2\alpha + 3 = 5 - 2\alpha$ edges.

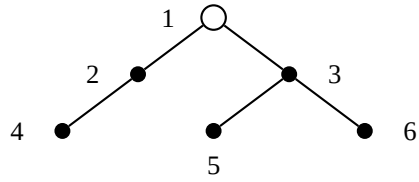


Figure 8.35. *Configuration 5 of case 3*

Configuration 5.2: vertex 2 has type 4. If we do not take it, we have to take vertex 3. We settle $2 - 2\alpha + 3 = 5 - 2\alpha$ edges.

In both configurations: $T(m) \leqslant T(m - 3 - \alpha) + T(m - 5 + 2\alpha) + O(p(m))$.

• Configuration n°6 (Figure 8.36):

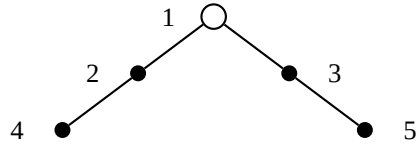


Figure 8.36. *Configuration 6 of case 3*

Configuration 6.1: vertices 2 and 3 have type 5. If we take vertex 2, we can assume that we do not take either 3 or 4, and we have to take 5. We settle 4 edges. Otherwise, we have to take vertices 3 and 4. We settle $4 - \alpha$ edges.

Configuration 6.2: vertices 2 and 3 have type 4. If we take vertex 2, we can assume that we do not take either 3 or 4. We settle $4 - 2\alpha + 2\alpha = 4$ edges. If we do not take vertex 2, we have to take vertex 3. We settle $4 - 2\alpha$ edges.

Configuration 6.3: vertex 2 has type 4, vertex 3 has type 5. If we take vertex 2, we can assume that we do not take either 3 or 4, and we have to take 5. We settle 4 edges. If we do not take vertex 2, we have to take vertex 3. We settle $4 - 2\alpha$ edges.

In the worst case, we have: $T(m) \leqslant T(m - 4 - 2\alpha) + T(m - 4) + O(p(m))$.

8.3.6. *Complexity analysis*

The complexity analysis shows that the bigger α is, the better cases 1 and 2 are, but the worse case 3 is. An optimal value is $\alpha = 0.456$.

For this value, we get the following:

– in case n°1, the configurations give:
1) $T(m) \leqslant T(m-4.912)+T(m-6)+T(m-6.368)+O(p(m))$;
2) always better than configuration 1 for $\alpha \leqslant 1/2$;
3) $T(m) \leqslant T(m-4)+T(m-4.912)+T(m-7.824)+O(p(m))$ but we can improve it, thanks to the fact that in one case the next branching is on a dominated vertex;
4) always better than 3 for $\alpha \leqslant 1/2$;
5) no branching;

– in case n°2, the configurations give:
1) $T(m) \leqslant T(m-0.912)+O(p(m))$;
2) $T(m) \leqslant T(m-2)+T(m-7.824)+O(p(m))$;
3) always better than 2 for $\alpha \leqslant 1/2$;
4) always better than 1;

The worst configuration (configuration 2) gives[1] $T(m) = O^*(1.178^m) \leqslant O^*(1.279^n)$

– in case n°3, the configurations give:
1) always better than 4 for $\alpha \leqslant 1/2$;
2) always better than 4 for $\alpha \leqslant 1/2$;
3) always better than 4 for $\alpha \leqslant 1/2$;
4) $T(m) \leqslant T(m-3.088)+T(m-4)+O(p(m))$;
5) $T(m) \leqslant T(m-3.456)+T(m-4.088)+O(p(m))$;
6) same as 4;

The worst configuration (configuration 4) gives $T(m) = O^*(1.2174^m) \leqslant O^*(1.344^n)$.

Let us now conclude this analysis in case 1.

For configuration 3, when we are in the case $m-4$, we branch on a dominated vertex (case 2). In this case, either we settle 2α (preprocessing case or configuration 1), or we are in case 2. Thus, we replace $T(m-4)$ by $T(m-4-2\alpha)$ in the first case and by $T(m-6)+T(m-10-4\alpha)$ in the second case. The worst complexity is obtained in the second case: $T(m) \leqslant T(m-4-2\alpha)+T(m-6)+T(m-6-4\alpha)+T(m-10-4\alpha)+O(p(m))$, which leads to a complexity of $T(m) = O^*(1.2174^m) \leqslant O^*(1.344^n)$.

Thus, we obtain a worst case complexity of $O^*(1.344^n)$.

1. Note that since the input graph has degree 3, $m \leqslant 3n/2$.

8.4. Bibliography

[BEI 05] BEIGEL R., EPPSTEIN D., "3-coloring in time $O(1.3289^n)$", *J. Algorithms*, vol. 54, num. 2, p. 168–204, 2005.

[BOD 00] BODLAENDER H. L., JANSEN K., "On the complexity of the maximum cut problem", *Nordic Journal of Computing*, vol. 7, num. 1, p. 14–31, 2000.

[DEL 07] DELLA CROCE F., ESCOFFIER B., PASCHOS V. T., "Improved worst-case complexity for the MIN 3-SET COVERING problem", *Oper. Res. Lett.*, vol. 35, num. 2, p. 205–210, 2007.

[EPP 01] EPPSTEIN D., "Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction", *Proc. Symposium on Discrete Algorithms, SODA'01*, p. 329–337, 2001.

[FED 06] FEDIN S. S., KULIKOV A. S., "A $2^{|E|/4}$-time algorithm for MAX-CUT", *Journal of Mathematical Sciences*, vol. 134, num. 5, p. 2383–2391, 2006.

[FOM 04] FOMIN F. V., KRATSCH D., WOEGINGER G. J., "Exact (Exponential) algorithms for the dominating set problem", HROMKOVIC J., NAGL M., WESTFECHTEL B., Eds., *Proc. International Workshop on Graph Theoretical Concepts in Computer Science, WG'04*, vol. 3353 of *Lecture Notes in Computer Science*, Springer Verlag, p. 245–256, 2004.

[FOM 05] FOMIN F., GRANDONI F., KRATSCH D., "Measure and conquer: domination – a case study", CAIRES L., ITALIANO G. F., MONTEIRO L., PALAMIDESSI C., YUNG M., Eds., *Proc. ICALP'05*, vol. 3580 of *Lecture Notes in Computer Science*, Springer Verlag, p. 191–203, 2005.

[FOM 06] FOMIN F. V., HØIE K., "Pathwidth of cubic graphs and exact algorithms", *Inform. Process. Lett.*, vol. 97, num. 5, p. 191–196, 2006.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and intractability. A guide to the theory of **NP**-completeness*, W. H. Freeman, San Francisco, 1979.

[GRA 03] GRAMM J., HIRSCH E., NIEDERMAIER R., ROSSMANITH P., "Worst-case upper bounds for Max2Sat with an application to MaxCut", *Discrete Appl. Math.*, vol. 130, p. 139–155, 2003.

[KNE 05a] KNEIS J., MÖLLE D., RICHTER S., ROSSMANITH P., "Algorithms based on the treewidth of graphs", KRATSCH D., Ed., *Proc. International Workshop on Graph Theoretical Concepts in Computer Science, WG'05*, vol. 3787 of *Lecture Notes in Computer Science*, Springer Verlag, p. 385–396, 2005.

[KNE 05b] KNEIS J., ROSSMANITH P., A new satisfiablity algorithm with applications to MAX-CUT, Technical Report num. AIB-2005-08, Department of Computer Science, RWTH Aachen, 2005.

[LOV 75] LOVÁSZ L., "Three short proofs in graph theory", *J. Combin. Theory Ser. B*, vol. 19, p. 269–271, 1975.

[POL 95] POLJAK S., TUZA Z., "Maximum cuts and large bipartite subgraphs", COOK W., LOVÁSZ L., SEYMOUR P., Eds., *Combinatorial Optimization. Papers from the DIMACS Special Year*, p. 181–224, AMS, 1995.

[SAK 03] SAKAI S., TOGASAKI M., YAMAZAKI K., "A note on greedy algorithms for the maximum weighted indepedent set problem", *Discrete Appl. Math.*, vol. 126, p. 313–322, 2003.

[SCH 86] SCHRIJVER A., *Theory of Linear and Integer Programming*, John Wiley & Sons, New York, 1986.

[SCO 03] SCOTT A. D., SORKIN G. B., "Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances", *Proc. RANDOM'03*, vol. 2764 of *Lecture Notes in Computer Science*, Spinger-Verlag, p. 382–395, 2003.

[SCO 04] SCOTT A. D., SORKIN G. B., Solving sparse semi-random instances of Max-Cut and Max-CSP in linear expected time, Research Report num. 23417 (W0411-056), IBM Research division, Thomas J. Watson Research Center, 2004.

[WIL 04] WILLIAMS R., A new algorithm for optimal constraint satisfaction and its implications, Report num. 32, Electronic Colloquium on Computational Complexity, 2004.

[WŒG 03] WŒGINGER G. J., "Exact algorithms for **NP**-hard problems: a survey", JUENGER M., REINELT G., RINALDI G., Eds., *Combinatorial Optimization - Eureka! You shrink!*, vol. 2570 of *Lecture Notes in Computer Science*, p. 185–207, Springer Verlag, 2003.

[WŒG 04] WŒGINGER G. J., "Space and time complexity of exact algorithms: some open problems", *Proc. IWPEC'04*, vol. 3162 of *Lecture Notes in Computer Science*, Springer Verlag, p. 281–290, 2004.

[YAN 78] YANNAKAKIS M., "Node- and edge-deletion **NP**-complete problems", *Proc. STOC'78*, p. 253–264, 1978.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 9

The Online Track Assignment Problem

9.1. Introduction

This chapter aims to start an analytical study of the computational complexity of some online shunting problems. We start our analysis with the following problem. We consider a depot consisting of a set of parallel tracks. Each track can be approached from one side only and the number of trains per track is limited. The departure times of the trains are fixed according to a given timetable. The problem is to assign a track to each train as soon as it arrives at the depot and such that it can leave the depot on time without being blocked by any other train. We show how to solve this problem as an online bounded coloring problem on special graph classes. For this problem we study the competitiveness of the First Fit algorithm and we show it matches the competitive ratio of the problem.

In a train depot, trains must be stored during the night on tracks. Each track has a finite capacity and may contain only a bounded number of trains. Throughout this chapter, we consider that this bound is fixed to an integer m. The tracks are organized as stacks, such that the last train to enter must be the first to leave on the next morning. In order to save upkeep costs, we want to use as few tracks as possible. Meanwhile, in order to save time and energy, we want to make sure that, when a train departs the next morning, it is always at the top of the stack. Since the trains have accumulated delays during the day, the time of arrival of each train is unpredictable. The tracks must thus be assigned online, as the trains arrive, on the basis of departure times and previous assignments.

Chapter written by Marc DEMANGE, Gabriele DI STEFANO and Benjamin LEROY-BEAULIEU.

We can represent on a time-axis the intervals during which each train must be stored in the depot. Two trains may be on the same track if their intervals are completely disjoint or if the interval of one train is contained in the interval of the other train (the first train arrives before and leaves after the second). We can make a graph where each vertex represents an interval and two vertices are joined by an edge if and only if the two intervals overlap but have no containment relationship. Such a graph is called an *overlap graph* [GOL 04]. To model our track assignment problem, we can thus use online coloring of overlap graphs with the constraint that each color may not contain more than m elements. However, as we only know the time of arrival when the train arrives at the station, the online model must state that the vertices of the overlap graph are presented in the order of the left extremity of their corresponding intervals along the time axis.

It is natural to consider that there is a time in the night at which all trains are in the depot. This is called the *midnight condition* and it means that the intervals of the trains all share at least one point. It has been proved [GAV 73] that in this particular case, the overlap graph is a *permutation graph*. It is thus natural to study the problem on online bounded coloring of permutation graphs. Here, since we know the time of arrival and of departure of our train, we consider that the permutation graph is given on a lattice plan, which is equivalent to consider that we receive the permutation online and which is more informative than receiving just the corresponding graph online [DEM 07].

9.1.1. *Related works*

The track assignment problem is closely related to the more general shunting problem, that concerns the rolling stock allocation on a railway infrastructure under time, space and operational constraints. The shunting problem occurs in practical optimization problems like, e.g., the storage of trams or buses in a depot outside the rush hours, the rearrangement of railroad cars among different trains, the freight car distribution, and also the assignment of trains to platforms in a station [BLA 99, DAH 00, DIS 04, FRE 05, GAL 01, HAM 05a, HAM 05b, HE 00, ROS 03, WIN 00, ZWA 96].

However, appropriate mathematical models which take into account all the operation restrictions, capacity limits, and also time and operating constraints that arise in this context are quite complex. In [COR 07] the authors, motivated by the purpose of analyzing the abstract background of the track-assignment problem, studied the following problem: a set of trains and a set of parallel tracks are given. Each train might approach the depot from the left-hand side or from the right-hand side and might leave the depot to the left-hand side or to the right-hand side. The direction from which it enters or from which it leaves the depot, however, is fixed. Also the arrival time and departure time are fixed. The problem is to assign tracks to the trains such that they can enter and leave the station on time without being blocked by any other train. The main difference with our model is that capacity constraints are not considered: the tracks

have infinite length. The authors consider some variations of the offline problem on linear timetables as well as on cyclic timetables. They also analyze the online version of one variant of the assignment problem: when a linear timetable and the midnight condition are assumed.

9.1.2. *Results*

We carefully analyze the competitive ratio of the track-assignment problem when the number of tracks k and their capacity m are known.

We prove that a lower bound for the competitive ratio of the track-assignment problem is $2 - 1/l$, where $l = \min\{m, k\}$.

Under the midnight condition, we show an online algorithm which is optimal. Thus, in this case, the competitive ratio of the track-assignment problem is $2 - 1/l$.

All the results are achieved by transforming the track-assignment problem into a bounded coloring problem on permutations.

9.2. Definitions and notations

Any term related to graph theory not defined here can be found in [BER 76].

A *graph* G consists of a finite set V and an irreflexive binary relation on V. V is called the set of *vertices*. The binary relation is represented by a collection E of unordered pairs, which are called *edges*. We denote $G = (V, E)$. If $(v, w) \in E$, then v and w are said to be *adjacent*.

A *stable set* of a given graph $G = (V, E)$ is a set $S \subseteq V$ such that for each pair of vertices v and w in S, $(v, w) \notin E$. A *clique* is a set $K \subseteq V$ such that for each pair of vertices v and w in S, $(v, w) \in E$.

A *proper coloring* of a graph, which is sometimes simply called a *coloring*, consists of a partition of V into stable sets. Each stable set is associated with a color. When talking about graph coloring, the terms "stable set" and "color" are often used as equivalents. A k-coloring is a coloring that uses at most k stable sets. The *chromatic number* $\chi(G)$ of a graph G is the smallest number k such that G admits a k-coloring. A k-mb-coloring is a coloring that uses at most k stable sets and where each set has a size smaller than m. The *bounded chromatic number* $\chi_m(G)$ of a graph G is the smallest number k such that G admits a k-mb-coloring.

In this chapter, we consider online versions of the problems of coloring graphs. An online problem can be seen as a two player game involving an adversary and an algorithm. The adversary presents the instance and the algorithm gives the solution. The

online problem is generally characterized by the underlying offline problem and two sets of rules that have to be respected by the adversary and the algorithm, respectively.

Online algorithms are traditionally evaluated according to their *competitive ratio*. Let A be an online algorithm and P a minimization problem. Then $p_A(I)$ denotes the maximum score that A achieves for P over the online presentation of the instance I respecting the given rules. An online algorithm is said to guarantee a competitive ratio of c (or to be c-competitive) if, for every instance I, $p_A(I) \leqslant c \cdot p(I)$, where $p(I)$ is the offline optimal solution for P on I. An online algorithm is called *exact* (or solves the problem exactly) if it calculates the optimal offline solution for any online instance (it has a competitive ratio of 1). It is called *optimal* if its competitive ratio cannot be improved by any other online algorithm. The *competitive ratio of a problem* is the competitive ratio of an optimal algorithm for the problem.

A *permutation graph* is an undirected graph for which there exists a permutation such that every vertex of the graph corresponds to an element in the permutation and two vertices are adjacent if and only if the corresponding elements appear in reverse order in the permutation.

In this chapter, we will use a very common representation of permutation graphs, called the *lattice representation*. The permutation is represented on a two-dimensional plan: the y-axis represents the values of the elements of the permutation and the x-axis represents the position of these elements. A point (x, y) on the plan means that an element of value y is at position x in the permutation. For example, the permutation $[3, 4, 1, 2]$ can be represented by the set of points $\{(1, 3), (2, 4), (3, 1), (4, 2)\}$. We consider sets S of points that can have values in $\mathbb{R}^2$ and such that, if (x, y) and (x', y') are two points of S, then $x \neq x'$ and $y \neq y'$. For these kinds of sets it is straightforward to find the associated permutation: it is sufficient to number the points following the decreasing order of the y values and to present the numbers following the increasing order of the x values.

Given a set of points in the plan as above, the corresponding permutation graph is such that the points are the vertices and two points (x, y) and (x', y') are adjacent if $x < x'$ and $y > y'$. Coloring a permutation graph is equivalent to partitioning a permutation into increasing subsequences [DIS 06]. For example, $[3, 6, 5, 10, 7, 1, 11, 8, 2, 12, 9, 4]$ can be decomposed into $[3, 6, 10, 11, 12]$, $[5, 7, 8, 9]$ and $[1, 2, 4]$. The equivalence with a partition into stable sets is due to the fact that two points (x, y) and (x', y') form a stable set iff $x < x'$ and $y < y'$. Thus, a *bounded coloring of a permutation* is a decomposition of the permutation into increasing sequences, each with a limited number of elements.

In the online version of the bounded coloring of a permutation, the rules we will use are the following: the adversary has to present an element of the permutation as a point in the plan and the algorithm has to permanently assign it to a stable set.

The adversary presents the elements following the order in which they appear in the permutation: from the left to the right.

A very common algorithm for coloring graphs is the greedy algorithm *First-Fit*, denoted by FF. It considers the vertices one after the other following a given order. It assigns to vertex v the first color that was not assigned to an adjacent vertex of v. Chvátal [CHV 84] characterized the *perfect orderings* that are such that FF solves the graph coloring problem on the graph and all its induced subgraphs optimally. This is, for example, true, if the vertices are ordered according to a transitive orientation of the arcs in the graph and, for permutation graphs, when the vertices are presented following the order in the permutation from the left to the right. Then FF, used as an online algorithm, is an exact algorithm for the coloring of a permutation. Note, however, that for arbitrary vertex-orderings the FF algorithm can behave arbitrarily bad even on permutation graphs [NIK 00].

In this chapter we also consider the m-bounded First-Fit algorithm (m-FF): it colors a new vertex with the smallest possible color used less than m times.

9.3. Bounds for the permutation graph in a left-to-right model

We start by studying the particular case which respects the midnight condition. In order to stick to our practical problem, we consider that the permutation is delivered from left to right on a lattice representation.

Let us suppose that the permutation is delivered from left to right.

LEMMA 9.1.– *A lower bound for the online bounded coloring of a permutation is* $2 - 1/l$, *where* $l = \min\{m, k\}$.

Proof. Suppose that k is known and fixed *a priori*. We prove Lemma 9.1 with a permutation presented on its lattice representation. The zone A denotes the admissible zone where any element in the future will be presented. At the beginning, A is the complete plan.

1) Present a stable set S (an increasing sequence) of size m in A.

2) If the elements of S are colored with two or more different colors, define A as the part of the plan which is below and to the right of S, such that every element on A will be in a clique with every element in S. S is said to be of type α. If all the elements of S are colored with the same color, define A as the part of the plan which is above and to the right of S, such that every element on A will be in a stable set with every element in S. S is said to be of type β. Figure 9.1 represents zone A after presenting one β sequence and one α sequence.

3) Repeat steps 1 and 2 until $l - 1$ sequences S have been presented.

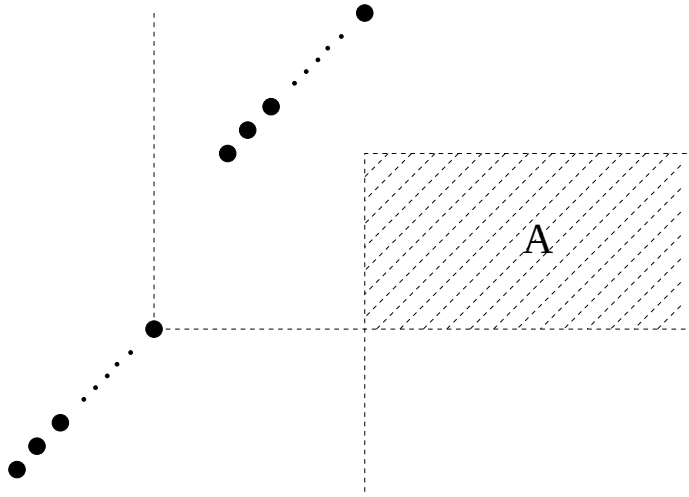


Figure 9.1. *Zone A after presenting one β increasing sequence and one α increasing sequence*

4) Let N_β be the number of sequences S of type β. Present a clique C_γ of size $N_\beta + 1$ in A.

Let λ_α be the number of colors used in all sequences of type α, λ_β be the number of colors used in all sequences of type β and λ_γ be the number of colors used on C_γ. Finally, let λ be the total number of colors used by the algorithm. Clearly:

$$\lambda = \lambda_\alpha + \lambda_\beta + \lambda_\gamma \qquad [9.1]$$

The definition of A after a sequence S of type α allows us to say that every vertex presented after S will form a clique with any element of S. Thus, any color used in S will never be used again on this instance. Besides that, since the sequences have size m, a color used on a sequence S of type β is saturated and may therefore not be used on any element which is not in S. Thus, no color is used in two different sequences. Therefore, if we call N_α the number of sequences of type α, we have:

$$\lambda_\alpha \geqslant 2N_\alpha \quad ; \quad \lambda_\beta = N_\beta \quad ; \quad \lambda_\gamma = N_\beta + 1 \qquad [9.2]$$

and thus:

$$\lambda \geqslant 2N_\alpha + 2N_\beta + 1 \qquad [9.3]$$

Since the algorithm presents exactly $l-1$ sequences of types α and β:

$$2N_\alpha + 2N_\beta + 1 = 2(N_\alpha + N_\beta) + 1 = 2(l-1) + 1 = 2l - 1 \quad [9.4]$$

$$\lambda \geqslant 2l - 1 \quad [9.5]$$

Let us now calculate the bounded chromatic number χ_m of the instance presented above. Each sequence of type α contains m elements and can thus be colored with exactly one color. Thus, it is possible to use exactly N_α colors on these sequences. Consider now the subpermutation induced by the elements of the sequences of type β and γ. By construction, we have that the sequences of type β form a long stable set S_β of size $N_\beta m$ and each element of C_γ is stable with all the elements of S_β. It is possible to cover this subpermutation with stable sets each containing at most $m-1$ elements of S_β and 1 element of C_γ. Since $|S_\beta| \leqslant (l-1)m$ and since $l \leqslant m$, at most $|C_\gamma| = N_\beta + 1$ such stable sets will be needed. Thus, this subpermutation can be colored with $N_\beta + 1$ colors. Therefore:

$$\chi_m \leqslant N_\alpha + N_\beta + 1 \leqslant l \quad [9.6]$$

From equations [9.5] and [9.6], we can deduce that the competitive ratio c of the online bounded coloring problem of permutations delivered from left to right is:

$$c \geqslant \frac{\lambda}{\chi_m} \geqslant 2 - \frac{1}{l} \quad [9.7]$$

that completes the proof. ∎

REMARK 9.1.– If k is unknown in advance, we can consider the same instance as above with $k \geqslant m$ and thus have $c \geqslant 2 - 1/m$.

Remark 9.1 and equation [9.7] end the proof of Lemma 9.1.

LEMMA 9.2.– *The performance ratio of First-Fit for the online bounded coloring of a k-mb-colorable permutation delivered from left to right is at most $2 - 1/l$, where $l = \min\{m, k\}$.*

Proof. Let $\lambda_{m\text{-FF}}(\pi)$ be the number of colors used by First-Fit for the online bounded-coloring of the permutation π presented from left to right. m is the bound imposed by the problem.

Let N_S be the number of colors which are saturated at the end of the execution, i.e., the colors which contain exactly m vertices. Let $N_{\bar{S}}$ be the number of colors which are not saturated. ∎

REMARK 9.2.– If we consider only the permutation π' induced by elements colored by colors in $N_{\bar{S}}$, we have $\lambda_{m\text{-FF}}(\pi') = \lambda_{\text{FF}}(\pi') = \chi(\pi')$, where $\lambda_{\text{FF}}(\cdot)$ denotes the number of colors used by the First-Fit algorithm for usual coloring and where the last equality holds since the permutation is presented from left to right (a perfect order).

Let χ_m be the exact bounded-chromatic number of the permutation π for the bound m:

$$\lambda_{m\text{-FF}}(\pi) = N_S + N_{\bar{S}}$$

Since colors in $N_{\bar{S}}$ are non-empty, we have:

$$n \geqslant mN_S + N_{\bar{S}} \quad [9.8]$$

$$\chi_m \geqslant \left\lceil \frac{n}{m} \right\rceil \geqslant \left\lceil \frac{mN_S + N_{\bar{S}}}{m} \right\rceil = N_S + \left\lceil \frac{N_{\bar{S}}}{m} \right\rceil \quad [9.9]$$

$$N_S \leqslant \chi_m - \left\lceil \frac{N_{\bar{S}}}{m} \right\rceil \quad [9.10]$$

We deduce from remark 9.2 that $N_{\bar{S}} \leqslant \chi_m$. Thus:

$$\lambda_{m\text{-FF}}(\pi) \leqslant 2\chi_m - \left\lceil \frac{\chi_m}{m} \right\rceil \quad [9.11]$$

We consider now two cases:

1) If $k \leqslant m$, then:

$$\lambda_{m\text{-FF}}(\pi) \leqslant 2\chi_m - 1 \quad [9.12]$$

and thus the performance ratio $c_{m\text{-FF}}$ of the First Fit algorithm is:

$$c_{m\text{-FF}} \leqslant \frac{\lambda_{m\text{-FF}}(\pi)}{\chi_m} \leqslant 2 - \frac{1}{\chi_m} \leqslant 2 - \frac{1}{k} \leqslant 2 - \frac{1}{l} \quad [9.13]$$

2) If $k > m$, then:

$$\lambda_{m\text{-FF}}(\pi) \leqslant 2\chi_m - \frac{\chi_m}{m} \quad [9.14]$$

and:

$$c_{m\text{-FF}} \leqslant \frac{\lambda_{m\text{-FF}}(\pi)}{\chi_m} \leqslant 2 - \frac{1}{m} \leqslant 2 - \frac{1}{l} \quad [9.15]$$

This ends the proof of Lemma 9.2.

THEOREM 9.1.– *The performance ratio c of the problem of online coloring a k-mb-colorable permutation is $c = 2 - 1/l$, where $l = \min\{m, k\}$, and m-First Fit achieves this ratio.*

Proof. The proof is immediate from Lemmas 9.1 and 9.2. ∎

9.4. Bounds for overlap graphs

Our first generalization is a move towards the original track-assignment problem. We analyze here the online bounded-coloring of overlap graphs.

CLAIM 9.1.– *For any algorithm, it is possible to force a stable set S of size at least three on a bipartite overlap graph such that the corresponding intervals are included in each other and there exists a limit L, situated to the left of the right-end of every interval in S, such that no interval other than those in S appear on the right of L and such that each element of the stable set has a different color.*

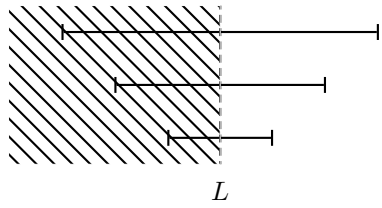


Figure 9.2. *Illustration of claim 9.1: on the left of the limit L, there can be anything. On the right side, there is nothing but the three represented intervals. All have a different color and the complete graph is bipartite*

Proof. We give here an instance which forces the result described in claim 9.1. For every interval I_i of this instance, we will call l_i the left bound of this interval and r_i the right bound of this interval.

1) (See Figure 9.3) Present an interval I_1. Then, present an interval I_2 such that $l_1 < l_2 < r_2 < r_1$. If I_2 is colored with color 2, define $I^* := I_2$; otherwise, present I_3 such that $l_2 < l_3 < r_2$ and $r_2 < r_3 < r_1$ and define $I^* := [r_2, r_3]$.

2) (See Figure 9.4) Let l^* be the left bound of I^* and r^* be the right bound of I^*. Then, present the following sequence and stop as soon as one interval is colored with color 3: present I_4 such that $l^* < l_4 < r_4 < r^*$ and let a be the color of I_4

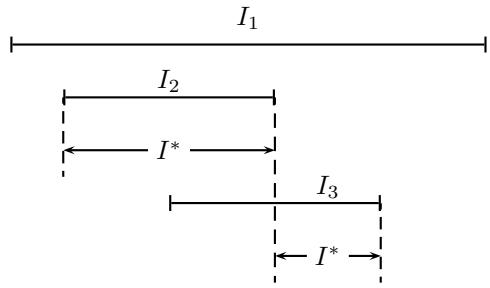


Figure 9.3. *This figure illustrates step 1. At the end of this step, in I^*, there are exactly two intervals, which form a stable set but are colored with two different colors*

($a \in \{1, 2\}$); present I_5 such that $l_4 < l_5 < r_4$ and $r_4 < r_5 < r^*$ and let b be the color of I_5 ($b \in \{1, 2\}, b \neq a$); present I_6 such that $r_4 < l_6 < r_6 < r_5$; if I_6 is colored with a, present $I_{7'}$ such that $l_6 < l_{7'} < r_6$ and $r_5 < r_{7'} < r^*$; if I_6 is colored with b, then present $I_{7''}$ such that $l_6 < l_{7''} < r_6$ and $r_6 < l_{7''} < r_5$ and I_8 such that $r_6 < l_8 < r_{7''}$ and $r_5 < r_8 < r^*$.

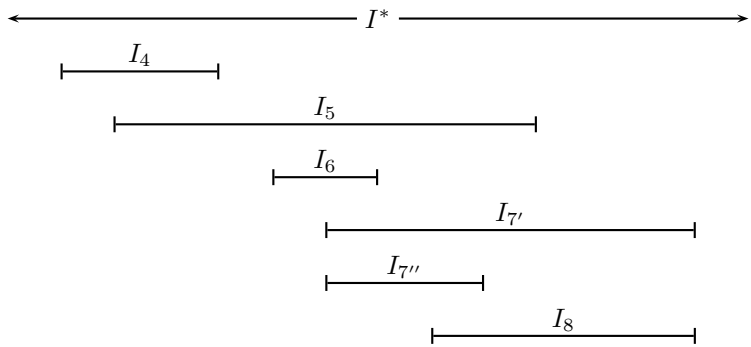


Figure 9.4. *This figure illustrates step 2. At the end of this step, in I^*, there are three intervals colored with three different colors: I_1, I_2 or I_3 and one element of $\{I_i\}_{i \geqslant 4}$*

Note that the given overlap graph is bipartite. At the end of step 1, we have a stable set of size two, colored with two different colors, and the limit L is either l_2, if I_2 was colored 2, or r_2 if I_2 was colored 1. At the end of step 2, at least one interval must have been colored with color 3. Let I_{i_3} be this interval. Since we do not present any

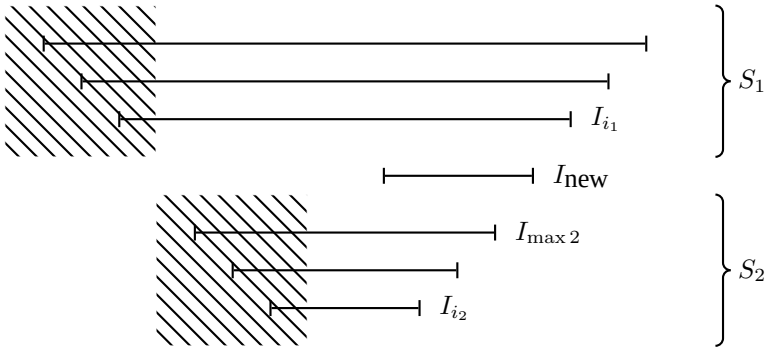


Figure 9.5. *Illustration of the proof of Theorem 9.2*

interval I_i such that $i > i_3$, we have a stable set of size at least three colored with three different colors. Note that, if $I_{7''}$ is colored 3, we have a stable set of size 4 colored with three different colors. Thus, the size of the stable set is at least 3 and at most 4. Let S be the considered stable set; we must set $L = \max\{\max_{I_i \notin S} r_i, \max_{I_i \in S} l_i\}$ to have the arrangement described in claim 9.1. ∎

THEOREM 9.2.– *For any online coloring algorithm, it is possible to force any positive number of colors on a bipartite overlap graph.*

Proof. We prove this by induction. Suppose that it is possible to force k colors on a stable set S_1 as in claim 9.1. We construct such an instance. Let I_{i_1} be the interval of S_1 such that $r_{i_1} = \min_{I_j \in S_1}\{r_j\}$ and let L_1 be the limit of S_1 as in claim 9.1. In the interval $[L_1, r_{i_1}]$, build a second stable set S_2 of the same size as S_1 and with the same number of colors on it. Let I_{i_2} be the interval of S_2 such that $r_{i_2} = \min_{I_j \in S_2}\{r_j\}$ and let L_2 be the limit of S_2. Also, let $I_{\max 2}$ be the interval of S_2 such that $r_{\max 2} = \max_{I_j \in S_2}\{r_j\}$.

If the colors used to color S_2 are different from those used to color S_1, define $S = S_1 \cup S_2$ and $L = L_2$ and S is now a stable set of the form of claim 9.1 and colored with at least $k+1$ colors. If all the colors used to color S_2 have been already used to color S_1, we present a new interval I_{new} such that $L_2 < l_{\text{new}} < r_{i_2}$ and $r_{\max 2} < r_{\text{new}} < r_{i_1}$ and define $S = S_1 \cup \{I_{\text{new}}\}$ and $L = r_{\max 2}$. S is a stable set of the form of claim 9.1 and colored with exactly $k+1$ colors.

Since it is possible for $k = 3$ (see claim 9.1), Theorem 9.2 holds. ∎

9.5. Bounds for permutation graphs in a more general model

Our second generalization does not make sense for the application of the track-assignement. However, it is natural in the problem of the steel industry [SPI 06]. We consider here that the permutation is not delivered from left-to-right, but may be delivered in any order.

THEOREM 9.3.– *The m-First Fit algorithm has a performance ratio between $\frac{m}{2} + \frac{1}{2k-1}$ and $\frac{m}{2} + \frac{1}{2}$ for any natural $k \geqslant 2$ and $m \geqslant 2$.*

Proof. The upper bound is given by Plumettaz and Bouille [PLU 06]. For the lower bound, consider the instance given by algorithm 1. ∎

Algorithm 1

Input: A natural number χ

Output: A permutation graph delivered online such that m-First Fit will have a performance ratio of $\frac{m}{2} + \frac{1}{2\chi-1}$.

1: Let ρ be a real number such that $\rho << \frac{1}{2\chi}$
2: Let $\varepsilon \leftarrow \frac{\rho}{2\chi}$
3: Write "The bound is m", where $m = \frac{2\chi-1}{p}$ and p is an arbitrary natural number.
4: **for** int $i = 0..(\chi - 2)$ **do**
5: **for** int $j = 0..(\chi - 2 - i)$ **do**
6: Present a vertex $u^1_{(i,j)} = (\chi - i + i\rho + 2j\varepsilon, 1 + i\rho + 2j\varepsilon)$
7: Present a vertex $u^2_{(i,j)} = (\chi - i + i\rho + 2j\varepsilon + \varepsilon, \chi - i - j + i\rho + 2j\varepsilon + \varepsilon)$
8: **end for**
9: **end for**
10: **for** int $i = 0..(\chi - 2)$ **do**
11: **for** int $j = 0..(\chi - 2 - i)$ **do**
12: Present a vertex $v^1_{(i,j)} = (\chi - i + \chi\rho + (i+1)\rho - 2j\varepsilon, 1 + \chi\rho + (i+1)\rho - 2j\varepsilon)$
13: Present a vertex $v^2_{(i,j)} = (j + 1 + \chi\rho + (i+1)\rho - 2j\varepsilon + \varepsilon, 1 + \chi\rho + (i + 1)\rho - 2j\varepsilon + \varepsilon)$
14: **end for**
15: **end for**
16: **for** int $i = 0..(\chi - 1)$ **do**
17: Present a vertex $w^1_i = (\chi - i + \chi\rho + i\rho + \varepsilon, 1 + \chi\rho + i\rho + \varepsilon)$
18: **end for**

Note that:

$$\chi - i - j \geqslant 2 \ \forall (i,j) \qquad [9.16]$$

For an easier understanding of the equations herafter, assume that terms in ε are neglectable compared to terms in ρ, which are themselves neglectable compared to terms in natural numbers.

Clearly, for every pair (i,j), $u^1_{(i,j)}$ and $u^2_{(i,j)}$ are in a stable set. Since they are presented consecutively, they are colored with the same color. Similarly, $v^1_{(i,j)}$ and $v^2_{(i,j)}$ are also in a stable set and are also colored with the same color.

Let us now prove that vertices $u^l_{(i,j+k)}$, where k is a strictly positive natural number such that the vertex $u^l_{(i,j+k)}$ exists and $l \in \{1,2\}$ are all linked by an edge to $u^2_{(i,j)}$. Indeed:

$$u^1_{(i,j+k)}|_x = \chi - i + i\rho + 2(j+k)\varepsilon = \chi - i + i\rho + 2j\varepsilon + \varepsilon + (2k-1)\varepsilon = u^2_{(i,j)}|_x + (2k-1)\varepsilon$$

$$u^1_{(i,j+k)}|_x > u^2_{(i,j)}|_x \qquad [9.17]$$

$$u^2_{(i,j)}|_y = \chi - i - j + i\rho + 2j\varepsilon + \varepsilon \geqslant_{[9.16]} 2 + i\rho + 2j\varepsilon + \varepsilon$$
$$u^2_{(i,j)}|_y \geqslant 1 + i\rho + 2j\varepsilon + \varepsilon + 1 > 1 + i\rho + 2j\varepsilon + k\varepsilon$$

$$u^2_{(i,j)}|_y > u^1_{(i,j+k)}|_y \qquad [9.18]$$

Equations [9.17] and [9.18] tell us that $u^2_{(i,j)}$ and $u^1_{(i,j+k)}$ form a clique:

$$u^2_{(i,j+k)}|_x = u^2_{(i,j)}|_x + 2k\varepsilon > u^2_{(i,j)}|_x \qquad [9.19]$$

$$u^2_{(i,j+k)}|_y = \chi - i - (j+k) + i\rho + 2(j+k)\varepsilon + \varepsilon = \underbrace{\chi - i - j + i\rho + 2j\varepsilon + \varepsilon}_{u^2_{(i,j)}|_y} - k + 2k\varepsilon$$

$$u^2_{(i,j+k)}|_y < u^2_{(i,j)}|_y \qquad [9.20]$$

Equations [9.19] and [9.20] tell us that $u^2_{(i,j)}$ and $u^2_{(i,j+k)}$ form a clique. Thus, for a given i, each time the algorithm enters the loop starting at step 1, a new color is opened.

We now want to prove that the elements $u^l_{(i+k,j_a)}$ are always linked to the elements $u^1_{(i,j_b)}$ for all natural numbers k, j_a, j_b such that the vertices $u^l_{(i+k,j_a)}$ and $u^1_{(i,j_b)}$ exist and $l \in \{0, 1\}$. First, note that:

$$2j_1\varepsilon - k - k\rho < 2j_2\varepsilon \quad \forall k, j_a, j_b \qquad [9.21]$$

$$u^1_{(i+k,j_a)}|_x = \chi - i + i\rho + 2j_a\varepsilon - k + k\rho$$
$$u^1_{(i,j_b)}|_x = \chi - i + i\rho + 2j_b\varepsilon$$

Thus, according to equation [9.21], $u^1_{(i+k,j_a)}|_x < u^1_{(i,j_b)}|_x$. Note now that, for all j_a and j_b, $k\chi > j_b - j_a$. Thus, $2k\chi\varepsilon > 2(j_b - j_a)\varepsilon$ and $k\rho > 2j_b\varepsilon - 2j_a\varepsilon$. Besides that:

$$u^1_{(i+k,j_a)}|_y = 1 + i\rho + 2j_a\varepsilon + k\rho$$
$$u^1_{(i,j_b)}|_y = 1 + i\rho + 2j_b\varepsilon$$

Thus, $u^1_{(i+k,j_a)}|_y > u^1_{(i,j_b)}|_y$ and $u^1_{(i+k,j_a)}$ and $u^1_{(i,j_b)}$ are always linked by an edge for all k, j_a, j_b. Let us now look at $u^2_{(i+k,j_a)}$:

$$u^2_{(i+k,j_a)}|_y = \chi - (i+k) - j_a + (i+k)\rho + 2j_a\varepsilon + \varepsilon$$

Since $\chi - (i+k) - j_a \geqslant 2$:

$$u^2_{(i+k,j_a)}|_y \geqslant 1 + i\rho + 2j_a\varepsilon + 1 + k\rho + \varepsilon > 1 + i\rho + 2j_b\varepsilon$$

$$u^2_{(i+k,j_a)}|_y > u^1_{(i,j_b)}|_y$$

and:

$$u^2_{(i+k,j_a)}|_x = \chi - i + i\rho + 2j_a\varepsilon - k - k\rho + \varepsilon < \chi - i + i\rho + 2j_b\varepsilon$$

$$u^2_{(i+k,j_a)}|_x < u^1_{(i,j_b)}|_x$$

Thus, $u^l_{(i+k,j_a)}$ and $u^1_{(i,j_b)}$ form a clique for all natural numbers k, j_a, j_b such that the vertices $u^l_{(i+k,j_a)}$ and $u^1_{(i,j_b)}$ exist. Each time the algorithm passes through step 1, a new set of colors must be opened.

Similarly, it can be proved that each time the algorithm goes through step 1, a new color must be opened. In addition, the same reasoning can be used to prove that:

– All vertices $v^1_{(i_0,j_a)}$ share an edge with the vertices $u^2_{(i,j_b)}$ for all $i \leqslant i_0$.
– All vertices $v^1_{(i_0,j_a)}$ share an edge with the vertices $u^1_{(i,j_b)}$ for all $i > i_0$.
– All vertices $v^2_{(i_a,j_0)}$ share an edge with the vertices $u^1_{(i,j_b)}$ for all $i \leqslant \chi - j_0 - 1$.
– All vertices $v^2_{(i_a,j_0)}$ share an edge with the vertices $u^2_{(i,j_b)}$ for all $i > \chi - j_0 - 1$.

Thus, the vertices v may not have the same colors as the vertices u. Finally, all vertices w_{i_0} share an edge with:
– All vertices $u^2_{(i,j)}$ for all $i_0 \leqslant i$.
– All vertices $u^1_{(i,j)}$ for all $i_0 > i$.
– All vertices $v^2_{(i,j)}$ for all $i_0 \leqslant i$.
– All vertices $v^1_{(i,j)}$ for all $i_0 > i$.

The total number of colors used by First-Fit on this instance is thus $\frac{\chi(\chi-1)}{2}$ for the loop starting at step 1, $\frac{\chi(\chi-1)}{2}$ for the loop starting at step 1 and χ for the loop starting at step 1, for a total of χ^2 colors.

Let us now look for an optimal coloring of this instance. First, draw χ straight lines on the plane with coordinates $d_i : y = x - i$, $i \in [0..(\chi - 1)]$.

It is easy to verify that the vertex $u^1_{(i,j)}$ is on the line $d_{\chi-i-1}$ and that the vertex $u^2_{(i,j)}$ is on the line d_j. Thus, on each line, there are $\chi - 2$ vertices of type u.

Similarly, the vertex $v^1_{(i,j)}$ is on the line $d_{\chi-i-1}$ and that the vertex $v^2_{(i,j)}$ is on the line d_j. Thus, on each line, there are $\chi - 2$ vertices of type v.

Finally, the vertex w^1_i is on the line $d_{\chi-i-1}$. Thus, on each line, there is 1 vertex of type w.

On each line d_i there are thus $2\chi - 1$ vertices. In an optimal unbounded coloring, each line could be used as one color and all colors would have the same size. Since $m = \frac{2\chi-1}{p}$ (see step 1), we have an evident solution for the bounded coloring, which is $\chi_m = p\chi$. The performance ratio c of m-First Fit on this instance is thus $c = \frac{\chi^2}{\chi_m}$:

$$c = \frac{\chi^2}{p\chi} = \frac{\chi}{p} = \frac{m}{2} + \frac{1}{2\chi - 1}$$

This ends the proof of Theorem 9.3.

9.6. Conclusion

The competitive analysis of the bounded coloring of permutations showed that the m-FF algorithm is optimal for this problem. However, the bounded coloring of permutations corresponds to a specific track-assignment problem under the midnight condition. If we remove this condition we have to analyze the online bounded-coloring of circle graphs. We do not know if the m-FF algorithm is optimal for this problem: we can only say that the $2 - 1/l$, $l = \min\{m, k\}$ is a lower bound since permutation graphs are included in circle graphs.

The work reported in this paper should be seen as a first contribution to the competitive analysis of online problems arising in the area of rolling stock allocation: further analysis should be carried out for more general instances of this problem.

9.7. Bibliography

[BER 76] BERGE C., *Graphs and Hypergraphs*, North Holland, Amsterdam, 1976.

[BLA 99] BLASUM U., BUSSIECK M., HOCHSTÄTTLER W., MOLL C., SCHEEL H., WINTER T., "Scheduling Trams in the Morning", *Mathematical Methods of Operations Research*, vol. 49, num. 1, p. 137–148, 1999.

[CHV 84] CHVÁTAL V., "Perfectly ordered graphs", *Topics on perfect graphs*, vol. 88 of *North-Holland Math. Stud.*, p. 63–65, North-Holland, Amsterdam, 1984.

[COR 07] CORNELSEN S., DI STEFANO G., "Track assignment", *Journal of Discrete Algorithms*, 2007, To appear.

[DAH 00] DAHLHAUS E., HORAK P., MILLER M., RYAN J. F., "The train marshalling problem", *Discrete Appl. Math.*, vol. 103, num. 1-3, p. 41–54, 2000.

[DEM 07] DEMANGE M., LEROY-BEAULIEU B., Online Coloring of Comparability Graphs: some results, Report , École Polytechnique Fédérale de Lausanne, 2007.

[DIS 04] DI STEFANO G., KOCI M. L., "A Graph Theoretical Approach To The Shunting Problem.", *Electr. Notes Theor. Comput. Sci.*, vol. 92, p. 16-33, 2004.

[DIS 06] DI STEFANO G., KRAUSE S., LÜBBECKE M. E., ZIMMERMANN U. T., "On Minimum-Modal Partitions of Permutations.", *LATIN*, p. 374-385, 2006.

[FRE 05] FRELING R., LENTINK R. M., KROON L. G., HUISMAN D., "Shunting of Passenger Train Units in a Railway Station", *Transportation Science*, vol. 39, p. 261–272, 2005.

[GAL 01] GALLO G., MIELE F. D., "Dispatching Buses in Parking Depots", *Transportation Science*, vol. 35, num. 3, p. 322–330, INFORMS, 2001.

[GAV 73] GAVRIL F., "Algorithms for a maximum clique and a maximum independent set of a circle graph", *Networks*, vol. 3, p. 261–273, 1973.

[GOL 04] GOLUMBIC M. C., *Algorithmic Graph Theory and Perfect Graphs*, Annals of discrete mathematics, Elsevier, 2nd edition, 2004.

[HAM 05a] HAMDOUNI M., DESAULNIERS G., MARCOTTE O., SOUMIS F., VAN PUTTEN M., Dispatching Buses in a Depot Using Block Patterns, Report num. G-2004-51, Les Cahiers du GERAD, 2005.

[HAM 05b] HAMDOUNI M., DESAULNIERS G., MARCOTTE O., SOUMIS F., VAN PUTTEN M., Parking Buses in a Depot Using Block Patterns: A Benders Decomposition Approach for Minimizing Type Mismatches, Report num. G-2004-70, Les Cahiers du GERAD, 2005.

[HE 00] HE S., SONG R., CHAUDHRY S. S., "Fuzzy dispatching model and genetic algorithms for railyards operations", *European Journal of Operations Research*, vol. 124, num. 2, p. 307–331, 2000.

[NIK 00] NIKOLOPOULOS S. D., PAPADOPOULOS C., "On the performance of the First-Fit coloring algorithm on permutation graphs", *Inf. Process. Lett.*, vol. 75, num. 6, p. 265–273, Elsevier North-Holland, Inc., 2000.

[PLU 06] PLUMETTAZ M., BOUILLE D., Coloration online Bornée dans les Graphes de Permutations, Technical Report, Ecole Polytechnique Fédérale de Lausanne, 2006.

[ROS 03] ROSSI A., Il problema dell'ordinamento dei treni in un deposito: modellazione e soluzione algoritmica, Master's thesis, Università dell'Aquila, 2003.

[SPI 06] SPIEKSMA F., MOONEN L., "Partitioning a weighted partial order", *5th International Colloquium on Operational Research (GO V), Leukerbad (Switzerland)*, 2006.

[WIN 00] WINTER T., ZIMMERMANN U. T., "Real-time dispatch of trams in storage yards", *Ann. Oper. Res.*, vol. 96, p. 287–315, 2000, Mathematics of industrial systems, IV (Valparaiso, 1996).

[ZWA 96] ZWANEVELD P. J., DAUZÈRE-PÉRÈS S., VAN HOESEL S., KROON L. G., "Routing trains through railway stations: model formulation and algorithms", *Transportation Science*, vol. 30, p. 181-194, 1996.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 10

Complexity and Approximation Results for the Min Weighted Node Coloring Problem

10.1. Introduction

This chapter presents some recent works accomplished by the authors ([WER 02, DEM 04, DEM 04, ESC 06]) about the complexity and the approximation properties of the MIN WEIGHTED NODE COLORING problem, which was originally studied in Guan and Zhu [GUA 97] and more recently in [PEM 04, PEM 05] under the name MAX COLORING.

A k-coloring of $G = (V, E)$ is a partition $\mathcal{S} = (S_1, \ldots, S_k)$ of the node set V of G into stable sets S_i (a stable set is a set of pairwise non-adjacent nodes). In the usual case, the objective is to determine a node coloring minimizing k. A natural generalization of this problem is obtained by assigning a strictly positive integer weight $w(v)$ for any node $v \in V$ and defining the weight of stable set S of G as $w(S) = \max\{w(v) : v \in S\}$. Then, the objective is to determine a node coloring $\mathcal{S} = (S_1, \ldots, S_k)$ of G minimizing the quantity $w(\mathcal{S}) = \sum_{i=1}^{k} w(S_i)$. This problem is easily shown to be **NP**-hard; it suffices to consider $w(v) = 1, \forall v \in V$ and MIN WEIGHTED NODE COLORING becomes the traditional node coloring problem. Other versions of weighted colorings have been studied in Hassin and Monnot [HAS 05].

Chapter written by Marc DEMANGE, Bruno ESCOFFIER, Jérôme MONNOT, Vangelis Th. PASCHOS and Dominique DE WERRA.

MIN WEIGHTED NODE COLORING arises in some types of scheduling problems. Consider that each node v of a graph G corresponds to some operation to be processed. The weight of a node corresponds to the processing time of the job associated with the node. We have to assign each operation to one time slot in such a way that for each time slot all operations assigned to this slot are compatible. As usual these incompatibility constraints are represented by the edges of the graph G. In other words, a slot corresponds to an independent set in the graph. In our situation, the lengths of the time slots $1, 2, \ldots, k$ are not given in advance; assuming that the jobs scheduled in time slot i may be processed simultaneously, the amount of time needed will be given by $w(S_i) = \max\{w(v) : v \in S_i\}$. As a consequence, the total amount of time needed to complete all jobs will be: $w(S) = \max\{w(v) : v \in S\}$ where $\mathcal{S} = (S_1, \ldots, S_k)$.

As we shall see, the number k of time slots is not always given in advance; it may turn out that if $\chi(G)$ is the chromatic number of G, we may have to use $k > \chi(G)$ time slots. So we shall admit in our formulation that some of the subsets S_i are possibly empty. We shall furthermore assume that all weights are positive integer numbers.

Our weighted coloring problem is related to the *batch scheduling* problem which has been studied by several authors (see for instance Potts and Kovalyov [POT 00] for a survey, or Potts *et al.* [POT 01] for a specific case). In the papers on batch scheduling, there are usually incompatibility constraints between operations belonging to a same job, or precedence constraints. The general case of incompatibility requirements represented by an arbitrary graph is formulated in Boudhar and Finke [BOU 00], where they consider the complement of our graph: edges indicate compatibilities and they partition the node set into cliques. On the other hand, several types of requirements are introduced, like sequencing constraints or limitations in the size of a batch. Our approach is characterized by the underlying graph theoretical formulation. Finally, in the special case where the graph G is the line graph $L(H)$ of a complete bipartite graph $H = K_{n,n}$ we have a known problem of decomposition of traffic matrices which arises in satellite communication (see Burkard [BUR 85] and Rendl [REN 85] for more references). In the case where $G = L(H)$ is the line graph of an arbitrary multi-graph H and where all weights are equal, the problem MIN WEIGHTED NODE COLORING amounts to finding the *chromatic index* of H, i.e., the smallest number of edges in an edge coloring of H; this problem is difficult (see Holyer [HOL 81]).

Consider an instance $I = (G, w)$ of MIN WEIGHTED NODE COLORING and a polynomial time algorithm A calculating feasible coloring $\mathcal{S}$ for MIN WEIGHTED NODE COLORING. Denote by $w(\mathcal{S})$ the value of the solution $\mathcal{S}$ calculated by A on I and by $\mathrm{opt}(I)$ the value of an optimal solution for I. The quality of A is expressed by the ratio (called approximation ratio in what follows) $\rho_{\mathtt{A}}(I) = w(\mathcal{S})/\mathrm{opt}(I)$, and the quantity $\rho_{\mathtt{A}} = \sup\{\rho_{\mathtt{A}}(I)\}$. A very favorable situation for polynomial approximation occurs when for any $\varepsilon > 0$ there exists an algorithm that achieves a ratio bounded above by $1 + \varepsilon$. We call such algorithms a *polynomial time approximation scheme* (PTAS) . The complexity of such schemes may be polynomial or exponential in $1/\varepsilon$

(they are always polynomial in the size of the instances). A polynomial time approximation scheme with complexity polynomial also in $1/\varepsilon$ is called *fully polynomial time approximation scheme* (FPTAS) .

In this chapter, we survey the complexity and the approximability properties of MIN WEIGHTED NODE COLORING in different classes of graphs. Of course, complexity issues only occur in classes of graphs where the usual coloring problem is polynomial.

In section 10.2, we present some general structural properties of optimum solution. We also give an approximation result in general graphs. Finally, we present some general links between MIN WEIGHTED NODE COLORING and other coloring problems, leading to interesting approximation ratios in some traditional classes of graphs.

In section 10.3, we deal with triangle free planar graphs and we show that, for this family, the problem studied is **NP**-hard, even if we further restrict ourselves to graphs with node-degree not exceeding 4.

Then, we deal with bipartite graphs. We show that MIN WEIGHTED NODE COLORING is **NP**-hard in bipartite graphs, even if we restrict ourselves to planar bipartite graphs or to P_{21}-free bipartite graphs. It is interesting to observe that these results are obtained as corollaries of a kind of generic reduction from the precoloring extension problem shown to be **NP**-complete in Bodlaender *et al.* [BOD 94], Hujter and Tuza [HUJ 93, HUJ 96], Kratochvil [KRA 93]. Then, we slightly improve the last result to P_8-free bipartite graphs and show that the problem becomes polynomial in P_5-free bipartite graphs. Finally, we note that, in general bipartite graphs, the upper bound of 8/7 given in section 10.2.2 matches a lower bound of $8/7 - \varepsilon$.

Subsequently, in section 10.5, we deal with the approximation of MIN WEIGHTED NODE COLORING in split graphs. We prove that MIN WEIGHTED NODE COLORING is strongly **NP**-hard in such graphs, even if the nodes of the input graph receive only one of the two distinct weights. It follows that this problem cannot be solved by fully polynomial time approximation schemes. Then, we show that MIN WEIGHTED NODE COLORING in split graphs can be solved by a polynomial time approximation scheme.

Section 10.6 is devoted to cographs, or equivalently P_4-free graphs. While the problem is **NP**-hard in P_5-free graphs (since split graphs are P_5-free), it is polynomial in cographs.

Then, in section 10.7, we answer a question raised in [GUA 97] by proving that MIN WEIGHTED NODE COLORING is (strongly) **NP**-hard in interval graphs.

The main results of the chapter are summarized in Table 10.1

Class of graphs	Status	Approximability	Inapproximability
General	**NP**-hard	$O(\log(n)/n)$	$O(n^{1/7-\varepsilon})$
K_3-free planar	**NP**-hard	$27/19$	$7/6-\varepsilon$
Complement of K_3-free	Polynomial		
Bipartite	**NP**-hard	$8/7$	$8/7-\varepsilon$
P_k-free Bipartite	**NP**-hard if $k \geqslant 8$	$8/7$	$32/31-\varepsilon$ if $k \geqslant 21$
P_5-free Bipartite	Polynomial		
k-Tree	? (even for $k=1$)	PTAS	
Split	**NP**-hard	PTAS	no FPTAS
Interval	**NP**-hard	2 [PEM 05]	no FPTAS
Cographs	Polynomial		

Table 10.1. *Summary of main results*

In the remainder of the chapter, we shall assume that for any weighted node or edge coloring $\mathcal{S} = (S_1, \ldots, S_\ell)$ considered, we will have $w(S_1) \geqslant \ldots \geqslant w(S_\ell)$. Moreover, without specification, we assume that $\forall v \in V,\ w(v) > 0$. For graph theoretical terms which are not defined here, see [BER 73].

10.2. General results

10.2.1. *Structural properties*

Before examining the variations on MIN WEIGHTED NODE COLORING, we shall give a general property which will be needed later. Let us recall that a subset $S \subseteq V$ is *maximum* with respect to a property `Pred` if its cardinality is maximum among all subsets which have property `Pred`. We shall say that S is *maximal* with respect to `Pred` if S has property `Pred` but any subset of V which properly contains S does not have property `Pred`. A maximum subset S is maximal but the opposite is not true. Now consider an instance of MIN WEIGHTED NODE COLORING given by a weighted graph $I = (G, w)$ with $G = (V, E)$; we want to find a k-coloring $\mathcal{S} = (S_1, \ldots, S_k)$ of (the nodes of) G such that $w(\mathcal{S})$ is minimum. k is not given in advance; we may choose $k = |V|$ and accept to have some subsets $S_i = \{\,\}$. For practical purposes it will be convenient to use the following.

THEOREM 10.1.– *Given $I = (G, w)$, there exists a k-coloring $\mathcal{S} = (S_1, \ldots, S_k)$ with $w(\mathcal{S}) = \min\{w(\mathcal{S}) : \mathcal{S}$ is a coloring of $G\}$ where for $i = 1, \ldots, k$ we have:*

(i) $S_i \neq \{\}$;

(ii) S_i *is a maximal stable set in* $G \setminus (S_1 \cup \ldots \cup S_{i-1})$;

(iii) $w(S_1) \geqslant w(S_2) \geqslant \ldots \geqslant w(S_k)$.

Proof. Assume that we have obtained a p-coloring $\mathcal{S} = (S_1, \ldots, S_p)$ which minimizes $w(\mathcal{S})$ over all possible colorings of G. By reordering the subsets S_i and removing the empty sets we get a k-coloring $\mathcal{S}$ satisfying (i) and (iii). If S_1 is not maximal we may find in some S_l ($l \geqslant 2$) some elements which can be added to S_1 in order to make it maximal; this will not increase $w(S_1)$ because (iii) holds. We may repeat this for $S_2, \ldots, S_{k-1}$ and obtain the required k-coloring. ∎

Let us denote by $\Delta(G)$ the maximum degree of the nodes in G.

COROLLARY 10.1.– For $I = (G, w)$, there always exists an optimal weighted coloring $\mathcal{S} = (S_1, \ldots, S_k)$ which satisfies $k \leqslant \Delta(G) + 1$.

Proof. Assume that we have a k-coloring $\mathcal{S}$ which minimizes $w(\mathcal{S})$ and where each S_i is maximal in $G \setminus S_1 \setminus \ldots \setminus S_{i-1}$. Suppose that $S_l \neq \{\}$ (with $l > \Delta(G) + 1$). Then let $v \in S_l$; since v has at most $\Delta(G)$ neighbors in G there is at least one stable set S_r ($r \leqslant \Delta(G) + 1$) containing no neighbor of v. Hence $S_r \cup \{v\}$ is a stable set of $G \setminus S_1 \setminus \ldots \setminus S_{r-1}$; this contradicts the fact that S_r was maximal in this graph. ∎

COROLLARY 10.2.– If $I = (G, w)$ where $G = (V, E)$ is the line-graph $L(G)$ of a graph H, then there always exists an optimal weighted k-coloring $\mathcal{S}$ of the nodes of G which satisfies $k \leqslant 2\Delta(H) - 1$.

Proof. Each edge $e = [u, v]$ of H corresponds to a node e of $L(H)$ which belongs to two maximal cliques K_u, K_v whose nodes correspond to the edges of H adjacent to u (for K_u) and to v (for K_v).

Since $\max\{|K_u|, |K_v|\} \leqslant \Delta(H)$, the degree of e in $L(H)$ satisfies: $d_{L(H)}(e) = (|K_u| - 1) + (|K_v| - 1) \leqslant 2\Delta(H) - 2 \leqslant \Delta(L(H))$. The result follows from Corollary 10.1. ∎

Note that if $G = L(H)$ for some H, then the node coloring problem in G is equivalent to an edge coloring problem in H where the weights are assigned to the edges of H. The stable sets S_i in G become matchings M_i in H. We shall consider later the case where H is bipartite. We can easily show that in Corollary 10.1 we have $k \leqslant p(\omega(G) - 1) + 1$ where $\omega(G)$ is the maximum cardinality of a clique in G and p is the maximum number of (maximal) cliques in which one node of G is contained. If G is a line-graph $L(H)$, then $p = 2$ and $\omega(G) = \Delta(G)$, so Corollary 10.2 follows. In

fact, it follows from Theorem 10.1 that the number k of colors in an optimal weighted k-coloring can be bounded above by any bound on the chromatic number which is derived by a sequential coloring algorithm which gives maximal stable sets in the subgraph generated by the colored nodes. In particular, the bounds of Welsh-Powell and of Matula are valid for k (see for instance de Werra [WER 90]).

We can also establish the following property of the size of any optimal k-colorings $\mathcal{S}$ for MIN WEIGHTED NODE COLORING.

LEMMA 10.1.– *Let $I = (G, w)$ be an instance of* MIN WEIGHTED NODE COLORING *and let $q = \chi(G)$ be the chromatic number of G. Assume that $w(v) \in \{t_1, \ldots, t_r\}$ with $t_1 > \ldots > t_r$ for each node v. Then every k-coloring $\mathcal{S} = (S_1, \ldots, S_k)$ which minimizes $w(\mathcal{S})$ satisfies: $w(S_i) > w(S_{i+q-1})$, for any $i \leqslant k - q$. In particular, $k \leqslant 1 + r(\chi(G) - 1)$.*

Proof. Assume that $\chi(G) \geqslant 2$ and there is an optimum k-coloring which does not have the property; let $\mathcal{S} = (S_1, \ldots, S_k)$ be this coloring and choose the smallest i such that $w(S_i) = \ldots = w(S_{i+q-1}) \geqslant w(S_k)$. We have $i \leqslant k - q$ by assumption. Now $S_i \cup S_{i+1} \cup \ldots \cup S_k$ generates a subgraph G' of G; we have therefore $\chi(G') \leqslant \chi(G) = q$, so there exists a q-coloring $(S'_i, \ldots, S'_{i+q-1})$ of G' (with $i + q - 1 < k$). Assuming that $w(S'_i) \geqslant w(S'_{i+1}) \geqslant \ldots \geqslant w(S'_{i+q-1})$ we have $w(S'_i) = w(S_i)$ and $w(S'_{i+s}) \leqslant w(S'_i) = w(S_{i+s})$ for $s = 1, \ldots, q-1$. Setting $S'_j = S_j$ for $j = 1, \ldots, i-1$ we get an $(i + q - 1)$-coloring $\mathcal{S}' = (S'_1, \ldots, S'_{i+q-1})$ of G with $w(\mathcal{S}') < w(\mathcal{S})$ which is a contradiction. ∎

For instance, if G is a bipartite graph, then any coloring $\mathcal{S} = (S_1, \ldots, S_k)$ minimizing $w(\mathcal{S})$ satisfies: $w(S_1) > w(S_2) > \ldots > w(S_{k-1}) \geqslant w(S_k)$. In particular, $k \leqslant 1 + r$.

10.2.2. *Approximation results*

10.2.2.1. *General graphs*

The approximation result presented in this section is based upon the so-called *master-slave approximation strategy*. Consider an **NP**-hard covering problem which consists of covering the nodes of the input graph G, of order n, by the minimum number of subgraphs G' satisfying a certain property `Pred`. Most of these problems can be approximated by the following strategy:

1) find a maximum subgraph G' of G verifying `Pred`;

2) delete $V(G')$ from V;

3) repeat steps 1 and 2 in the remaining graph until $V = \{\}$.

The maximization problem solved at step 1 is called the *slave*, while the original minimization problem is called the *master*. These terms are due to Simon [SIM 90]

who points out the fact that if the slave problem is polynomial then the master problem is approximable within $O(\log n)$. A standard example of master-slave approximation is given by Johnson (in [JOH 74], algorithm D3 at the end of section 7 devoted to graph coloring). At each iteration this algorithm calculates a maximum stable set of the remaining graph, it colors its nodes with a new color and removes them from the graph. The master problem in this case is the minimum graph coloring, while the slave problem is the maximum stable set. The following result will be used later on.

THEOREM 10.2.– *[ALF 00] Let $\mathcal{G}'$ be a class of graphs. In the master-slave approximation strategy for weighted problems, if the weighted slave problem is approximable within ratio ρ in $\mathcal{G}'$, then the weighted master problem is approximable within ratio $\ln n/\rho$ in $\mathcal{G}'$.*

For our problem, the (maximization) slave problem, denoted by SLAVE_WC, consists of *determining a stable set S^* maximizing the quantity $|S|/w(S)$, over all stable sets S, where $w(S) = \max\{w(v) : v \in S\}$*. Consequently, the overall algorithm `W_COLOR` which we devise for weighted coloring can be outlined as follows:

`W_COLOR`

1) solve SLAVE_WC in G; let $S^{\overline{v}}$ be the solution obtained; set $V = V \setminus S^{\overline{v}}$, $G = G[V]$;

2) color the nodes of $S^{\overline{v}}$ with a new color;

3) repeat steps 1 and 2 until all the nodes of the input graph are colored.

LEMMA 10.2.– *SLAVE_WC is approximable within $O(\log^2 n/n)$ in polynomial time.*

Proof. We show in what follows that SLAVE_WC is equi-approximable with the maximum stable set problem. Consider the following algorithm, called `SLAVE_WC` in what follows:

`SLAVE_WC`

1) for every $v \in V$ do:

a) let G_v be the subgraph of G induced by all nodes with weight not exceeding $w(v)$;

b) run the algorithm $\mathcal{A}$ of [BOP 92] on G_v and store the output stable set S^v;

2) among the sets stored in step 1b, choose $\overline{v}$ maximizing $|S^{\overline{v}}|/w(\overline{v})$.

It is easy to see that since the algorithm of [BOP 92] runs in polynomial time, algorithm `SLAVE_WC` does so. On the other hand, for any node v, S^v is indeed a stable set containing v; moreover, $w(S^v) = w(v)$ since all the nodes heavier than v have been removed from G.

Denote by S^* an optimal solution of SLAVE_WC in G, and set:

$$v^* = \texttt{argmax}\{w(v) : v \in S^*\}$$

If there exists more than one v^*, fix one of them. Of course v^* has been considered in step 1 of algorithm SLAVE_WC and set S^{v^*} has been calculated and stored in step 1b. Consequently (we recall that $w(S^*) = w(v^*)$):

$$\frac{\left|S^{\overline{v}}\right|}{w\left(S^{\overline{v}}\right)} \geqslant \frac{\left|S^{v^*}\right|}{w\left(v^*\right)} = \frac{\left|S^{v^*}\right|}{w\left(S^*\right)} \qquad [10.1]$$

On the other hand, when S^{v^*} was calculated, the whole set S^* was present in the current graph. Denote it by G^*, denote by $S(G^*)$ a maximum stable set of G^* and by ρ_S the approximation ratio of a maximum stable set algorithm. Then:

$$\rho_S = \frac{\left|S^{v^*}\right|}{\left|S\left(G^*\right)\right|} \leqslant \frac{\left|S^{v^*}\right|}{\left|S^*\right|} \qquad [10.2]$$

By combining [10.1] and [10.2], we obtain:

$$\frac{\frac{\left|S^{\overline{v}}\right|}{w(S^{\overline{v}})}}{\frac{|S^*|}{w(S^*)}} \geqslant \frac{\frac{\left|S^{v^*}\right|}{w(S^*)}}{\frac{|S^*|}{w(S^*)}} \geqslant \frac{\left|S^{v^*}\right|}{\left|S\left(G^*\right)\right|} = \rho_S \qquad [10.3]$$

Taking into account that, in terms of n, the best known approximation ratio for the maximum stable set problem is $O(\log^2 n/n)$ [BOP 92], [10.3] concludes the proof of the lemma. ∎

Using Theorem 10.2 and Lemma 10.2, the following holds for algorithm W_COLOR and the weighted coloring problem.

THEOREM 10.3.– MIN WEIGHTED NODE COLORING *can be approximately solved in polynomial time within approximation ratio* $O(n/\log n)$.

10.2.2.2. *k-colorable graphs*

We study in this section the approximability of the MIN WEIGHTED NODE COLORING problem in natural classes of graphs which are colorable with a few number of colors. We first focus on subfamilies of k-colorable graphs where the minimum coloring problem is polynomial. Note that it is proved in [PEM 05] that if the usual

min coloring problem is polynomial in a class of graphs, then MIN WEIGHTED NODE COLORING is approximable within ratio $4 - 3/k$ in this class of graphs.

Let us prove the following theorem.

THEOREM 10.4.– *Let $\mathcal{G}$ be a class of k-colorable graphs, where a k-coloring is computable in polynomial time. Then, in $\mathcal{G}$,* MIN WEIGHTED NODE COLORING *is approximable within ratio $k^3/(3k^2 - 3k + 1)$.*

Proof. Consider any graph $G = (V, E) \in \mathcal{G}$ of order n and assume that any $v_i \in V$ has weight $w_i = w(v_i)$. Suppose that $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_n$. Consider an optimal weighted coloring $\mathcal{S}^* = (S_1^*, \cdots, S_l^*)$, with $w(S_1^*) \geqslant \cdots \geqslant w(S_l^*)$ and denote by i_k^*, the index of the heaviest node in color S_k^* (hence, $w(S_k^*) = w_{i_k^*}$), by V_i the set of nodes $\{v_1, \cdots, v_i\}$ (hence, $V_n = V$) and by $G[V']$ the subgraph of G induced by $V' \subseteq V$.

We calculate several colorings of G and choose as final solution the best one among the colorings calculated. We first calculate a k-coloring $\mathcal{S}^0$ of G. Clearly:

$$w\left(\mathcal{S}^0\right) \leqslant kw_1 = kw\left(S_1^*\right) \qquad [10.4]$$

Then, for $j = 2, \cdots, n+1$, we do the following:

– if $G[V_{j-1}]$ is bipartite then:

- consider the best weighted 2-coloring (S_1^j, S_2^j) among the 2-colorings of $G[V_{j-1}]$ (S_2^j may be empty);

- color the remaining nodes $v_j, \cdots, v_n$ with k colors $(S_3^j, S_4^j, \cdots, S_{k+2}^j)$, thus obtaining a coloring $\mathcal{S}^j = (S_1^j, S_2^j, \cdots, S_{k+2}^j)$ of G.

Note that the first step is polynomially computable (merge optimally the unique 2-colorings of any connected component).

Consider now the iterations where $j = i_2^*$ and $j = i_3^*$. For $j = i_2^*$, V_{j-1} is an independent set; hence, $S_1^j = V_{j-1}$. We get in this case:

$$w(\mathcal{S}^j) \leqslant w_1 + kw_j = w\left(S_1^*\right) + kw\left(S_2^*\right) \qquad [10.5]$$

On the other hand, for $j = i_3^*$, $G[V_{j-1}]$ is bipartite; hence, $w(S_1^j) + w(S_2^j) \leqslant w(S_1^*) + w(S_2^*)$. In this case:

$$w(\mathcal{S}^j) \leqslant w\left(S_1^*\right) + w\left(S_2^*\right) + kw\left(S_3^*\right) \qquad [10.6]$$

Recall that the algorithm returns the best coloring among those calculated. Note also that if the number l of colors in $\mathcal{S}^*$ is smaller than 2, then this algorithm calculates an optimal coloring. Combination of equations [10.4], [10.5] and [10.6] with coefficients $(k-1)^2/k^3$, $k(k-1)/k^3$ and $k^2/k^3 = 1/k$, respectively, concludes that the output coloring $\mathcal{S}$ is such that: $w(\mathcal{S}) \leqslant (k^3/(3k^2-3k+1))w(\mathcal{S}^*)$ and the result follows. ∎

Note that this improves the $(4-3/k)$-approximation algorithm given in [PEM 05] for $k \leqslant 10$. Moreover, it can be applied also in classes of k-colorable graphs where the problem remains hard (such as planar graphs)

COROLLARY 10.3.– MIN WEIGHTED NODE COLORING is approximable within ratio $8/7$ in bipartite graphs.

COROLLARY 10.4.– MIN WEIGHTED NODE COLORING is approximable within ratio $27/19 < 1.42$ in polynomially 3-colorable graphs.

It is well known that the coloring problem is polynomial in planar triangle-free graphs ([GRO 59]) and that the chromatic number in these graphs is bounded by 3. Moreover, as we will see in section 10.3, on the one hand the MIN WEIGHTED NODE COLORING problem is strongly **NP**-hard and, on the other hand, the MIN WEIGHTED NODE COLORING problem cannot be approximated with performance ratio better than $\frac{7}{6} - \varepsilon$ for any $\varepsilon > 0$ unless **P**≠**NP**, in planar triangle-free graphs, even if the maximum degree is bounded by 4. Using Theorem 10.4, we obtain:

COROLLARY 10.5.– MIN WEIGHTED NODE COLORING is $27/19$-approximable in planar triangle-free graphs.

As another corollary of Theorem 10.4, MIN WEIGHTED NODE COLORING is approximable within ratio $64/37$ in polynomially 4-colorable graphs.

On the other hand, note that minimum coloring is not $(4/3-\varepsilon)$-approximable in planar graphs, that these graphs are polynomially 4-colorable and that the MIN WEIGHTED NODE COLORING problem is a generalization of the coloring problem. Putting all this together, we obtain:

THEOREM 10.5.– MIN WEIGHTED NODE COLORING *is approximable within ratio* $64/37 < 1.73$ *in planar graphs, but it is not* $(4/3-\varepsilon)$*-approximable in these graphs.*

10.2.2.3. *When list coloring is easy*

Let us define the List Coloring problem, where we want to answer the following question: given a graph $G = (V, E)$ with, for any $v \in V$, a set $L(v)$ of admissible colors, does there exists a (proper) coloring of G with colors from $L(V) = \cup_{v \in V} L(v)$

such that any node v is colored with a color from $L(v)$? The complexity of List Coloring has been widely studied (see for instance [KRA 94, JAN 97]).

THEOREM 10.6.– *If $\mathcal{G}$ is a class of t-colorable graphs (where t is a constant) where List Coloring is polynomial, then* MIN WEIGHTED NODE COLORING *admits a PTAS in $\mathcal{G}$.*

Proof. Let $\mathcal{G}$ be such a class of graphs, consider a graph $G \in \mathcal{G}$ and let $\varepsilon > 0$. Set $k = \left\lceil (t-1)(1+\frac{1}{\varepsilon}) \right\rceil$.

Consider the following algorithm. For any $k' \leqslant k$:

– consider any k'-tuple $(x_1, \cdots, x_{k'}) \in W^{k'}$ where W denotes the set of different weights used in the instance, i.e., $W = \{w(v) : v \in V\}$;

– find a k'-coloring $(S_1, \cdots, S_{k'})$ of G such that $w(S_i) \leqslant x_i, i = 1, \cdots, k'$, if such a coloring exists;

– output the best coloring among those found in the previous step.

To achieve the second step, we use the fact that List Coloring is polynomial in $\mathcal{G}$. Indeed, given $(x_1, \cdots, x_{k'}) \in W^{k'}$, we can define an instance of List Coloring on G: v_i can be colored with color S_j for all $j \in \{1, \cdots, k'\}$ such that $w(v_i) \leqslant x_j$. We can easily see that a coloring is valid for this instance of List Coloring, if and only if this coloring is such that $w(S_i) \leqslant x_i, i = 1, \cdots, k'$.

We claim that the solution calculated by this algorithm is $1+\varepsilon$-approximate, for any $\varepsilon > 0$.

Indeed, consider an optimal solution $\mathcal{S}^* = (S_1^*, \cdots, S_l^*)$, with $w(S_1^*) \geqslant \cdots \geqslant w(S_l^*)$. If $l \leqslant k$, then we have found, by our exhaustive search, a coloring $(S_1, \cdots, S_l)$ such that $w(S_i) \leqslant w(S_i^*)$ for all i, hence an optimal solution.

If $l > k$, then consider the k-tuple $(w_1, \cdots, w_k)$ where $w_i = w(S_i^*)$ for $i \leqslant k+1-t$ and $w_i = w(S_{k+1-t}^*)$ for $i \geqslant k+1-t$. If we consider the $k-t$ colors $S_i^*, i = 1, \cdots, k-t$, and any t-coloring $S_j, j = k+1-t, \cdots, k$ of the remaining nodes (the graph is polynomially t-colorable), then $w(S_j) \leqslant w(S_{k+1-t}^*)$ for $j = k+1-t, \cdots, k$. So, the algorithm finds a coloring for this particular tuple $(w_1, \cdots, w_k)$. Consequently, the solution $\mathcal{S}$ given by the algorithm is such that:

$$\begin{aligned} \frac{w(\mathcal{S})}{w(\mathcal{S}^*)} &\leqslant \frac{\sum_{i=1}^{k+1-t} w(S_i^*) + (t-1)w(S_{k+1-t}^*)}{\sum_{i=1}^{l} w(S_i^*)} \\ &\leqslant 1 + \frac{(t-1)w(S_{k+1-t}^*)}{\sum_{i=1}^{k+1-t} w(S_i^*)} \leqslant 1 + \frac{t-1}{k+1-t} \leqslant 1+\varepsilon \end{aligned}$$

Since we use less than $|W^{k+1}| \leqslant n^{k+1}$ times the algorithm for List Coloring as a subroutine, our algorithm is polynomial, hence we get the expected result. ∎

A k-tree is a graph that can be reduced to a clique of size k by deleting iteratively some nodes the neighborhood of which is a clique of size k. A partial k-tree is a subgraph of a k-tree. There are several characterizations of partial k-trees. One of them is the following: G is a partial k-tree iff G is a subgraph of a chordal graph G' with a clique number equal to $k+1$ (i.e., $\omega(G') = k+1$). k-trees are (polynomially) $k+1$-colorable, and we can get a ρ_{k+1}-approximation, but we can improve this result since List Coloring is polynomial in partial k-trees ([JAN 97]).

COROLLARY 10.6.– MIN WEIGHTED NODE COLORING admits a PTAS in partial k-trees (hence, in particular, in trees).

Although we have proposed an approximation scheme in partial k-trees, the complexity of the MIN WEIGHTED NODE COLORING problem remains open for these graphs, even for $k = 1$. A partial answer is given when we consider the case where Δ or the number of different weights is bounded by a constant. Hence, using Corollary 10.1 or Lemma 10.1, we deduce that MIN WEIGHTED NODE COLORING is polynomial in $\mathcal{G}$ if $\mathcal{G}$ is a class of k-colorable graphs (where k is a constant) where List Coloring is polynomial and if Δ or the number of different weights is bounded by a constant.

10.3. Weighted node coloring in triangle-free planar graphs

A planar graph is a graph which can be embedded in the plane such that no cross occurs between two edges. The node coloring problem in planar graphs has been shown to be **NP**-hard by Garey and Johnson [GAR 79], even if the maximum degree does not exceed 4. On the other hand, this problem becomes easy in triangle-free planar graphs (see Grotzsch [GRO 59]). A graph is triangle-free if it does not contain a triangle as induced subgraph. Note that we have proved in section 10.2.2 that MIN WEIGHTED NODE COLORING is approximable within ratio $27/19$ in triangle-free planar graphs.

Here, we show that the MIN WEIGHTED NODE COLORING problem is **NP**-hard in triangle-free planar graphs with maximum degree 4 by using a reduction from 3-SAT PLANAR, which is proved to be **NP**-complete in Lichtenstein [LIC 82]. This problem is defined as follows: Given a collection $\mathcal{C} = (C_1, \ldots, C_m)$ of clauses over the set $X = \{x_1, \ldots, x_n\}$ of Boolean variables such that each clause C_j has at most three literals (and at least two), is there a truth assignment f satisfying $\mathcal{C}$? Moreover, the bipartite graph $BP = (L, R; E)$ is planar where $|L| = n$, $|R| = m$ and $[x_i, c_j] \in E$ if and only if the variable x_i (or $\overline{x_i}$) appears in the clause C_j.

THEOREM 10.7.– MIN WEIGHTED NODE COLORING *is* ***NP****-hard in triangle-free planar graphs with maximum degree 4. Moreover, if* ***P*****≠*****NP****, it is not approximable within ratio* $7/6 - \varepsilon$*, for any* $\epsilon > 0$*, in this class of graphs.*

Proof. Let $BP = (L, R; E)$ be the bipartite graph representing an instance $(X, \mathcal{C})$ of 3-SAT PLANAR where $L = \{x_1, \ldots, x_n\}$, $R = \{c_1, \ldots, c_m\}$. We construct an instance $I = (G, w)$ of MIN WEIGHTED NODE COLORING by using two gadgets:

– The gadgets clause $F(C_j)$ are given in Figure 10.1 for clause C_j of size 3 and in Figure 10.2 for clause C_j of size 2. The nodes c_j^k are those that will be linked to the rest of the graph.

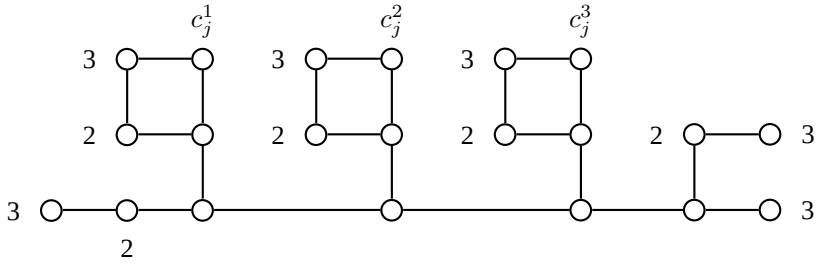


Figure 10.1. *Graph* $F(C_j)$ *representing a clause* C_j *of size 3*

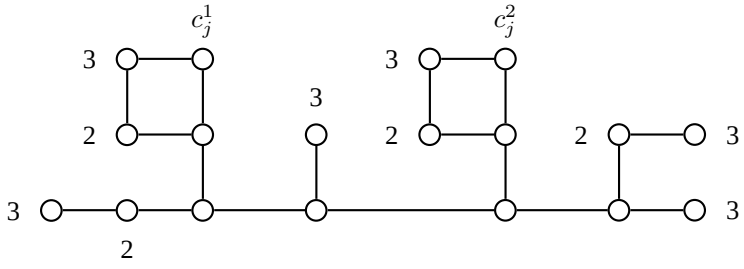


Figure 10.2. *Graph* $F(C_j)$ *representing a clause* C_j *of size 2*

– The gadgets variable $H(x_i)$ is given in Figure 10.3 for variable x_i. Assume that x_i appears p_1 times positively and p_2 times negatively in $(X, \mathcal{C})$, then in $H(x_i)$ there

are $2p = 2(p_1 + p_2)$ special nodes $x_i^k, \overline{x_i^k}$, $k = 1, \ldots, p$. These nodes form a path which meets node x_i^k, $\overline{x_i^k}$ alternatively.

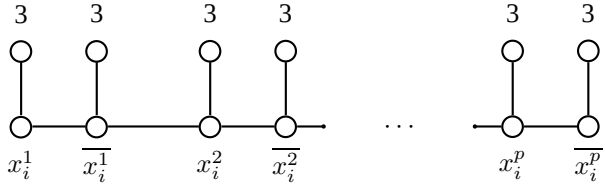


Figure 10.3. *Graph $H(x_i)$ representing variable x_i*

– The weights of nodes which are not given in Figures 10.1, 10.2 and 10.3 are 1.

– These gadgets are linked together by the following process. If variable x_i appears positively (resp. negatively) in clause C_j, we link one of the variables $\overline{x_i^k}$ (resp. x_i^k), with a different k for each C_j, to one of the three nodes c_j^k of gadget $F(C_j)$. This can be done in a way which preserves the planarity of the graph. Indeed, for each node v of degree $\delta(v)$ in the planar graph BP, let us call $e_v^1, \cdots, e_v^{\delta(v)}$ the endpoints on v of the edges adjacent to v considered in a circular order. Then, for each edge in BP which joins node x_i in endpoint $e_{x_i}^k$ to node C_j in endpoint $e_{c_j}^l$, we put an edge from x_i^k (if x_i appears negatively in C_j, $\overline{x_i^k}$ otherwise) to c_j^l.

Observe that G is triangle-free and planar with maximum degree 4. Moreover, we assume that G is not bipartite (otherwise, we add a disjoint cycle Γ with $|\Gamma| = 7$ and $\forall v \in V(\Gamma), w(v) = 1$).

It is then not difficult to check that $(X, \mathcal{C})$ is satisfiable iff $\mathrm{opt}(I) \leqslant 6$.

Let g be a truth assignment satisfying $(X, \mathcal{C})$. We set $S_1' = \{v : w(v) = 3\}$ and $S_2' = \{v : w(v) = 2\} \cup \{x_i^k : g(x_i) = 1\} \cup \{\overline{x_i^k} : g(x_i) = 0\}$. Since g satisfies the formula, we can color at least one node c_j^k with color 2 and then easily extend (S_1', S_2') to a coloring $\mathcal{S} = (S_1, S_2, S_3)$ of G with $S_i' \subseteq S_i$ for $i = 1, 2$. We have $w(S_1) = 3, w(S_2) = 2, w(S_3) = 1$ and then $w(\mathcal{S}) \leqslant 6$.

Conversely, let $\mathcal{S} = (S_1, \ldots, S_\ell)$ be a coloring of G with $w(\mathcal{S}) \leqslant 6$. Assume $w(S_1) \geqslant \ldots \geqslant w(S_\ell)$. We have $\ell \geqslant 3$ since G is not bipartite and $w(S_1) = 3$. We deduce $w(S_2) < 3$ (otherwise $w(\mathcal{S}) \geqslant 3 + 3 + 1$). Moreover, since each node of weight 2 is adjacent to a node of weight 3, we have $w(S_2) = 2$. For the same reasons

as previously, we deduce $\ell = 3$ and $w(S_3) = 1$. We claim that for any $j = 1, \ldots, m$, $S_2 \cap \{c_j^1, c_j^2, c_j^3\} \neq \{\}$ where c_j^1, c_j^2, c_j^3 are the nodes of $F(C_j)$ (with maybe $c_j^3 = \{\}$). Otherwise, we must have $\{c_j^1, c_j^2, c_j^3\} \subseteq S_3$ but in this case, we cannot have colored $F(C_j)$ with 3 colors. Thus, setting $g(x_i) = 1$ iff $x_i^k \in S_2$, we deduce that g is a truth assignment satisfying $(X, \mathcal{C})$. ∎

To conclude this section, let us note that MIN WEIGHTED NODE COLORING is polynomial in complements of triangle free graphs (i.e. graphs G for which $\alpha(G) \leqslant 2$). The algorithm is similar in spirit to that given in Garey and Johnson [GAR 75] for solving a two-processor problem with unit processing times and resource constraints: a matching in an auxiliary graph (where nodes are operations) gives the pairs of operations to be processed simultaneously (the algorithm and the correctness are given in [WER 02, DEM 07]). In addition, it is shown in Boudhar and Finke [BOU 00] that the problem becomes difficult as soon as there are release dates (even if all weights $w(v)$ equal 1).

10.4. Weighted node coloring in bipartite graphs

A graph is bipartite if it is 2-colorable. We proved in Corollary 10.3 that the problem is approximable within ratio 8/7 in bipartite graphs. Now, we mainly show that on the one hand, MIN WEIGHTED NODE COLORING is strongly **NP**-hard in bipartite graphs and, on the other hand, the bound of Corollary 10.3 is tight, since the problem in bipartite graphs is not approximable within ratio $8/7 - \varepsilon$, for any $\varepsilon > 0$. We also show that some more restrictive versions are also **NP**-hard, namely bipartite planar graphs and P_{21}-free bipartite graphs, i.e. bipartite graphs which do not contain induced chains of length 21 or more. We use a generic reduction from the precoloring extension node coloring problem (in short PREXT NODE COLORING). Then, using another reduction we improve this result to P_8-free bipartite graphs. Finally, we show that the problem becomes polynomial in P_5-free bipartite graphs.

10.4.1. *Hardness results*

PREXT NODE COLORING can be described as follows. Given a positive integer k, a graph $G = (V, E)$ and k subsets $V_1, \ldots, V_k$ of V that are pairwise disjoint, we want to decide if there exists a node coloring $\mathcal{S} = (S_1, \ldots, S_k)$ of G such that $V_i \subseteq S_i$, for all $i = 1, \ldots, k$.

Moreover, we restrict ourselves to some class of graphs $\mathcal{G}$: we assume that $\mathcal{G}$ is closed when we add a pending edge with a new node (i.e., if $G = (V, E) \in \mathcal{G}$ and $x \in V$, $y \notin V$, then $G + [x, y] \in \mathcal{G}$).

THEOREM 10.8.– *Let $\mathcal{G}$ be a class of graphs which is closed when we add a pending edge with a new node. If* PREXT NODE COLORING *is* ***NP****-hard for graphs in $\mathcal{G}$, then* MIN WEIGHTED NODE COLORING *is* ***NP****-hard for graphs in $\mathcal{G}$.*

Proof. Let $\mathcal{G}$ be such a class of graphs. We shall reduce PREXT NODE COLORING in $\mathcal{G}$ graphs to MIN WEIGHTED NODE COLORING in $\mathcal{G}$ graphs. Consider a graph $G = (V, E) \in \mathcal{G}$ and k pairwise disjoint subsets $V_1, \ldots, V_k$ of V. We build instance $I = (G', w)$ of MIN WEIGHTED NODE COLORING using several gadgets T_i, for $i = 1, \ldots, k$. The construction of T_i is given by induction as follows:

– T_1 is simply a root v_1 with weight $w(v_1) = 2^{k-1}$.

– Given $T_1, \ldots, T_{i-1}$, T_i is a tree with a root v_i of weight $w(v_i) = 2^{k-i}$ that we link to tree T_p via edge $[v_i, v_p]$ for each $p = 1, \ldots, i-1$.

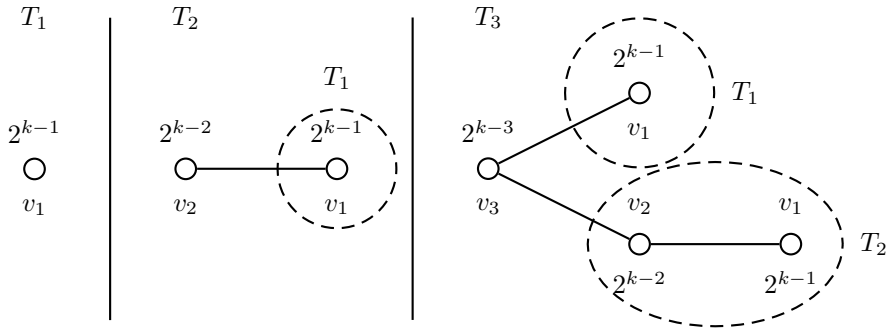


Figure 10.4. *Gadgets for T_1, T_2 and T_3*

Figure 10.4 illustrates the gadgets T_1, T_2, T_3. Now, $I = (G', w)$ where $G' = (V', E')$ is constructed in the following way:

– G' contains G.

– For all $i = 1, \ldots, k$, we replace each node $v \in V_i$ by a copy of the gadget T_i where we identify v with root v_i.

– For all $v \in V \setminus (\cup_{i=1}^{k} V_i)$ we set $w(v) = 1$.

Note that, by hypothesis, $G' \in \mathcal{G}$. We prove that the precoloring of G (given by $V_1, \ldots, V_k$) can be extended to a proper node coloring of G using at most k colors iff $\mathrm{opt}(I) \leqslant 2^k - 1$.

Let $\mathcal{S} = (S_1, \ldots, , S_k)$ with $V_i \subseteq S_i$ be a node coloring of G. We get $\mathcal{S}' = (S'_1, \ldots, , S'_k)$ where each stable S'_i is given by $S'_i = (S_i \setminus V_i) \cup \{v : \exists j \leqslant k,\ v \in$

T_j and $w(v) = 2^{k-i}\}$. It is easy to check that $\mathcal{S}'$ is a coloring of G' and $\text{opt}(I) \leqslant w(\mathcal{S}') = \sum_{i=1}^{k} 2^{k-i} = 2^k - 1$.

Conversely, let $\mathcal{S}' = (S'_1, \ldots, S'_\ell)$ with $w(S'_1) \geqslant \ldots \geqslant w(S'_\ell)$ be a weighted node coloring of G' with cost $w(\mathcal{S}') \leqslant 2^k - 1$. First, we prove by induction that $V'_i = \{v : \exists p \leqslant k,\ v \in T_p,\ w(v) = 2^{k-i}\}$ is a subset of S'_i, for all $i \leqslant k$. For $i = 1$, the result is true since otherwise we have $w(S'_1) = w(S'_2) = 2^{k-1}$ and then, $w(\mathcal{S}') \geqslant w(S'_1) + w(S'_2) = 2^k$. Now, assume that $V'_j \subseteq S'_j$ for $j < i$ and let us prove that $V'_i = \{v : \exists p \leqslant k,\ v \in T_p,\ w(v) = 2^{k-i}\} \subseteq S'_i$. By construction of gadget $T_j,\ j \geqslant i$, each node v of weight 2^{k-i} is adjacent to a node of weight 2^{k-p} for all $p < i$. Thus, $v \notin S'_p$. Now, if $V'_i \nsubseteq S'_i$, then $w(S'_i) = w(S'_{i+1}) = 2^{k-i}$ and we deduce that $w(\mathcal{S}') \geqslant w(S'_1) + \ldots + w(S'_{i+1}) = \sum_{j=1}^{i} 2^{k-j} + 2^{k-i} = 2^k$, which is a contradiction. Since $V'_i \neq \{\,\}$ for $i \leqslant k$, we deduce that $\ell \geqslant k$. Consequently, $\ell = k$, since $\forall v \in V', w(v) \geqslant 1$. Now, getting $\mathcal{S} = (S_1, \ldots, S_k)$ where $S_i = (S'_i \setminus V'_i) \cup V_i$ for each $i = 1, \ldots, k$, we obtain a node coloring of G. ∎

By using the results of Kratochvil [KRA 93] on the **NP**-completeness of PREXT NODE COLORING in bipartite planar graphs and P_{13}-free bipartite graphs, we deduce:

COROLLARY 10.7.– In bipartite planar graphs, MIN WEIGHTED NODE COLORING is strongly **NP**-hard and it is not $\frac{8}{7} - \varepsilon$-approximable for all $\varepsilon > 0$ unless **P**=**NP**.

Proof. PREXT NODE COLORING with $k = 3$ has been proved **NP**-complete in [KRA 93] for bipartite planar graphs. Since these graphs are closed when we add a pending edge with a new node, the result follows. Moreover, from the proof of Theorem 10.8 with $k = 3$, we deduce that it is **NP**-complete to distinguish whenever $\text{opt}(I) \leqslant 7$ and $\text{opt}(I) \geqslant 8$. ∎

As a consequence, MIN WEIGHTED NODE COLORING is also **NP**-hard if G is a comparability graph (i.e., a graph whose edges can be transitively oriented). The above proof also shows that the problem of MIN WEIGHTED NODE COLORING is **NP**-hard for a bipartite graph G as soon as the weights $w(v)$ can take three different values. Finally, note that in [WER 02, DEM 07], it is also proved that on the one hand, MIN WEIGHTED NODE COLORING remains **NP**-hard in bipartite graphs with a maximum degree at most 14 and, on the other hand, MIN WEIGHTED NODE COLORING is polynomial in bipartite graphs when the weights take at most two distinct values.

COROLLARY 10.8.– In P_{21}-free bipartite graphs, MIN WEIGHTED NODE COLORING is strongly **NP**-hard and it is not $\frac{32}{31} - \varepsilon$-approximable for all $\varepsilon > 0$ unless **P**=**NP**.

Proof. PREXT NODE COLORING with $k = 5$ has been proved **NP**-complete in [KRA 93] for P_{13}-free bipartite graphs. When, we add gadgets T_i with $i \leqslant 5$, G' becomes P_{21}-free bipartite graphs. Moreover, from the proof of Theorem 10.8 with

$k = 5$, we deduce that it is **NP**-complete to distinguish whenever $\mathrm{opt}(I) \leqslant 31$ and $\mathrm{opt}(I) \geqslant 32$. ∎

In Hujter and Tuza [HUJ 96], it is shown that PREXT NODE COLORING is **NP**-complete in P_6-free bipartite chordal graphs for unbounded k (a bipartite graph is chordal if the induced cycles of length at least 5 have a chord). Unfortunately, we cannot use this result in Theorem 10.8 since the resulting graph has an induced chain with arbitrarily large length. However, we can adapt their reduction to our problem.

THEOREM 10.9.– MIN WEIGHTED NODE COLORING *is **NP**-hard in P_8-free bipartite graphs.*

Proof. We shall reduce 3-SAT-3, proved to be **NP**-complete in Papadimitriou [PAP 94], to our problem. Given a collection $\mathcal{C} = (C_1, \ldots, C_m)$ of clauses over the set $X = \{x_1, \ldots, x_n\}$ of Boolean variables such that each clause C_j has at most three literals and each variable appears twice positively and once negatively, we construct an instance $I = (BP, w)$ in the following way:

– we start from $BP_1 = (L_1, R_1; E_1)$, which is a complete bipartite graph $K_{n,m}$ where $L_1 = \{x_1, \ldots, x_n\}$ and $R_1 = \{c_1, \ldots, c_m\}$. Moreover, each node of BP_1 has weight 1;

– there is also another bipartite graph BP_2 isomorphic to $K_{2n,2n}$ where a perfect matching has been deleted. More formally, $BP_2 = (L_2, R_2; E_2)$ where $L_2 = \{l_1, \ldots, l_{2n}\}$, $R_2 = \{r_1, \ldots, r_{2n}\}$ and $[l_i, r_j] \in E_2$ if and only if $i \neq j$. Finally, $w(l_i) = w(r_i) = 2^{2n-i}$ for $i = 1, \ldots, 2n$. Indeed, sets $\{l_{2i-1}, r_{2i-1}\}$ and $\{l_{2i}, r_{2i}\}$ will correspond to variable x_i and $\overline{x_i}$ respectively;

– between BP_1 and BP_2 there is a set E_3 of edges. $[x_i, r_j] \notin E_3$ iff $j = 2i - 1$ or $j = 2i$ and $[l_i, c_j] \notin E_3$ iff $i = 2k - 1$ and x_k is in C_j or $i = 2k$ and $\overline{x_k}$ is in C_j.

Figure 10.5 illustrates the construction of the complement of BP with the clause $c_m = \overline{x_1} \vee x_2 \vee \overline{x_n}$.

Let us show that BP is P_8-free. We represent in Figure 10.6 the possible subgraphs on BP_1 (configuration A_1, A_2 and A_3) and on BP_2 (configuration B_1 to B_9) induced by a chain on BP. In configurations A_3 and B_9, the number of nodes is arbitrary. Note that the upper line may correspond either to L_1 or R_1 for the left part (and L_2 or R_2 for the right part). Now we look at the possible ways to link a configuration A_i to a configuration B_j to obtain a chain of length (at least) 8:

– if we choose A_1, we easily see that it is impossible;

– if we choose A_2, the only way to have a chain of length at least 8 is to choose B_8 and link a node of A_2 to a node of B_8. In this case, we can see that the upper line corresponds to R_1 (left part) and L_2 (right part), and that there is a clause which contains a variable and its negation;

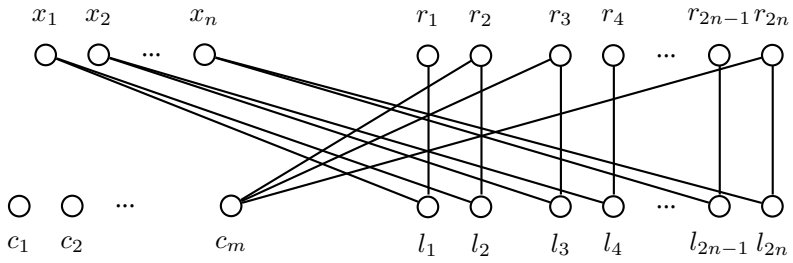


Figure 10.5. *Complement of graph BP with the clause $c_m = \overline{x_1} \vee x_2 \vee \overline{x_n}$*

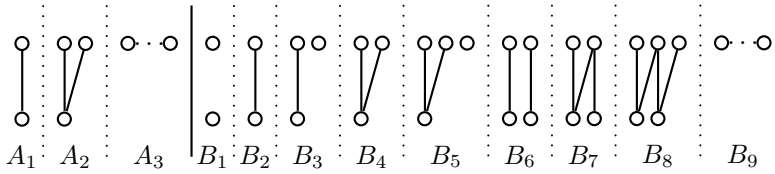


Figure 10.6. *Subgraphs on BP_1 and BP_2 induced by a chain*

– if we choose A_3, the only possibility to have a chain of length at least 8 is to choose B_9. But in this case, the chain simply alternates a node of R_1 and a node of L_2. Then, at least one node of L_2 is not linked to at least 3 nodes of R_1, i.e. a literal appears in at least 3 clauses.

We claim that $(X, \mathcal{C})$ is satisfiable if and only if $\mathrm{opt}(I) \leqslant 2^{2n} - 1$.

Let g be a truth assignment satisfying $(X, \mathcal{C})$. We build inductively the colors. $S_0 = \{\}$ and for $i = 1, \ldots, n$, $S_{2i-1} = \{l_{2i-1}, r_{2i-1}\} \cup \{c_j : c_j \notin S_p, p < 2i-1, g(x_i) = 1$ and x_i is in $C_j\}$, $S_{2i} = \{l_{2i}, r_{2i}\} \cup \{c_j : c_j \notin S_p, p < 2i, g(x_i) = 0$ and $\overline{x_i}$ is in $C_j\}$. Finally, if $g(x_i) = 1$ then we add x_i to S_{2i}; otherwise, we add x_i to S_{2i-1}. We can easily see that $\mathcal{S} = (S_1, \ldots, S_{2n})$ is a node coloring of BP with $w(\mathcal{S}) = 2^{2n} - 1$.

Conversely, let $\mathcal{S} = (S_1, \ldots, S_\ell)$ be a node coloring of BP with $w(\mathcal{S}) = 2^{2n} - 1$. An inductive proof on i shows that $\{l_i, r_i\} \subseteq S_i$ (otherwise, we have $w(\mathcal{S}) \geqslant 2^{2n}$);

consequently, $\ell = 2n$. Thus, by setting $g(x_i) = 1$ if $x_i \in S_{2i}$ and $g(x_i) = 0$ if $x_i \in S_{2i-1}$, we obtain a truth assignment satisfying $(X, \mathcal{C})$. ∎

10.4.2. P_5-*free bipartite graphs*

We now prove that MIN WEIGHTED NODE COLORING is polynomial for P_5-free bipartite graphs, i.e., without induced chain on 5 nodes. Notice that in general P_5-free graphs, MIN WEIGHTED NODE COLORING is **NP**-hard since on the one hand, the split graphs are P_5-free and on the other hand, MIN WEIGHTED NODE COLORING is **NP**-hard in split graphs (see section 10.5). There are several characterizations of P_5-free bipartite graphs (see, for example, Hammer *et al.* [HAM 90], Chung *et al.* [CHU 90] and Hujter and Tuza [HUJ 93]). In particular, BP is a P_5-free bipartite graph iff BP is bipartite and each connected component of BP is $2K_2$-free, i.e., its complement is C_4-free.

LEMMA 10.3.– *In P_5-free bipartite graph, any optimal weighted node coloring uses at most 3 colors.*

Proof. Let $BP = (L, R; E)$ be a P_5-free bipartite graph with connected components $BP_1, \ldots, BP_p$. Assume the reverse and let us consider an optimal solution $\mathcal{S}^* = (S_1^*, \ldots, S_\ell^*)$ with $\ell \geqslant 4$ and $w(S_1^*) \geqslant \ldots \geqslant w(S_\ell^*)$. Observe that, without loss of generality, we can assume that there exists a connected component BP_{k_0} colored with ℓ colors and any connected component BP_i using j colors is colored with colors $1, \ldots, j$. Moreover, we also suppose that in any connected component BP_j, each node colored with color $i \geqslant 2$ is adjacent to nodes with colors $1, \ldots, i-1$ (by applying a greedy rule on $\mathcal{S}^*$, see Theorem 10.1).

We claim that there exists $1 \leqslant i < j \leqslant \ell$ such that $S_k^* \cap L \neq \{\}$ and $S_k^* \cap R \neq \{\}$ for $k = i, j$.

Otherwise, since $\ell \geqslant 4$, we must have $S_{i_0}^* \subseteq L$ (resp., $S_{i_0}^* \subseteq R$) and $S_{j_0}^* \subseteq L$ (resp., $S_{j_0}^* \subseteq R$) for some $i_0 < j_0$. In this case, by merging $S_{i_0}^*$ with $S_{j_0}^*$, we obtain a better node coloring than $\mathcal{S}^*$, which is a contradiction.

So, consider the connected component BP_{k_0} and let $l_j \in S_j^* \cap L$ and $r_j \in S_j^* \cap L$ be two nodes of BP_{k_0}. From this claim, we deduce that there exist 2 other nodes l_i, r_i of BP_{k_0} such that $l_i \in S_i^* \cap L$, $r_i \in S_i^* \cap R$ and $[l_i, r_j] \in E$, $[l_j, r_i] \in E$. Since BP is bipartite, these 2 edges are independent, which is a contradiction with the characterization of P_5-free bipartite graphs. ∎

Let $BP_1, \ldots, BP_p$ be the connected components of BP where $BP_i = (L_i, R_i; E_i)$. Let $\mathcal{S}^* = (S_1^*, S_2^*, S_3^*)$ (with maybe some $S_i^* = \{\}$) be an optimal solution with $w(S_1^*) \geqslant w(S_2^*) \geqslant w(S_3^*)$ and denote by $\mathcal{S}_i^* = (S_1^{*,i}, S_2^{*,i}, S_3^{*,i})$ the restriction of $\mathcal{S}^*$ to the subgraph BP_i. Note that we may assume $w(S_1^{*,i}) \geqslant w(S_2^{*,i}) \geqslant$

$w(S_3^{*,i})$ (otherwise, we can flip the color without increasing the weight). Moreover, we have:

LEMMA 10.4.– *We can always assume that one of these situations occurs, for any* $i = 1, \ldots, p$*:*

(i) $S_1^{*,i} = L_i$ *(resp.,* $S_1^{*,i} = R_i$*),* $S_2^{*,i} = R_i$ *(resp.,* $S_2^{*,i} = L_i$*) and* $S_3^{*,i} = \{\,\}$.

(ii) $S_1^{*,i} \cap L_i \neq \{\,\}$ *and* $S_1^{*,i} \cap R_i \neq \{\,\}$, $S_2^{*,i} \subset R_i$ *(resp.,* $S_2^{*,i} \subset L_i$*) and* $S_3^{*,i} \subset L_i$ *(resp.,* $S_3^{*,i} \subset R_i$*).*

Proof. Let $BP = (L, R; E)$ be a P_5-free bipartite graph with connected components $BP_1, \ldots, BP_p$. Assume that $S_1^{*,i} \cap L_i = \{\,\}$ or $S_1^{*,i} \cap R_i = \{\,\}$. In this case, it is clear that we are in the first item (i) (since we have assumed that $w(S_1^{*,i}) \geqslant w(S_2^{*,i}) \geqslant w(S_3^{*,i})$). Now, suppose that $S_1^{*,i} \cap L_i \neq \{\,\}$ and $S_1^{*,i} \cap R_i \neq \{\,\}$; from the proof of Lemma 10.3, the result follows. ∎

The algorithm calculating an optimal solution is described in the following way:

P_5-FREEBIPARTITECOLOR

1) For all $k_1, k_2 \in \{w(v) : v \in V\}$, $k_1 \geqslant k_2$, do: for all connected component $BP_i = (L_i, R_i; E_i)$, $i = 1, \ldots, p$, do

a) If $L_i \cup R_i \setminus (L_i' \cup R_i')$ is an independent set where $L_i' = \{v \in L_i : w(v) \leqslant k_1\}$ and $R_i' = \{v \in R_i : w(v) \leqslant k_2\}$ then set $S_{2,i}^{k_1,k_2} = L_i'$, $S_{3,i}^{k_1,k_2} = R_i'$ and $S_{1,i}^{k_1,k_2} = L_i \cup R_i \setminus (L_i' \cup R_i')$;

b) Otherwise, if $L_i \cup R_i \setminus (L_i' \cup R_i')$ is an independent set where $L_i' = \{v \in L_i : w(v) \leqslant k_2\}$ and $R_i' = \{v \in R_i : w(v) \leqslant k_1\}$ then set $S_{2,i}^{k_1,k_2} = R_i'$, $S_{3,i}^{k_1,k_2} = L_i'$ and $S_{1,i}^{k_1,k_2} = L_i \cup R_i \setminus (L_i' \cup R_i')$;

c) Otherwise go to step 1;

d) Set $S_j^{k_1,k_2} = \cup_{i=1}^{p} S_{j,i}^{k_1,k_2}$ for $j = 1, 2, 3$ and $\mathcal{S}^{k_1,k_2} = (S_1^{k_1,k_2}, S_2^{k_1,k_2}, S_3^{k_1,k_2})$ (with possibly $S_1^{k_1,k_2} = \{\,\}$);

2) Output $\mathcal{S} = argmin\{w(\mathcal{S}^{k_1,k_2}) : k_2 \leqslant k_1\}$.

This algorithm has a complexity $O(n|w|^2)$ where $|w| = |\{w(v) : v \in V\}|$. By applying a dichotomy technique on k_2, we can improve it to $O(n|w|log|w|)$. Note that this algorithm also calculates the best node 2-coloring among the colorings using at most 2 colors (when $k_1 = w_{max}$).

THEOREM 10.10.– MIN WEIGHTED NODE COLORING *is polynomial in* P_5*-free bipartite graphs.*

Proof. Let $\mathcal{S}^* = (S_1^*, S_2^*, S_3^*)$ (with possibly $S_1^* = \{\,\}$) be an optimal solution satisfying Lemmas 10.3 and 10.4. We assume that $w(S_2^*) \geqslant w(S_3^*)$ and if $\mathcal{S}^*$

is a node 3-coloring, then we have $w(S_1^*) = w_{max}$; otherwise $w(S_1^*) = 0$. Let $k_1 = w(S_2^*)$ and $k_2 = w(S_3^*)$; consider the step of algorithm corresponding to k_1, k_2. If $\mathcal{S}^*$ is a node 2-coloring, then the result is true. So, assume that $S_1^* \neq \{\}$; by construction, P_5-FREEBIPARTITECOLOR calculates a feasible solution $\mathcal{S}^{k_1,k_2}$ with $w(S_1^{k_1,k_2}) \leqslant w_{max}$, $w(S_2^{k_1,k_2}) \leqslant k_1$ and $w(S_3^{k_1,k_2}) \leqslant k_2$. Thus, we deduce the expected result. ∎

Let us finish this section by mentioning that if $I = (G, w)$ where $G = (V, E)$ is the complement of a bipartite graph, then MIN WEIGHTED NODE COLORING is polynomial. Indeed, this is a particular case of the result for complements of triangle-free graphs.

10.5. Split graphs

In this section, we examine the situation of *split* graphs, i.e., graphs $G = (K_1, V_2; E)$ in which the node set V can be partitioned into a stable set V_2 and a clique K_1. These graphs can be considered as intermediate between bipartite graphs and complements of bipartite graphs. We mainly prove that MIN WEIGHTED NODE COLORING is strongly **NP**-hard and admits a PTAS in this class of graphs.

10.5.1. *Complexity result*

The following theorem, characterizes the complexity of MIN WEIGHTED NODE COLORING in split graphs.

THEOREM 10.11.– MIN WEIGHTED NODE COLORING *is strongly **NP**-hard for split graphs.*

Proof. We shall use a reduction from the following problem called MIN SET COVER: given a collection $\mathcal{C} = \{C_j : j \in J\}$ of subsets of a set X and a positive integer q ($q \leqslant |J|$), does there exist $J' \subseteq J$ such that the subcollection $\mathcal{C}' = \{C_j : j \in J'\}$ satisfies $|J'| \leqslant q$ and $\cup_{j \in J'} C_j = X$? MIN SET COVER was shown to be **NP**-complete in [KAR 72].

We shall transform an instance $(\{C_j : j \in J\}, X)$ of MIN SET COVER into an instance $I = (G, w)$ of MIN WEIGHTED NODE COLORING, where $G = (V, E)$ is a split graph. Let us construct the graph G as follows. Each element x of X becomes a node v_x of the stable set V_2; each subset C_i in $\mathcal{C}$ corresponds to a node c_i of the clique K_1 of G. The set $N(c_i)$ of neighbors of node c_i is given by: $N(c_i) = \{v_x : x \in X \setminus \{v_x : x \in C_i\}$. The weights are given by $w(c_j) = |J|$, for $j \in J$, and $w(v_x) = |J| + 1$, for $v_x \in V_2$.

Then there exists a set cover $\mathcal{C}' = \{C_j : j \in J'\}$ with $\cup_{j\in J'} C_j = X$ and $|J'| \leqslant q$ if and only if there exists in G a k-coloring $\mathcal{S} = (S_1, \ldots, S_k)$ with $w(\mathcal{S}) \leqslant |J|^2 + q$. This can be seen as follows. Assume that we have a set cover $\mathcal{C}' = \{C_j : j \in J'\}$ with $|J'| \leqslant q$ and for each $x \in X$ let $j(x) = \min\{j \in J' : x \in C_j\}$; for $j = 1, \ldots, |J|$ let $S_i = \{c_i\} \cup \{v_x : j(x) = i\}$. This gives a $|J|$-coloring $\mathcal{S}$ of G: since $\mathcal{C}'$ is a set cover, every node $v_x \in V_2$ is introduced into a set S_i. Furthermore, $x \in C_j$ implies that nodes v_x and c_j of G are not linked by an edge and hence each S_i is a stable set. Now $w(\mathcal{S}) = \sum_{j=1}^{|J|} w(S_j)$ and we have $w(S_j) = |J|$ if $j \notin J'$ and $w(S_j) \leqslant |J| + 1$ if $j \in J'$. Hence $w(\mathcal{S}) \leqslant |J|^2 + q$. Conversely, let $\mathcal{S} = (S_1, \ldots, S_k)$ be a k-coloring of G with $w(\mathcal{S}) \leqslant |J|^2 + q$. We have $q \leqslant |J|$ and hence $q < |J| + 1$. Since G has a clique K_1 on $|J|$ nodes c_j satisfying $w(c_j) = |J|$ and since all nodes $v_x \in V_2$ have $w(v_x) = |J| + 1$, the coloring $\mathcal{S}$ is a $|J|$-coloring. This means that we can assume that $c_j \in S_j$ for $j = 1, \ldots, |J|$. Since $w(\mathcal{S}) \leqslant |J|^2 + q$, there are at most q subsets S_j in $\mathcal{S}$ with $w(S_j) = |J| + 1$. For every node $v_x \in V_2$ there exists a stable set $S_{i(x)}$ which contains v_x. This implies that for each $v_x \in V_2$, there exists a subset $C_{i(x)}$ in $\mathcal{C}$ which contains x. Let $J' = \{j : j = i(x) \text{ for some } v_x \in V_2\}$; it follows that $\cup_{j\in J'} C_j = X$. Furthermore, since $w(S_{i(x)}) = |J| + 1$ for any v_x and since $w(\mathcal{S}) \leqslant |J|^2 + q$, we have $|J'| \leqslant q$. Hence, $\mathcal{C}' = \{C_j : j \in J'\}$ is the required set cover. ∎

The proof of Theorem 10.11 shows that the problem is **NP**-hard even if the weights can take only two values. As observed in [BOU 00], the problem is easy if all weights are 1; it amounts to finding the chromatic number of a split graph $G = (K_1, V_2; E)$. We have $\chi(G) = |K_1|$ where K_1 is the maximum clique. More generally, when all weights are 1, the problem is just graph coloring, which is polynomial for perfect graphs and some other classes (see for instance Brandstädt *et al.* [BRA 99]).

It also follows from the proof of Theorem 10.11 that MIN WEIGHTED NODE COLORING is **NP**-hard for chordal graphs (i.e., graphs where every cycle of length at least four has a chord), since a split graph is a chordal graph.

10.5.2. *Approximation result*

From Theorem 10.11 we deduce that there is no fully polynomial time approximation scheme in this class of graphs. Here, we propose a polynomial time approximation scheme using structural properties of optimal solutions. An immediate observation of split graphs is that any optimal node coloring $\mathcal{S}^* = (S_1^*, \ldots, S_\ell^*)$ satisfies $|K_1| \leqslant \ell \leqslant |K_1| + 1$ and any color S_i^* is a subset of V_2 with possibly one node of K_1. In particular, for any optimal node coloring $\mathcal{S}^* = (S_1^*, \ldots, S_\ell^*)$, there exists at most one index $i(\mathcal{S}^*)$ such that $S^*_{i(\mathcal{S}^*)} \cap K_1 = \{\,\}$.

LEMMA 10.5.– *There is an optimal weighted node coloring $\mathcal{S}^* = (S_1^*, \ldots, S_\ell^*)$ with $w(S_1^*) \geqslant \ldots \geqslant w(S_\ell^*)$ and an index $i_0 \leqslant \ell + 1$ such that:*

– $\forall j < i_0$ $S_j^* = \{v_j\} \cup \{v \in V_2 : v \notin \cup_{k=1}^{j-1} S_k^*$ *and* $[v, v_j] \notin E\}$ *for some* $v_j \in K_1$.

– $S_{i_0}^* = V_2 \setminus (S_1^* \cup \ldots \cup S_{i_0-1}^*)$.

– $\forall j > i_0$ $S_j^* = \{v_j\}$ *for some* $v_j \in K_1$.

Proof. Let $G = (K_1, V_2; E)$ be a split graph and let $\mathcal{S}^* = (S_1^*, \ldots, S_\ell^*)$ with $w(S_1^*) \geqslant \ldots \geqslant w(S_\ell^*)$ be an optimal weighted node coloring of G. If $\ell = n_1$ (we recall that $n_1 = |K_1|$), then we set $i_0 = \ell+1$, otherwise let i_0 be the unique i such that $S_i^* \cap K_1 = \{\}$. We build set $S_i^{*'}$ in the following way:

– for $i = 1, \ldots, i_0 - 1$, $S_i^{*'} = \{v_i\} \cup \{v \in V_2 : v \notin \cup_{k=1}^{i-1} S_k^{*'}$ and $[v, v_i] \notin E\}$ where we assume that $S_i^* \cap K_1 = \{v_i\}$;

– $S_{i_0}^{*'} = V_2 \setminus (S_1^{*'} \cup \ldots \cup S_{i_0-1}^{*'})$;

– for $i = i_0 + 1, \ldots, \ell$, $S_i^{*'} = S_i^* \cap K_1$.

Thus, when $i_0 = \ell+1$, the sets resulting from the second and third steps are empty. Let us prove that:

$$\forall i = 1, \ldots, \ell,\ w(S_i^{*'}) \leqslant w(S_i^*) \qquad [10.7]$$

Since $w(S_1^*) \geqslant \ldots \geqslant w(S_\ell^*)$, we have $w(S_i^*) = \max\{w(v) : v \in K_1 \cup V_2 \setminus (S_1^* \cup \ldots \cup S_{i-1}^*)\}$. Moreover, by construction, $\cup_{j=1}^{i-1} S_j^* \subseteq \cup_{j=1}^{i-1} S_j^{*'}$. Thus, the result follows.

Using inequality [10.7], we deduce that node coloring $\mathcal{S}^{*'} = (S_1^{*'}, \ldots, S_\ell^{*'})$ has a cost $w(\mathcal{S}^{*'}) \leqslant \sum_{i=1}^{\ell} w(S_i^*) = \mathrm{opt}(I)$ and then $\mathcal{S}^{*'}$ is an optimal weighted node coloring satisfying Lemma 10.5. ∎

SPLITNODECOLOR$_k$

1) For all subset $K_1' \subseteq K_1$ with $|K_1'| = p \leqslant k$ do: for all bijections f: $\{1, \ldots, p\} \longmapsto K_1'$ do

a) For $i = 1$ to p set $S_i^{K_1', f} = \{f(i)\} \cup \{v \in V_2 : v \notin \cup_{k=1}^{i-1} S_k^{K_1', f}$ and $[v, f(i)] \notin E\}$;

b) Set $S_{p+1,f}^{K_1'} = V_2 \setminus (S_1^{K_1', f} \cup \ldots \cup S_p^{K_1', f})$;

c) For $i = p+2$ to n_1+1 (assume $K_1 \setminus K_1' = \{v_{p+2}, \ldots, v_{n_1+1}\}$) set $S_i^{K_1', f} = \{v_i\}$;

d) Set $\mathcal{S}^{K_1', f} = (S_1^{K_1', f}, \ldots, S_{n_1+1}^{K_1', f})$;

2 Output $\mathcal{S} = argmin\{w(\mathcal{S}^{K_1', f})\}$.

This algorithm has a time complexity $O(k!n^{k+1})$.

THEOREM 10.12.– *For all $\varepsilon > 0$, $\texttt{SPLITNODECOLOR}_{\lceil \frac{1}{\varepsilon} \rceil}$ produces a $(1+\varepsilon)$-approximation for* MIN WEIGHTED NODE COLORING *in split graphs.*

Proof. Let $G = (K_1, V_2; E)$ be a split graph and let $\mathcal{S}^* = (S_1^*, \ldots, S_\ell^*)$ with $w(S_1^*) \geqslant \ldots \geqslant w(S_\ell^*)$ be an optimal weighted node coloring of G satisfying Lemma 10.5. Let $k = \lceil \frac{1}{\varepsilon} \rceil$. If $i_0 \leqslant k$, then by construction the solution $\mathcal{S}$ returned by $\texttt{SPLITNODECOLOR}_k$ is optimal. So, assume that $i_0 > k$ and let $K_1^{*'} = (\cup_{j=1}^{k} S_j^*) \setminus V_2$. Obviously, $|K_1^{*'}| = k$ and let $f^*(i) = S_i^* \cap K_1$ for $i = 1, \ldots, k$.

Let us examine the solution $\mathcal{S}^{K_1^{*'}, f^*}$ corresponding to the step $K_1' = K_1^{*'}$ and $f = f^*$ of $\texttt{SPLITNODECOLOR}_k$. By construction, we have:

$$\forall i = 1, \ldots, k,\ S_i^{K_1^{*'}, f^*} = S_i^* \qquad [10.8]$$

Moreover, since $K_1 \setminus K_1^{*'} \subseteq S_{k+1}^* \cup \ldots \cup S_\ell^*$ and $K_1 \setminus K_1^{*'}$ is a clique, we obtain:

$$\sum_{j=k+2}^{n_1+1} w(S_i^{K_1^{*'}, f^*}) \leqslant \sum_{j=k+1}^{\ell} w(S_i^*) \qquad [10.9]$$

Thus, combining [10.8] and [10.9], we deduce:

$$w(\mathcal{S}^{K_1^{*'}, f^*}) - w(S_{k+1}^{K_1^{*'}, f^*}) \leqslant \mathrm{opt}(I) \qquad [10.10]$$

Moreover, by construction, $w(S_{k+1}^{K_1^{*'}, f^*}) \leqslant w(S_k^*) \leqslant \ldots \leqslant w(S_1^*)$ and then:

$$w(S_{k+1}^{K_1^{*'}, f^*}) \leqslant \frac{1}{k} \times \mathrm{opt}(I) \qquad [10.11]$$

Finally, using these two last inequalities with $\frac{1}{k} \leqslant \varepsilon$, we obtain the expected result. ∎

10.6. Cographs

The case of P_4-free graphs (or, equivalently, graphs containing no induced chain P_4 on four nodes) has to be mentioned. These graphs, also called cographs, are a subclass of the perfectly ordered graphs introduced in Chvátal [CHV 84]. For the perfectly ordered graphs, an order θ on the node set V can be defined in such a way that for any induced subgraph G' of the original graph G the *greedy sequential algorithm* (GSC) based on the order θ' induced by θ on the nodes of G' gives a minimum coloring of G' (i.e., a coloring in exactly $\chi(G')$ colors). Here the GSC algorithm based on an order θ consists of examining consecutively the nodes as they occur in θ and coloring them with the smallest possible color. As observed in [WER 90], a graph G is a cograph if and only if for all induced subgraphs G' of G the GSC based on any order θ gives a coloring of G' in $\chi(G')$ colors.

LEMMA 10.6.– *If $I = (G, w)$ where $G = (V, E)$ is a cograph, then all k-colorings $\mathcal{S} = (S_1, \ldots, S_k)$ minimizing $w(\mathcal{S})$ satisfy $k = \chi(G)$.*

Proof. Assume that there is a k'-coloring $\mathcal{S}^* = (S_1^*, \ldots, S_{k'}^*)$ which is an optimal solution of MIN WEIGHTED NODE COLORING on $I = (G, w)$, where G is a cograph and for which $k' > \chi(G)$. As before, we assume that $w(S_1^*) \geqslant w(S_2^*) \geqslant \ldots \geqslant w(S_{k'}^*)$. We can order the nodes of G by taking consecutively the nodes of S_1^*, those of S_2^* and so on. Using the resulting order we can apply the GSC algorithm which will produce a k-coloring with $k = \chi(G)$. Each node v with color j will have after recoloring a color $i \leqslant j$.

Let us examine what happens when some node v initially with color j gets a color $i < j$. Let $\mathcal{S} = (S_1, \ldots, S_k)$ be the resulting coloring; we have $S_r = S_r^*$ for $r \neq i, j$. Also, $S_j = S_j^* \setminus \{v\}$, $S_i = S_i^* \cup \{v\}$. So $w(S_j) \leqslant w(S_j^*)$ and since $w(v) \leqslant w(S_j^*) \leqslant w(S_i^*)$, we also have $w(S_i) \leqslant w(S_i^*)$ and hence $w(S_r) \leqslant w(S_r^*)$ for $r = 1, \ldots, k$. So, $w(\mathcal{S}) \leqslant \mathrm{opt}(I)$.

Now, if we call $\mathcal{S}' = (S_1', \ldots, S_k')$ the k-coloring resulting from the GSC algorithm with the order described above, we will have $w(\mathcal{S}') \leqslant w(\mathcal{S}^*)$. But now $S_{k+1}' = \{\,\}$, so $0 = w(S_{k+1}') < w(S_{k+1}^*)$ because all weights are positive. Hence, $w(\mathcal{S}') < w(\mathcal{S}^*)$, which is a contradiction. ∎

We can now show that there is a polynomial algorithm which constructs a k-coloring $\mathcal{S}$ which is an optimal solution of MIN WEIGHTED NODE COLORING; such a result can be expected from graphs like cographs for which several generally difficult coloring problems are easier (see Jansen and Scheffler [JAN 97] for example).

THEOREM 10.13.– *Let $G = (V, E, w)$ be a a weighted cograph. Then the coloring $\mathcal{S}$ constructed by the GSC algorithm based on any order θ where $u < v$ (u before v*

in θ) implies that $w(u) \geqslant w(v)$ is an optimal coloring of MIN WEIGHTED NODE COLORING.

Proof. Let $t_1 > t_2 > \ldots > t_r$ be the values taken by the weights $w(v)$ in G. Every k-coloring $\mathcal{S} = (S_1, \ldots, S_k)$ of G with $k = \chi(G)$ and $w(S_1) \geqslant w(S_2) \geqslant \ldots \geqslant w(S_k)$ satisfies:

$$w(S_i) \geqslant \max\{t_s : \omega(G(s)) \geqslant i\}.$$

where $\omega(H)$ denotes the maximum size of a clique in a graph H and $G(s)$ is the subgraph generated by all nodes v with $w(v) \geqslant t_s$. Indeed, any such k-coloring will have the first $\omega(G(1))$ sets S_i with $w(S_i) = t_1$; also the first $\omega(G(2))$ sets S_i will have $w(S_i) \geqslant t_2$ and generally the first $\omega(G(s))$ sets S_i will have $w(S_i) \geqslant t_s$. Now consider the k-coloring $\mathcal{S}' = (S'_1, \ldots, S'_k)$ obtained by applying the GSC algorithm based on any order θ with non-increasing weights. Let $p(s)$ be the largest color given to a node v with $w(v) = t_s$; let v_0 be such a node. Since cographs are perfectly ordered graphs (with respect to our order θ), it follows by considering the subgraph G' of G generated by v_0 and all its predecessors in θ that there is in G' a clique $K \ni v_0$ with $K \cap S'_i \neq \{\}$ for $i = 1, \ldots, p(s)$. Clearly, $w(v) \geqslant t_s$ for every node $v \in K$. This means that $p(s) \leqslant \omega(G(s))$. Now $\mathcal{S}'$ (where we have $w(S'_1) \geqslant \ldots \geqslant w(S'_k)$) satisfies $w(S'_i) = \max\{t_s : p(s) \geqslant i\}$; but then $w(S'_i) \leqslant \max\{t_s : \omega(G(s)) \geqslant i\}$ and we have $w(S'_i) = \max\{t_s : \omega(G(s)) \geqslant i\}$ for $i = 1, \ldots, k$. So $\mathcal{S}'$ is an optimum coloring of MIN WEIGHTED NODE COLORING. ∎

The above proof shows in fact that if we are given a perfectly ordered graph G and if the order θ of non-increasing weights is such that the GSC algorithm gives a minimum coloring (i.e., a k-coloring with $k = \chi(G)$), then we can find an optimum coloring of MIN WEIGHTED NODE COLORING. For cographs, this condition was satisfied since any order θ could be chosen to construct a minimum coloring.

10.7. Interval graphs

Interval graphs are frequently used in practice, in particular when dealing with scheduling problems. An interval graph is the intersection graph of a set of intervals. A well known characterization of interval graphs is the following: G is an interval graph iff G is a chordal graph and G is a co-comparability graph. Although the coloring problem is polynomial in interval graphs (since an interval graph is a chordal graph), in this section, we prove that the MIN WEIGHTED NODE COLORING problem is strongly **NP**-hard in interval graphs. The status of MIN WEIGHTED NODE COLORING in interval graphs was an open question raised in [GUA 97]. From the approximation viewpoint, a 2-approximate algorithm is provided in [PEM 05].

Note that the result given in Theorem 10.11 does not allow us to conclude that MIN WEIGHTED NODE COLORING is **NP**-hard in interval graphs since this class is strictly included in the class of chordal graphs.

THEOREM 10.14.– MIN WEIGHTED NODE COLORING *is strongly* ***NP****-hard in interval graphs.*

Proof. We reduce the Circular Arc Coloring problem to our problem. A circular arc graph is the intersection graph of the arcs of a circle. Garey *et al.* [GAR 80] proved that the Circular Arc Coloring problem, i.e., the problem of finding a minimum size coloring in circular arc graphs, is **NP**-hard. Let G be the intersection graph of the n-tuple of circular arcs $\mathcal{A} = (A_j)_{j\in\{1,\cdots,n\}}$ and let $k \in \{1,\cdots,n\}$. Assume, without loss of generality, that the intervals A_j are open. We transform this instance of Circular Arc Coloring in an instance $I' = (G', w)$ of MIN WEIGHTED NODE COLORING as follows. Let a be any point on the circle, and $J_0 = \{j : a \in A_j\}$. For simplicity, assume wlog. that point a belongs to some arcs and that $J_0 = \{1,\cdots,j_0\}$, for some $j_0 \geqslant 1$. For any $j \leqslant j_0$, we split interval $A_j = (c_j, d_j)$ in $A'_j = (c_j, a)$ and $A''_j = (a, d_j)$. For $j > j_0$, we define $A'_j = A_j$. Let $\mathcal{A}'$ be the $(n + j_0)-$tuple of intervals $(A'_j)_{j\in\{1,\cdots,n\}}$ and $(A''_j)_{j\leqslant j_0}$. Let G' be the intersection graphs of $\mathcal{A}'$. We set the weights w of G' in the following way: $w(v'_j) = w(v''_j) = 2k(j_0+1-j)$ if $j \leqslant j_0$ and $w(v'_j) = 1$ for $j > j_0$. The description of instance $I' = (G', w')$ of MIN WEIGHTED NODE COLORING is now complete.

Note that $\{v_j, j \leqslant j_0\}$ is a clique in G. We can suppose that $k \geqslant j_0$ (otherwise G is trivially not k-colorable). We claim that $\chi(G) \leqslant k$ (where $\chi(G)$ denotes the chromatic number of G) if and only if $\mathrm{opt}(G') \leqslant kj_0(j_0+1) + k - j_0 = B$.

Suppose that $\mathcal{S} = (S_1,\cdots,S_k)$ is a coloring of G. Then, set $\mathcal{S}' = (S'_1,\cdots,S'_k)$ where $S'_i = S_i \backslash \{v_j : j \leqslant j_0\} \cup \{v'_j, v''_j : v_j \in S_i, j \leqslant j_0\}$. We can easily see that $\mathcal{S}'$ is a coloring of G'. Furthermore, we have $\mathrm{opt}(G') \leqslant w(\mathcal{S}') = 2k\sum_{j=1}^{j_0} j + (k - j_0) = B$.

Conversely, let $\mathcal{S}' = (S'_1,\cdots,S'_l)$ be a coloring of G' with $\mathrm{opt}(I') = w(\mathcal{S}') \leqslant B$. Assume that $w(S_i) \geqslant w(S_j)$ for any $j \geqslant i$ and note that $\{v'_1, v''_1\} \in S'_1$, otherwise $\mathrm{opt}(I') = w(\mathcal{S}') \geqslant 2kj_0 + 2kj_0 + 2k\sum_{j=1}^{j_0-2} j = kj_0(j_0+1) + 2kj_0 > B$. With a similar argument, we can show that $\{v'_j, v''_j\} \in S'_j$ for any $j \leqslant j_0$. Consequently, $w(\mathcal{S}') = kj_0(j_0+1) + (l - j_0)$ and then $l \leqslant k$. Set $S_i = S'_i \setminus \{v'_i, v''_i\} \cup v_i$ for $i \leqslant j_0$ and $S_i = S'_i$ for $i > j_0$. $\mathcal{S} = (S_1;\cdots,S_l)$ is a l-coloring of G, and $\chi(G) \leqslant l \leqslant k$. ∎

Using Theorem 10.14 and the characterization of interval graphs, we deduce that the MIN WEIGHTED NODE COLORING problem is strongly **NP**-hard in co-comparability graphs.

10.8. Bibliography

[ALF 00] ALFANDARI L., PASCHOS V. T., "Master-slave strategy and polynomial approximation", *Comput. Opti. Appl.*, vol. 16, p. 231–245, 2000.

[BER 73] BERGE C., *Graphs and Hypergraphs*, North-Holland Publishing Company, 1973.

[BOD 94] BODLAENDER H. L., JANSEN K., WOEGINGER G. J., "Scheduling with Incompatible Jobs", *Discrete Applied Mathematics*, vol. 55, num. 3, p. 219-232, 1994.

[BOP 92] BOPPANA B. B., HALLDÓRSSON M. M., "Approximating maximum independent sets by excluding subgraphs", *BIT*, vol. 32, num. 2, p. 180–196, 1992.

[BOU 00] BOUDHAR M., FINKE G., "Scheduling on a batch machine with job compatibilities", *Special issue ORBEL-14: Emerging challenges in operations research (Mons, 2000). Belg. J. Oper. Res. Statist. Comput. Sci.*, vol. 40, num. 1-2, p. 69–80, 2000.

[BRA 99] BRANDSTÄDT A., LE V. B., SPINRAD J. P., *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[BUR 85] BURKARD R. E., "Time-slot assignment for TDMA-systems", *Computing*, vol. 35, num. 2, p. 99–112, 1985.

[CHU 90] CHUNG F. R., GYÁRFÁS A., TUZA Z., TROTTER W. T., "The maximum number of edges in $2K_2$-free graphs of bounded degree", *Discrete Mathematics*, vol. 81, num. 2, p. 129–135, 1990.

[CHV 84] CHVÀTAL V.., "Perfectly ordered graphs", In C. Berge and V. Chvàtal, editors, *Topics on Perfect Graphs*, of *Annals of Discrete Math.*, vol. 21, p. 253–277, 1984.

[DEM 04] DEMANGE M., DE WERRA D., ESCOFFIER B., MONNOT J., PASCHOS V. TH., "Weighted coloring on planar, bipartite and split graphs: complexity and improved approximation", FLEISCHER R., TRIPPEN G., Eds., *Proceedings of the* 15th *International Symposium on Algorithms and Computation*, vol. 3341 of *LNCS*, Springer-Verlag, p. 896–907, 2004.

[DEM 07] DEMANGE M., DE WERRA D., MONNOT J., PASCHOS V. TH., "Time slot scheduling of compatible jobs", *J. of Scheduling*, (to appear), 2007.

[ESC 06] ESCOFFIER B., MONNOT J., PASCHOS V. TH., "Weighted Coloring: further complexity and approximability results.", *Information Processing Letters*, vol. 97, num. 3, p. 98-103, 2006.

[GAR 75] GAREY M. R., JOHNSON D. S., "Complexity results for multiprocessor scheduling under resource constraints", *SIAM J. Comput.*, vol. 4, p. 397–411, 1975.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and intractability: a guide to the theory of* ***NP****-completeness*, Freeman, 1979.

[GAR 80] GAREY M. R., JOHNSON D. S., MILLER G. L., PAPADIMITRIOU C. H., "The complexity of coloring circular arcs and chords", *SIAM Journal on Algebraic and Discrete Methods*, vol. 1, p. 216–227, 1980.

[GRO 59] GROTZSCH H., "Ein dreifarbensatz fur dreikreisfreie netze auf der Kugel", *Wiss. Z. Martin Luther Univ. Halle-Wittenberg, Math. Naturwiss Reihe*, vol. 8, p. 109–120, 1959.

[GUA 97] GUAN D. J., ZHU X., "A Coloring Problem for Weighted Graphs", *Information Processing Letters*, vol. 61, num. 2, p. 77–81, 1997.

[HAM 90] HAMMER P. L., PELED U. N., SUN X., "Difference graphs", *Discrete Applied Mathematics*, vol. 28, num. 1, p. 35–44, 1990.

[HAS 05] HASSIN R., MONNOT J., "The maximum saving partition problem", *Operations Research Letters*, vol. 33, p. 242–248, 2005.

[HOL 81] HOLYER I., "The **NP**-completeness of edge-colouring", *SIAM J. Comput.*, vol. 10, num. 4, p. 718–720, 1981.

[HUJ 93] HUJTER M., TUZA Z., "Precoloring extension. II. Graph classes related to bipartite graphs", *Acta Mathematica Universitatis Comeniane*, vol. LXII, num. 1, p. 1–11, 1993.

[HUJ 96] HUJTER M., TUZA Z., "Precoloring extension. III. Classes of perfect graphs", *Combinatorics, Probability and Computing*, vol. 5, num. 1, p. 35–56, 1996.

[JAN 97] JANSEN K., SCHEFFLER P., "Generalized coloring for tree-like graphs", *Discrete Applied Mathematics*, vol. 75, num. 2, p. 135–155, 1997.

[JOH 74] JOHNSON D. S., "Approximation algorithms for combinatorial problems", *J. Comput. System Sci.*, vol. 9, p. 256–278, 1974.

[KAR 72] KARP R. M., "Reducibility among combinatorial problems", In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, Plenum Press, New York, p. 85–103, 1972.

[KRA 93] KRATOCHVÍL J., "Precoloring extension with fixed color bound", *Acta Mathematica Universatis Comenianae*, vol. 62, p. 139–153, 1993.

[KRA 94] KRATOCHVÍL J., TUZA Z., "Algorithmic complexity of list colorings", *Discrete Applied Mathematics*, vol. 50, num. 3, p. 297–302, 1994.

[LIC 82] LICHTENSTEIN D., "Planar formulae and their uses", *SIAM Journal on computing*, vol. 11, num. 2, p. 329–343, 1982.

[PAP 94] PAPADIMITRIOU C. H., *Computational complexity*, Addison-Wesley publishing company, 1994.

[PEM 04] PEMMARAJU S. V., RAMAN R., VARADARAJAN K. R.., "Buffer minimization using max-coloring", *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms 2004*, p. 562-571, 2004.

[PEM 05] PEMMARAJU S. V., RAMAN R., "Approximation Algorithms for the Max-Coloring Problem", *In proceedings of the* 32nd *International Colloquium on Automata, Languages and Programming*, vol. 3580 of *LNCS*, p. 1064–1075, 2005.

[POT 00] POTTS C. N., KOVALYOV M. Y., "Scheduling with batching: a review", *European Journal of Operational Research*, vol. 120, p. 228–249, 2000.

[POT 01] POTTS C. N., STRUSEVICH V. A., TAUTENHAHN T., "Scheduling batches with simultaneous job processing for two-machine shop problems", *J. of Scheduling*, vol. 4, p. 25–51, 2001.

[REN 85] RENDL F., "On the complexity of decomposing matrices arising in satellite communication", *Operations research letters*, vol. 4, p. 5–8, 1985.

[SIM 90] SIMON H. U., "On approximate solutions for combinatorial optimization problems", *SIAM J. Disc. Math.*, vol. 3, num. 2, p. 294–310, 1990.

[WER 90] DE WERRA D., "Heuristics for graph coloring", *Computing*, vol. 7, p. 191–208, 1990.

[WER 02] DE WERRA D., DEMANGE M., MONNOT J., PASCHOS V. TH., "Weighted node coloring: when stable sets are expensive", *Proceedings of the* 28th *International Workshop on Graph-Theoretic Concepts in Computer Science*, vol. 2573 of *LNCS*, Springer-Verlag, p. 114–125, 2002.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 11

Weighted Edge Coloring

11.1. Introduction

In several communication systems [INU 79, BON 81, GOP 85, REN 85, RIB 89, PRA 00, KES 04] messages are to be transmitted in a single hop from senders to receivers through direct connections established by an underlying switching network. In such a system, a sender (resp. receiver) cannot send (resp. receive) more than one message at a time, while the transmission of messages between different senders and receivers can take place simultaneously. The scheduler of such a system establishes successive configurations of the switching network, each one routing a non-conflicting subset of the messages from senders to receivers. Given the transmission time of each message, the transmission time of each configuration equals to that of the heaviest message transmitted. Moreover, in practice there is a significant set-up delay to establish each configuration. Note, for example, that in optical switches this overhead may dominate the transmission time. The aim of the scheduler is to find a sequence of configurations such that all the messages are finally transmitted and the total transmission time (including set-up delays) is minimized.

This problem can be formulated in graph-theoretic terms as following: senders and receivers can be considered as the vertices V of a weighted graph $G = (V, E)$, whose edges E correspond to messages and their weights $w(e),\ e \in E$, to the lengths (transmission times) of messages. Although the graph G obtained is originally a weighted directed one, it can be considered as an undirected one, since the directions of its edges do not play any role in the objective function we study here. Clearly, a configuration

Chapter written by Marc DEMANGE, Bruno ESCOFFIER, Giorgio LUCARELLI, Ioannis MILIS, Jérôme MONNOT, Vangelis Th. PASCHOS and Dominique DE WERRA.

corresponds to a matching or a color (in terms of edge coloring) of the graph G. In this context, we ask for a partition $M = \{M_1, M_2, \ldots, M_s\}$ of the set of edges E of the graph G into matchings (colors), each one of weight $w_i = \max\{w(e)|e \in M_i\}$, such that $W = \sum_{i=1}^{s} w_i$ is minimized.

Due to the fact that the weight w_i of each color is defined as the maximum weight of the edges colored M_i, in what follows we shall refer to this problem as Max-weight Edge Coloring (MEC) problem. Clearly, if all the edges of G have the same weight then the MEC problem amounts to the traditional *edge coloring* problem, where the objective is to minimize the number of colors (matchings) required in order to assign different colors to neighboring edges.

An alternative way to represent the input as well as the solution is by using a traffic matrix $A = (a_{xy})$, where every element a_{xy} is equal to the transmission time from sender x to receiver y. The goal is to find a collection of switching matrices (i.e., with at most one non-zero entry in each row and in each column) $A_1, A_2, ..., A_s$, such that: (i) $A_1 + A_2 + \ldots + A_s = A$ and (ii) $\sum_{i=1}^{s} w(A_i)$ is minimized, where $w(A_i) = \max\{a_{xy}|\ a_{xy} \in A_i\}$.

The MEC problem is equivalent to the parallel batch scheduling problem with (in)compatibilities between jobs. According to the standard notation for scheduling problems we denote this variant as $1 \mid p - batch,\ graph \mid C_{max}$. In this variant jobs are no longer independent but they correspond to the edges of a weighted $graph$. Edge weights correspond to processing times of jobs and the graph G describes (in)compatibilities between jobs, i.e. jobs corresponding to adjacent edges cannot be scheduled in the same batch (color).

During the last years, several complexity and approximability results for the MEC problem have been presented. Table 11.1 summarizes all the existing and known results for the MEC problem.

In this chapter, which is an extension of Chapter 10 to edge coloring, we review the known complexity and approximability results for the MEC problem. In the next section we present briefly some related problems, while in section 11.3 we give the notation we use and some preliminaries. In section 11.4 the most recent **NP**-completeness proof for the MEC problem on cubic planar bipartite graphs is presented, while in section 11.5 a polynomial algorithm for graphs of maximum degree two is shown. Then, a 2-approximation algorithm for general graphs is presented in section 11.6. In section 11.7 three approximation algorithms for some special cases of bipartite graphs are presented. Finally, in section 11.8 we present two polynomial algorithms for trees: the first one for trees of bounded degree and the second one for stars of chains. We conclude in section 11.9.

Class of graph G	Known results
$\Delta = 2$	Polynomially solvable [ESC 06]
Stars of chains	Polynomially solvable [LUC 07]
Trees	Polynomially solvable for bounded Δ [LUC 07] Open if Δ is part of the instance
Bipartite	**NP**-hard and $7/6$-inapproximable, even if G is cubic and planar [WER 04] $(2\Delta - 1)/3$ approximation [DEM 02] $\Delta = 3 : 7/6$ approximation [WER 04] $\Delta \leqslant 7 : \rho(\Delta) < 2$ approximation [LUC 07]
General	2 approximation [KES 04] $\Delta \leqslant 4 : \rho(\Delta) < 2$ approximation [ESC 06, LUC 07]

Table 11.1. *Known results for the MEC problem*

11.2. Related problems

A natural idea to decrease the cost of a solution to such a weighted edge coloring problem is to allow the division of each edge e of G into parallel edges of weights adding up to $w(e)$. In fact, this idea corresponds to the notion of pre-emption in scheduling problems: interrupt the execution of a job (the transmission of a message) and complete it later. The weight of each color M_i is now defined as $w_i = \max\{w_i(e)|e \in M_i\}$, where $w_i(e)$ is the portion of the e's weight belonging to matching M_i, and we ask for a partition M such that $\sum_{i=1}^{s} w_i(e) = w(e), \forall e \in E$, and $W = \sum_{i=1}^{s} w_i$ is minimized. We shall refer to this variant as pre-emptive MEC (p-MEC) problem.

The existence of a set-up delay, d, encountered in practical applications to establish each matching (configuration), does not make any difference for the MEC problem: this can be taken into account by increasing the weight of all edges of G by d. Thus, the weight of each matching in M will be also increased by d, incorporating the set-up delay for this matching.

On the contrary, the presence of such a parameter affects crucially the complexity of the p-MEC problem. In the absence of d the p-MEC problem is equivalent to the pre-emptive open shop scheduling problem which can be solved optimally in polynomial time [LAW 78]. However, in the presence of d the p-MEC problem becomes strongly **NP**-hard [GOP 85] and non-approximable within a factor less than $7/6$ [CRE 01]. Approximation algorithms of factors 2 and $2 - \frac{1}{d+1}$ have been presented in [CRE 01] and [AFR 05], respectively.

Max-weight Vertex Coloring (MVC) problem, the analog to the MEC problem, has been studied more extensively in other works during the last years [BRU 98, FIN 04, DEM 02, WER 04, ESC 06, PEM 04, PEM 05]. In the MVC problem we ask for a partition of the vertices of G into independent sets (colors), each one of weight equal to the maximum weight of its vertices, so that the total weight of the partition is minimized.

Note that the MEC problem, on a general graph G, is equivalent to the MVC problem on the line graph, $L(G)$, of G and thus any algorithm for the MVC problem applies also to the MEC problem. However, this is not true for special graph classes, since the line graph of a special graph (e.g. bipartite or tree) is no longer in the same special class.

11.3. Preliminaries and notation

In what follows, we consider the MEC problem on a weighted graph $G = (V, E)$. By $d(v),\ v \in V$, we denote the degree of vertex v and by $\Delta(G)$ (or simply Δ) the maximum vertex degree of G. We define the degree of each edge $e(u, v) \in E$ as $d(u, v) = d(u) + d(v)$ and $\Delta'(G)$ (or simply Δ') denotes the maximum edge degree.

It is well known that the traditional *edge coloring* problem, is **NP**-hard even in cubic graphs [HOL 81], although its optimal solution is either Δ or $\Delta + 1$ [VIZ 64]. On the other hand, it is solvable in polynomial time for bipartite graphs [KON 16]. Obviously, by applying such an algorithm on a weighted bipartite graph we obtain a Δ-colors solution, in general non-optimal, to the MEC problem. For the number of matchings in an optimal solution of the MEC problem the following bound holds.

PROPOSITION 11.1.– *For the number of matchings, s^*, in an optimal solution it holds that $\Delta \leqslant s^* \leqslant \Delta' - 1 \leqslant 2\Delta - 1$.*

Proof. Any solution consists of at least Δ matchings, since there is a vertex with exactly Δ adjacent edges which belong in different matchings.

Assume that an optimal solution consists of Δ' or more matchings. Consider those matchings sorted in non-increasing order of their weights. Each edge of G has at most $\Delta' - 2$ neighboring edges. So, for each edge e in any $(\Delta' + i)$-th matching, $i > 0$, there is one of the first $\Delta' - 1$ matchings where e can be moved without increasing the weight of this matching.

The last part of the inequality follows directly from the definition of Δ'. ∎

Henceforth, we consider the edges of G sorted in non-increasing order of their weights, $w(e_1) \geqslant w(e_2) \geqslant \ldots \geqslant w(e_m)$. Thus, e_1 denotes the heaviest edge of G.

By opt we denote the cost of an optimal solution to the MEC problem. We also assume that in such an optimal solution the graph is decomposed into s^* matchings, each one of weight w_i^*. Without loss of generality we consider the matchings of any solution in non-increasing order of their weights, i.e. $w_1 \geqslant w_2 \geqslant \ldots \geqslant w_s$, and for the optimal solution $w_1^* \geqslant w_2^* \geqslant \ldots \geqslant w_{s^*}^*$.

11.4. Complexity and (in)approximability

Kesselman and Kogan [KES 04], by extending the results of the p-MEC problem given in [GOP 85, CRE 01], prove that the MEC problem is **NP**-hard and $7/6$ inapproximable even in bipartite graphs of maximum degree three and the edge weights restricted to be 1, 2 or 3. Another **NP**-hardness proof for the equivalent formulation of the MEC problem on bipartite graphs in terms of matrix decomposition has been proposed in [REN 85].

Demange *et al.* [DEM 02] prove that the MEC problem is strongly **NP**-complete for k-regular bipartite graphs for $k \geqslant 3$. They, also, prove that it is **NP**-complete to approximate the MEC problem for these graphs within a ratio less than $\frac{2^k}{2^k-1}$, which for $k = 3$ becomes $8/7$.

Here, we present a proof for the strongly **NP**-completeness of the MEC problem in cubic bipartite planar graphs proposed in [WER 04]. This proof also yields to an $7/6$ inapproximability result.

THEOREM 11.1.– *The* MEC *problem is strongly* ***NP****-complete in cubic bipartite planar graphs even if the edge weights are restricted to* 1, 2 *or* 3.

Proof. In [MAR 05] it has been shown that the following *Edge Precoloring Extension* (PREXT) problem, in cubic bipartite planar graphs, is strongly **NP**-complete: given a cubic bipartite planar graph $G = (V, E)$ and 3 pairwise disjoint matchings $E_1, E_2, E_3 \subseteq E$, we ask if it is possible to extend the edge precoloring E_1, E_2, E_3 to a proper edge 3-coloring of G.

An instance $I = (G', w)$ of the MEC problem will be constructed such that the answer to an instance (G, E_1, E_2, E_3) of the PREXT problem is yes if and only if there exists a solution for the MEC problem on I with cost $W' \leqslant 6$. The construction of instance I is the following:

– Each edge in E_1 receives weight 3.

– Each edge $(x, y) \in E_2$ is replaced by a gadget F_2 described in Figure 11.1, where x and y correspond to v_0 and v_9, respectively.

– Each edge in E_3 is replaced by a gadget F_3 which is the same as the gadget F_2 except that the weights 1 and 2 have been exchanged.

– The remaining edges of G receive weight 1.

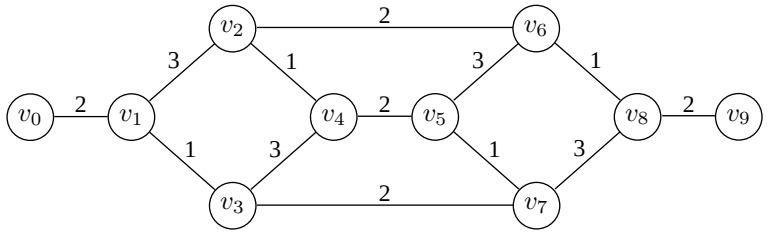


Figure 11.1. *Gadget F_2 for $e \in E_2$*

Note that G' is still a bipartite cubic planar graph.

Assume, first, that G admits an edge 3-coloring $M = \{M_1, M_2, M_3\}$ where $E_i \subseteq M_i$, for any $i = 1, 2, 3$. We get a coloring $M' = \{M'_1, M'_2, M'_3\}$ of G', where $M'_1 = M_1 \cup \{e \in F_2 \cup F_3 : w(e) = 3\}$ and, for $i = 2, 3$, $M'_i = (M_i - E_i) \cup \{e \in F_2 \cup F_3 : w(e) = 4 - i\}$. It is easy to check that $W' \leqslant 3 + 2 + 1 = 6$.

Conversely, consider an edge coloring $M' = \{M'_1, M'_2, \ldots, M'_s\}$ of G' with $W' \leqslant 6$ and w.l.o.g. assume that $w'_1 \geqslant w'_2 \geqslant \ldots \geqslant w'_s$. It holds that $s \geqslant 3$, since $\Delta(G') = 3$ (Proposition 11.1). All the edges of weight 3 must be in the matching M'_1, and no edge of weight 2 is in M'_p, with $p \geqslant 3$, since otherwise $W' \geqslant 7$ ($3 + 3 + 1$ in the first case and $3 + 2 + 2$ in the second case). Moreover, each edge of weight 2 is adjacent to an edge of weight 3 and, then, these edges are necessarily in M'_2. Finally, note that the edges of the gadgets of weight 1 are neighbors to an edge of weight 2 and an edge of weight 3 and must be in M'_p with $p \geqslant 3$. Therefore, it holds that $s = p = 3$, since $W' \leqslant 6$. Thus, consider the edge coloring $M = \{M_1, M_2, M_3\}$ of G, where $M_i = (M'_i - \{e \in F_2 \cup F_3 : w(e) = 4 - i\}) \cup E_i$, for any $i = 1, 2, 3$. It is easy to see that $\{M_1, M_2, M_3\}$ is a solution for the PREXT problem. ∎

THEOREM 11.2.– *For all $\varepsilon > 0$, the* MEC *problem is not $7/6 - \varepsilon$ approximable in bipartite cubic planar graphs, unless* ***P*** $=$ ***NP***.

11.5. Graphs of $\Delta = 2$

A graph G with $\Delta(G) = 2$ is a collection of chains and cycles. We will call such a chain or cycle a component of G. According to Proposition 11.1, the number of matchings in an optimal solution of the MEC problem in graphs with $\Delta(G) = 2$ is either two or three. Specifically, if G does not contain an odd cycle, then the optimal

solution for the MEC problem could have two or three matchings. Otherwise, the optimal solution has exactly three matchings, since an odd cycle cannot be decomposed into less than three matchings.

The MEC problem is polynomial for graphs of maximum degree $\Delta = 2$. This result follows from the same variant of the MVC problem. In [ESC 06] has been presented an $O(|V|^2)$ algorithm for the MVC problem on chains, which can be easily adapted for the MVC problem on graphs of maximum degree $\Delta = 2$ (collections of chains and cycles). Moreover, if G is a graph of maximum degree 2, then its line graph $L(G)$ is also a graph with $\Delta(L(G)) = 2$.

The algorithm which is presented in [ESC 06], denoted by DEGREE_2 and adapted for the MEC problem is as follows:

```
 - DEGREE_2
1. for each edge weight c = w(e_1), w(e_2), ... do
2.   Insert into M_1 each edge e ∈ E with w(e) > c
3.   If M_1 is a matching then
4.     For each pair of edges x, y ∈ E with w(x), w(y) > c
       in the same connected component in G,
       without any other edge e ∈ E with w(e) > c between them
5.       If the number of edges between x and y is even then
6.         Insert into M_3 the minimum-weight edge
           between x and y
7.       Insert alternately in M_1 and M_2 the (remaining) edges
         between x and y
8. Return the best solution found
```

Algorithm DEGREE_2 creates at most $|E|$ different solutions and returns the minimum of them. For each possible weight of the second matching c the algorithm finds the minimum weight of the third one, while the weight of the first one is known and equal to the heaviest edge of the graph, i.e. $w_1^* = w(e_1) = \max\{w(e), e \in E\}$. Thus, in the iteration where $c = w_2^*$ DEGREE_2 finds the optimal. Note that can find the optimal solution with two matchings, if such a solution exists and it is optimal, since in the iteration where $c = w_2^*$ the third matching may not contain any edge, i.e. $w_3 = 0$. Finally, observe that algorithm DEGREE_2 can be applied in graphs with cycles. In this case, edges may exist both between x and y and between y and x (see Line 4), but this fact does not affect the algorithm.

THEOREM 11.3.– *An optimal solution to the MEC problem for graphs of maximum degree $\Delta = 2$ consists of at most three (i.e., two or three) matchings and can be found in $O(|E|^2)$ time.*

11.6. A 2-approximation algorithm for general graphs

Since the weight w_i of the matching M_i is equal to the weight of its maximum edge, a greedy algorithm must group in the same matching edges whose weights are close to one another. Kesselman *et al.* in [KES 04] used this intuition and presented an algorithm with a tight approximation ratio of 2. Note that in [KES 04] it is proved that this approximation ratio holds for bipartite graphs; however, it is easy to see that this result also holds for general graphs.

The greedy approximation algorithm, denoted by GREEDY, proposed in [KES 04] for the MEC problem for any graph follows.

- GREEDY
1. Sort the edges of G in non-increasing order of their weight
2. Using this order find greedily successive maximal matchings $M_1, M_2, \ldots, M_s$

THEOREM 11.4.– *Algorithm GRREDY is a tight 2-approximation algorithm for the MEC problem.*

Proof. Let $M = \{M_1, M_2, \ldots, M_s\}$ be the coloring constructed by the greedy algorithm. Obviously, it holds that $w_1 \geqslant w_2 \geqslant \ldots \geqslant w_s$. Consider the i-th matching and let $e = (u, v)$ be the heaviest edge in the i-th matching, i.e., $w_i = w(e)$. Let, also, x and y be the number of edges of weight at least $w(e)$ that are incident to u and v, respectively. Since the edge e has not been selected by the algorithm until the i-th matching, it follows that at least one of these $x+y$ edges has been selected in each one of the matchings $M_1, M_2, \ldots, M_{i-1}$; otherwise, the edge e would have been already selected. Therefore, $x + y \geqslant i$. Hence, we can assume w.l.o.g. that $x \geqslant \lceil i/2 \rceil$.

Let us now assume an optimal solution to the MEC problem with weight $\text{opt} = w_1^* + w_2^* + \ldots + w_{s^*}^*$, where $w_1^* \geqslant w_2^* \geqslant \ldots \geqslant w_{s^*}^*$. In order for this solution to cover all the heaviest edges incident to u it must contain at least $x \geqslant \lceil i/2 \rceil$ matchings of weight greater than or equal to $w(e)$. Therefore, the weight of the i-th matching in M is at most the weight of the $\lceil i/2 \rceil$-th matching in the optimal solution, i.e., $w_i \leqslant w_{\lceil i/2 \rceil}^*$.

Moreover, observe that $s^* \geqslant \Delta$ and $\Delta \leqslant s \leqslant 2\Delta - 1$. Thus, it holds that $s^* \geqslant \lceil s/2 \rceil$.

Summing over all $i = 1, \ldots, s$, we obtain:

$$W = \sum_{i=1}^{s} w_i \leqslant \sum_{i=1}^{s} w_{\lceil i/2 \rceil}^* \leqslant 2 \sum_{i=1}^{\lceil s/2 \rceil} w_i^* \leqslant 2 \sum_{i=1}^{s^*} w_i^* \leqslant 2 \cdot \text{opt}$$

In order to show the tightness of the algorithm, consider a bipartite instance $G = (U, V; E)$, where $|U| = n - 1$ and $|V| = n$. For each $u_i \in U$ and $v_j \in V$, $1 \leqslant i, j \leqslant n - 1$, there is an edge $e = (u_i, v_j) \in E$ with weight $w(e) = a + \varepsilon$, $\varepsilon > 0$. Moreover, for each $u_i \in U$, $1 \leqslant i \leqslant n - 1$, there is an edge $e = (u_i, v_n)$ that connects it with v_n, with weight $w(e) = a$.

Algorithm GREEDY, taking for input the above instance, returns a solution that consists of $2(n - 1)$ matchings. In the first $n - 1$ matchings all the $(n - 1)^2$ edges of weight $a + \varepsilon$ are selected. Next, GREEDY creates $n - 1$ matchings each one containing an edge of weight a adjacent to the vertex v_n.

On the other hand, the optimal solution contains only n matchings: in the i-th matching are selected $n - 2$ edges of weight $a + \varepsilon$ from all the $u \in U$, but u_i, to all the $v \in V$, but v_i and v_n, plus the edge (u_i, v_n) of weight a. Thus, it holds that:

$$\frac{W}{\text{opt}} = \frac{(a + \varepsilon) \cdot (n - 1) + a \cdot (n - 1)}{(a + \varepsilon) \cdot n} = \left(2 - \frac{\varepsilon}{a + \varepsilon}\right) \cdot \frac{n - 1}{n}$$

which asymptotically matches the 2 approximation ratio. ∎

11.7. Bipartite graphs

Yu [YU 06] shows that the line graph of a bipartite planar graph is an interval graph. Thus, taking into account that [PEM 04] presented a 2-approximation algorithm for the MVC problem in interval graphs, Yu proved that the MEC problem is 2 approximable in bipartite planar graphs. Nevertheless, this result is dominated by the more general algorithm given in [KES 04] (see section 11.6).

In this section, three algorithms that improve the two approximation ratio given in [KES 04] (see section 11.6) in some special cases in bipartite graphs are presented.

11.7.1. *A $\frac{2\Delta-1}{3}$-approximation algorithm*

Observe that, according to König's theorem [KON 16] in bipartite graphs, the optimal solution of the (unweighted) edge coloring problem consists of exactly Δ matchings. Thus, an algorithm that just creates a Δ-coloring for the bipartite graph G has an approximation ratio Δ, for any $\Delta \geqslant 3$, for the MEC problem, since $W \leqslant \Delta \cdot w_1^* \leqslant \Delta \cdot \text{opt}$.

Demange *et al.* [DEM 02] presented a tight approximation algorithm for the MEC problem in bipartite graphs. This algorithm has an unbounded ratio equal to $(2\Delta - 1)/3$, for any $\Delta \geqslant 3$. The algorithm, denoted by MEC_ALGO_1, follows.

```
 - MEC_ALGO_1
1. For each weight c = w(e_1), w(e_2), ... do
2.   Split G into two edge induced subgraphs:
       - G_1 = {e|w(e) > c}
       - G_2 = {e|w(e) ⩽ c}
3.   If Δ(G_1) ⩽ 2 then
4.     Find the optimal solution of at most 2 colors for G_1
5.     Use algorithm GREEDY for G_2, completing the above
solution
6. Return the best solution found
```

THEOREM 11.5.– *Algorithm* `MEC_ALGO_1` *is a tight* $(2\Delta - 1)/3$*-approximation algorithm for the MEC problem in bipartite graphs.*

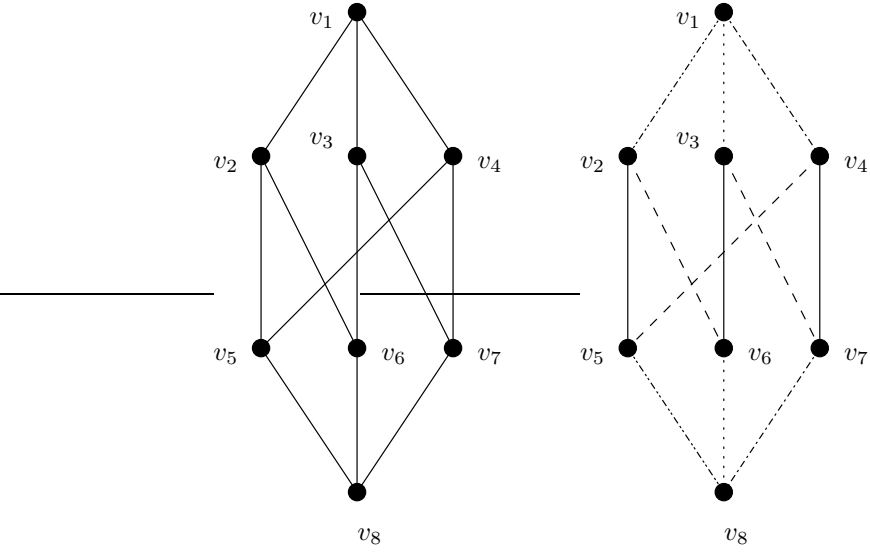
Proof. Algorithm `MEC_ALGO_1` returns the best solution found with weight $W = \min\{w_1 + w_2 + w_3 + \ldots + w_s\}$ (the weights $w_{\Delta+1}, \ldots, w_{w_s}$ may be equal to zero) for different values of $c = w_3$. Note that the algorithm does not create a solution with more than $2\Delta - 1$ matchings, i.e. $s \leqslant 2\Delta - 1$. Consider the cost of the solution found by the algorithm in the iteration for which $w_3 = w_3^*$. In this iteration the edges of G_1 are a subset of the edges in the two heaviest matchings of the optimal solution. Therefore, for this iteration it holds that $W \leqslant w_1^* + w_2^* + (2\Delta - 3) \cdot w_3^*$, since $\Delta(G_1) \leqslant 2$ and thus an optimal solution of at most 2 colors can be found for it. The edges added in M_1 and M_2 in line 5 of the algorithm do not increase the weights of these matchings. Therefore, observing that $\text{opt} \geqslant w_1^* + w_2^* + w_3^* \geqslant 3 \cdot w_3^*$, it holds that:

$$\begin{aligned} W &\leqslant w_1^* + w_2^* + (2\Delta - 3) \cdot w_3^* \\ &\leqslant \text{opt} + (2\Delta - 4) \cdot w_3^* \\ &\leqslant \text{opt} + (2\Delta - 4) \cdot \tfrac{\text{opt}}{3} \\ &\leqslant \tfrac{2\Delta - 1}{3} \cdot \text{opt} \end{aligned}$$

Next, the tightness of this approximation ratio when $\Delta = 3$ is shown. In this case the ratio of algorithm `MEC_ALGO_1` becomes $5/3$.

Consider a graph $G = (V, E)$ with $\Delta(G) = 3$, where $V = \{v_1, v_2, \ldots, v_8\}$ and $E = \{(v_1, v_j) : j = 2, 3, 4\} \cup \{(v_2, v_j) : j = 5, 6\} \cup \{(v_3, v_j) : j = 6, 7\} \cup \{(v_4, v_j) : j = 5, 7\} \cup \{(v_i, v_8) : i = 5, 6, 7\}$ (see Figure 11.2.a). Set $w(v_2, v_5) = w(v_3, v_6) = w(v_4, v_7) = n$ and $w(u, v) = n - 1$ for the remaining edges.

In any iteration algorithm `MEC_ALGO_1` finds a solution containing exactly 5 matchings. For example, the solution $M = \{M_1, \ldots, M_5\}$ found by the algorithm in the case where $c = w_3^*$ is $M_1 = \{(v_2, v_5), (v_3, v_6), (v_4, v_7)\}$, $M_2 =$



(a) The graph G

(b) The solution calculated by algorithm MEC_ALGO_1

Figure 11.2. *The graph illustrating the tightness of the ratio of algorithm MEC_ALGO_1*

$\{(v_2, v_6), (v_3, v_7), (v_4, v_5)\}$, $M_3 = \{(v_1, v_2), (v_5, v_8)\}$, $M_4 = \{(v_1, v_3), (v_6, v_8)\}$ and $M_5 = \{(v_1, v_4), (v_7, v_8)\}$ (see Figure 11.2.b). On the other hand, the optimal solution has exactly three matchings. Thus, the tightness is proved by considering n tending to ∞. ∎

11.7.2. *A $\frac{7}{6}$-approximation algorithm for $\Delta = 3$*

In order to improve the result given in [DEM 02] (see section 11.7.1), de Werra *et al.* [WER 04] presented an approximation algorithm for the MEC problem in bipartite graphs with maximum degree three. This algorithm achieves an approximation ratio equal to the known $7/6$ inapproximability result for bipartite graphs with $\Delta = 3$.

The proposed algorithm consists of three subroutines and selects the minimum solution found by them. Each subroutine creates iteratively a different partition of the edges of graph G in such a way that the degree and the ranges of edge weights of each partition are known. A bound to the cost of the solution found, with respect to the optimal solution, is proved for each subroutine. Combining these bounds the

$7/6$-approximability result is obtained. In this section, the three subroutines and the complete proof are presented.

Recall that the edges of the graph $G = (V, E)$ are sorted in non-increasing order of their weights, i.e. $w(e_1) \geqslant w(e_2) \geqslant \ldots \geqslant w(e_m)$. Denote by $G[V']$, $V' \subseteq V$, and $G[E']$, $E' \subseteq E$, the subgraph of G induced by V' and the partial graph of G induced by E', respectively. For any $i \leqslant m$, consider the subset $E_i = \{e_1, e_2, \ldots, e_i\} \subseteq E$ of the i heaviest edge of E and $\overline{E_i} = E - E_i$. Finally, V_i denotes the set of vertices of G incident to an edge in E_i, i.e., the subset of non-isolated vertices of $G[E_i]$.

Consider the following algorithm, denoted by HYBRID_MEC.

```
- HYBRID_MEC
1. For i = m downto 1 do
2.   Apply algorithm SOL1 on G[E_i]
3.   If SOL1 returns a non-empty set of solutions
4.     Greedily complete these solutions with the edges of Ē_i
       and find the best of them, let M_{1,i}
5.   For j = i downto 1 do
6.     Apply algorithm SOL2 on G[E_j]
7.     If SOL2 returns a non-empty set of solutions
8.       Greedily complete these solutions with the edges of Ē_j
         and find the best of them, let M_{2,j,i}
9.     Apply algorithm SOL3 on G[E_j]
10.    If SOL3 returns a non-empty set of solutions
11.      Greedily complete these solutions with the edges of Ē_j
         and find the best of them, let M_{3,j,i}
12. Return M = argmin{w_{1,i}, w_{2,j,i}, w_{3,j,i} : j = 1, 2, ..., i, i = 1, 2, ..., m}
```

The greedy steps 4, 8 and 11 can be described as follows: for each edge which is not yet colored, try to color it with an existing color, and otherwise take a new color. A simple argument shows that these edge colorings do not use more than 5 colors. Indeed, assume the reverse and consider an edge with color M_6. Since the maximum degree of G is 3, this edge is adjacent to at most 4 edges and then to at most 4 colors. Thus, this edge can be recolored with a missing color in $\{M_1, M_2, \ldots, M_5\}$. Obviously, this result also holds for an optimal solution (see Proposition 11.1).

The three algorithms used are applied on several partial graphs $G' = (V', E')$ of G, where $E' = \{e_1, e_2, \ldots, e_{m'}\}$. Let $\overline{V_i'} = V' - V_i'$ and $\overline{E_i'} = E' - E_i'$ be the complements of the sets V_i' and E_i', respectively. Moreover, let $M = \{M_1, M_2, \ldots, M_l\}$ be an edge coloring of G'. Define the index $i_j = \min\{k : e_k \in M_j\}$, for any $j = 1, 2, \ldots, l$. Assume, for reasons of readability, that some colors M_j may be empty

(in this case $i_j = m' + 1$). So, the principle of these three algorithms consists of finding a decomposition of G' into two subgraphs G'_1 and G'_2 having each a maximum degree two. If such a decomposition exists, G'_1 and G'_2 can be colored with at most 2 colors each, since both G and G' are bipartite graphs.

```
 - SOL1
1. For j = m' downto 1 do
2.   If the degree of G'[E'_j] is at most two then
3.     Consider the graph B:
         - induced by the vertices of G' incident to
           at least two edges of the set complement of E'_j
         - restricted to the edges of the set complement of E'_j
4.     If there exists a matching A^j of B
       such that every vertex of the set complement of V'_j is saturated
5.       Consider the decomposition G'_{1,j} and G'_{2,j} of G' induced
         by E'_j ∪ A^j and E' − (E'_j ∪ A^j), respectively
6.       Find an optimal edge coloring {M^j_1, M^j_2} among the edge
         2-colorings of G'_{1,j}
7.       Color greedily the edges of G'_{2,j}
         with two colors {M^j_3, M^j_4}
8.       Set M^j = {M^j_1, M^j_2, M^j_3, M^j_4} the edge coloring found
9. Return {M^j,  j = 1, 2, ..., m' − 1}
```

Note that the line 7 of algorithm SOL1 runs in polynomial time. Indeed, more generally, given a graph $G = (V, E)$ and a set $V' \subseteq V$, it is polynomial to determine if there exists a matching such that each vertex of V' is saturated. Consider the graph G' where all missing edges between vertices of $V - V'$ have been added to G. If $|V|$ is odd, then a vertex to the clique $V - V'$ is added. It is easy to see that G' has a perfect matching if and only if G has a matching such that each vertex of V' is saturated.

LEMMA 11.1.– *If $M = \{M_1, M_2, M_3, M_4\}$ with $w_1 \geqslant w_2 \geqslant w_3 \geqslant w_4$ is an edge coloring of G', then algorithm SOL1 produces a solution M^j satisfying $W^j \leqslant w_1 + w_2 + 2 \cdot w_3$.*

Proof. Consider the iteration of algorithm SOL1 corresponding to $j = i_3 - 1$. By construction, $G'[E'_{i_3-1}]$ is 2 edge colorable, since it holds that $E'_{i_3-1} \subseteq M_1 \cup M_2$. Moreover, in the subgraph induced by $\overline{E'_{i_3-1}}$, each vertex of degree 3 has at least one edge of $M_1 \cup M_2$ incident to it. Thus, in B, there exists a matching where each vertex of $\overline{V'_{i_3-1}}$ is saturated. The subgraph G'_{1,i_3-1} has a maximum degree 2 and contains by construction the subgraph $G'[E'_{i_3-1}]$. Moreover, any two connected components of $G'[E'_{i_3-1}]$ have not been merged in G'_{1,i_3-1}, since each edge $e = (x, y) \in A^{i_3-1}$ has at least one

vertex (for example, x) satisfying $d_{G'[E'_{i_3-1}]}(x) = 0$. Thus, any edge 2-coloring of $G'[E_{i'_3-1}]$ can be extended to an edge 2-coloring of G'_{1,i_3-1} . So, since $\forall e \in A^{i_3-1}$, $\forall e' \in E'_{i_3-1}\ w(e) \leqslant w(e')$, and $\{M_1^{i_3-1}, M_2^{i_3-1}\}$ is an optimal weighted 2 edge coloring of G'_{1,i_3-1}, it holds that:

$$w_1^{i_3-1} + w_2^{i_3-1} \leqslant w_1 + w_2 \qquad [11.1]$$

By construction, G'_{2,i_3-1} has no vertex with degree 3, and then G'_{2,i_3-1} has a maximum degree 2. Moreover, $\forall e \notin (A^{i_3-1} \cup E'_{i_3-1})$ and we have $w(e) \leqslant w(e_{i_3}) = w_3$. Thus, any edge coloring of G'_{2,i_3-1} using at most 2 colors and in particular $\{M_3^{i_3-1}, M_4^{i_3-1}\}$ satisfies:

$$w_3^{i_3-1} + w_4^{i_3-1} \leqslant 2 \cdot w_3 \qquad [11.2]$$

Combining [11.1] and [11.2], we obtain:

$W^j \leqslant W^{i_3-1} \leqslant w_1 + w_2 + 2 \cdot w_3$ ∎

```
 - SOL2
1. For k = m' downto 1 do
2.   If E'_k is a matching then
3.     If there exists a matching A_k of G'[V'_k] such that
       each vertex of G'[V'_k] having a degree 3 in G' is saturated
4.       Consider the decomposition G'_{1,k} and G'_{2,k} of G'
         induced by E'_k ∪ A_k and E' − (E'_k ∪ A_k), respectively
5.       Color G'_{1,k} with one color M_1^k
6.       Color greedily G'_{2,k} with two colors M_2^k and M_3^k
7.       Set M^k = {M_1^k, M_2^k, M_3^k} the edge coloring found
8. Return {M^k : k = 1, 2, ..., m'}
```

LEMMA 11.2.– *If $M = \{M_1, M_2, M_3\}$ with $w_1 \geqslant w_2 \geqslant w_3$ is an edge coloring of G', then algorithm SOL2 produces a solution M^k satisfying $W^k \leqslant w_1 + 2 \cdot w_2$.*

Proof. Consider the iteration of algorithm SOL2 corresponding to $k = i_2 - 1$. By construction, $E'_{i_2-1} \subseteq M_1$ and among $M_1 - E'_{i_2-1}$ there is a matching of $G'[\overline{V'_{i_2-1}}]$ where each vertex of degree 3 is saturated (otherwise, $M = \{M_1, M_2, M_3\}$ is not

feasible). Thus, G'_{1,i_2-1} can be considered and colored with one color $M_1^{i_2-1}$, and it holds that:

$$w_1^{i_2-1} = w_1 \qquad [11.3]$$

It is also deduced that G'_{2,i_2-1} has a maximum degree 2. Then, it can be edge colored with 2 colors $M_2^{i_2-1}$ and $M_3^{i_2-1}$. Moreover, since $\forall e \notin E'_{i_2-1}, w(e) \leqslant w(e_{i_2}) = w_2$, it is obtained that:

$$w_2^{i_2-1} + w_3^{i_2-1} \leqslant 2 \cdot w_2 \qquad [11.4]$$

Using [11.3] and [11.4], it holds that:

$$W^k \leqslant W^{i_2-1} \leqslant w_1 + 2 \cdot w_2 \qquad \blacksquare$$

```
 - SOL3
1. For k = m' downto 1 do
2.     If there is a matching A_k in G'[E'_k] such that
       each vertex of degree 3 in G' is saturated
3.       Consider the decomposition G'_{1,k} and G'_{2,k} of G'
         induced by A_k and E' - A_k, respectively
4.       Color G'_{1,k} with one color M_3^k
5.       Color greedily G'_{2,k} with two colors M_1^k and M_2^k
6.       Set M^k = {M_1^k, M_2^k, M_3^k} the edge coloring found
7. Return {M^k : k = 1, 2, ..., m' - 1}
```

LEMMA 11.3.– *If $M = \{M_1, M_2, M_3\}$ with $w_1 \geqslant w_2 \geqslant w_3$ is an edge coloring of G', then algorithm SOL3 produces a solution M^k satisfying $W^k \leqslant 2 \cdot w_1 + w_3$.*

Proof. As previously, consider one particular iteration of algorithm SOL3. In this lemma, consider the case where $k = i_3 - 1$. By construction, it holds that $M_3 \subseteq \overline{E'_{i_3-1}}$ and M_3 contains a matching where each vertex of $G'[\overline{E'_{i_3-1}}]$ having a degree 3 in G' is saturated. Thus, G'_{2,i_3-1} exists. Moreover, since $\forall e \in \overline{E'_{i_3-1}}, w(e) \leqslant w(e_{i_3}) = w_3$, it is obtained that:

$$w_3^{i_3-1} \leqslant w_3 \qquad [11.5]$$

As previously, it is deduced that G'_{1,i_3-1} can be edge colored with 2 colors $M_1^{i_3-1}$ and $M_2^{i_3-1}$ and it holds that:

$$w_1^{i_3-1} + w_2^{i_3-1} \leqslant 2 \cdot w_1 \qquad [11.6]$$

Combining [11.5] and [11.6], it is obtained that:

$$W^k \leqslant W^{i_3-1} \leqslant 2 \cdot w_1 + w_3 \quad \blacksquare$$

REMARK 11.1.– Observe that if a color $M_j^{i_3-1}$ is empty, then the bound for algorithm SOL3 can be improved: in this case, $W^{i_3-1} \leqslant 2 \cdot w_1$. This is also valid for algorithms SOL1 and SOL2 and if several colors are empty. For algorithm SOL1 for instance, if $M_2^{i_3-1}$ and $M_2^{i_3-1}$ are empty, then $W^{i_3-1} \leqslant w_1 + w_3$.

THEOREM 11.6.– *Algorithm HYBRID_MEC produces a $\frac{7}{6}$ approximation for the MEC problem in bipartite graphs with maximum degree* 3.

Proof. Let $\{M_1^*, M_2^*, \ldots, M_5^*\}$ with $w_1^* \geqslant w_2^* \ldots \geqslant w_5^*$ be an optimal solution for the MEC problem for the graph G. Denote by i_k^* the smallest index of an edge in M_k^* ($i_k^* = m + 1$ if the color is empty).

Consider the iteration of algorithm HYBRID_MEC corresponding to the cases $i = i_5^* - 1$ and $j = i_4^* - 1$. Then:

– by applying Lemma 11.1, an edge coloring of weight at most $w_1^* + w_2^* + 2 \cdot w_3^*$ is produced on $G' = G[E_i]$. Then, the greedy coloring of the edges of $\overline{E_i}$ produces a coloring of weight at most:

$$w_1^* + w_2^* + 2 \cdot w_3^* + w_5^* \qquad [11.7]$$

– by applying Lemma 11.2, an edge coloring of weight at most $w_1^* + 2 \cdot w_2^*$ is produced on $G' = G[E_j]$. Then, the greedy coloring of the edges of $\overline{E_j}$ produces a coloring of weight at most:

$$w_1^* + 2 \cdot w_2^* + 2 \cdot w_4^* \qquad [11.8]$$

– by applying Lemma 11.3, an edge coloring of weight at most $2 \cdot w_1^* + w_3^*$ is produced on $G' = G[E_j]$. Then, the greedy coloring of the edges of $\overline{E_j}$ produces a coloring of weight at most:

$$2 \cdot w_1^* + w_3^* + 2 \cdot w_4^* \qquad [11.9]$$

Note that if there is an empty color or several empty colors produced by one of the algorithms SOL1, SOL2, SOL3, then the bounds are still valid. Indeed, for algorithm SOL3 for instance, according to Remark 11.1, the value of the coloring calculated at

step $j = i_3 - 1$ has a weight at most $2 \cdot w_1^*$, and the greedy step produces a coloring of value at most $2 \cdot w_1^* + 3 \cdot w_4^* \leqslant 2 \cdot w_1^* + w_3^* + 2 \cdot w_4^*$.

By using [11.7], [11.8] and [11.9], it is deduced that the coloring M calculated by algorithm HYBRID_MEC satisfies:

$$W \leqslant \min\{w_1^* + w_2^* + 2 \cdot w_3^* + w_5^*, w_1^* + 2 \cdot w_2^* + 2 \cdot w_4^*, 2 \cdot w_1^* + w_3^* + 2 \cdot w_4^*\} \quad [11.10]$$

The convex combination of these 3 values with coefficients $\frac{3}{6}$, $\frac{2}{6}$ and $\frac{1}{6}$ respectively and the inequality [11.10] give the expected result, i.e.:

$W \leqslant \frac{7}{6}w_1^* + \frac{7}{6}w_2^* + \frac{7}{6}w_3^* + w_4^* + \frac{1}{2}w_5^* \leqslant \frac{7}{6}\,\mathrm{opt}$ ∎

11.7.3. *An approximation algorithm for* $\Delta \leqslant 7$

In [LUC 07] is presented an algorithm, denoted by $\mathcal{A}(G)$, that improves the 2 approximation ratio given in [KES 04] for the MEC problem in bipartite graphs of maximum degree $\Delta \leqslant 7$. The main idea of this algorithm is the following: for a given bipartite graph G, of maximum degree Δ, run $\Delta - 1$ algorithms, $A_\Delta, A_{\Delta+1}, \ldots, A_{2\Delta-1}$ and select the best solution found. Each $A_{\Delta+k}$, $0 \leqslant k \leqslant \Delta - 1$, algorithm splits the graph G into two subgraphs G_1 and G_2 such that the graph G_1 contains heavy edges and has maximum degree $\Delta(G_1) \leqslant k$. Note that, in general, $\Delta(G_2) \leqslant \Delta(G)$. Given this splitting of G, each algorithm $A_{\Delta+k}$ returns a solution of cost $W_{\Delta+k}$, by concatenating the following solutions of the MEC problem on G_1 and G_2: (i) for the graph G_2 the algorithm finds a Δ-matchings solution by solving the traditional edge coloring problem on G_2, (ii) for the graph G_1 the algorithm finds a solution of at most $2\Delta(G_1) - 1$ matchings using (recursively) algorithm $\mathcal{A}(G_1)$.

For $k = 0$, G_2 coincides with G and therefore algorithm A_Δ returns a Δ-matchings solution found as in point (i) above. For the weight of such a solution it holds that $W_\Delta \leqslant \Delta \cdot w_1^*$, since for the weight, w_i, of any matching of this solution it holds that $w_i \leqslant w(e_1) = w_1^*$.

For $k = 1$, G_1 is a maximal matching of G created by examining its edges in non-increasing order of their weights. This matching is of weight $w_1 = w(e_1)$. Algorithm $A_{\Delta+1}$ also uses algorithm $A_\Delta(G_2)$. The cost of the solution that algorithm $A_{\Delta+1}$ returns is $W_{\Delta+1} \leqslant w_1^* + \Delta \cdot w_2^*$, since no edge e of weight $w(e) > w_2^*$ belongs to G_2 (if such an edge belongs to G_2, then it is not in M_1 because a heavier one of its adjacent edges is in M_1, which is a contradiction).

In general, algorithm $A_{\Delta+k}, k \geqslant 2$, repeatedly splits the graph G into graphs G_1 and G_2, with G_1 containing edges which are heavier than a parameter c, taking as values the weights of the edges of the graph G. The algorithm returns the best of the solutions found in these iterations.

- $A_{\Delta+k}$
1. *For each weight* $c = w(e_1), w(e_2), \ldots$ *do*
2. *Split* G *into edge induced subgraphs:*
   - $G_1 = \{e | w(e) > c\}$ *and*
   - $G_2 = \{e | w(e) \leqslant c\}$
3. *If* $\Delta(G_1) \leqslant k$ *then*
4. *If* $\Delta(G_1) = 2$ *then find an optimal solution for the graph* G_1
5. *Else run* $\mathcal{A}(G_1)$
6. *Use* A_Δ *for* G_2
7. *Return the best solution found*

The case of $k = 2$ is analyzed in a different way than the cases of $k \geqslant 3$, since in this case $\Delta(G_1) = 2$ and an optimal solution for G_1 can be found by Theorem 11.3. Thus, for $k = 2$ we obtain:

LEMMA 11.4.– *Algorithm $A_{\Delta+2}$ returns a solution of cost $W_{\Delta+2} \leqslant w_1^* + w_2^* + \Delta \cdot w_3^*$.*

Proof. $A_{\Delta+2}$ returns the best solution found with weight $W_{\Delta+2} = \min\{w_1 + w_2 + w_3 + \Delta \cdot w_4\}$ for different values of $c = w_4$. Consider the cost of the solution found by the algorithm in the iteration for which $w_4 = w_3^*$. In this iteration the edges of G_1 are a subset of the edges in the two heaviest matchings of the optimal solution. Therefore for this iteration it holds that $W_{\Delta+2} \leqslant w_1^* + w_2^* + \Delta \cdot w_3^*$, since $\Delta(G_1) = 2$ and thus the optimal solution for G_1 can be found by Theorem 11.3. ∎

Using algorithms A_Δ, $A_{\Delta+1}$ and $A_{\Delta+2}$, algorithm $\mathcal{A}(G)$ returns a solution of cost $W = \min\{W_\Delta, W_{\Delta+1}, W_{\Delta+2}\}$, i.e.:

$$\begin{aligned} W &\leqslant \Delta \cdot w_1^* \\ W &\leqslant w_1^* + \Delta \cdot w_2^* \\ W &\leqslant w_1^* + w_2^* + \Delta \cdot w_3^* \end{aligned}$$

By multiplying these inequalities by z_1, z_2 and z_3, respectively, and adding them we obtain:

$$W \cdot (z_1 + z_2 + z_3) \leqslant w_1^* \cdot (z_1 \cdot \Delta + z_2 + z_3) + w_2^* \cdot (z_2 \cdot \Delta + z_3) + w_3^* \cdot (z_3 \cdot \Delta)$$

$$\frac{W}{\text{opt}} \leqslant \frac{t}{z_1 + z_2 + z_3}$$

where $t = z_1 \cdot \Delta + z_2 + z_3 = z_2 \cdot \Delta + z_3 = z_3 \cdot \Delta$. Thus, the best approximation ratio is achieved when the quantity $\frac{t}{z_1+z_2+z_3}$ is minimized. Therefore, the object is to:

$$\min\left\{\frac{t}{z_1+z_2+z_3}\right\}$$

such that

$$\begin{aligned}
&(z_1 \cdot \Delta + z_2 + z_3) = t\\
&(z_2 \cdot \Delta + z_3) = t\\
&(z_3 \cdot \Delta) = t\\
&z_1, z_2, z_3 \geqslant 0
\end{aligned}$$

By solving this problem, a new approximation ratio for the MEC problem in bipartite graphs with $\Delta \geqslant 3$ is obtained, which is:

$$\frac{W}{\text{opt}} \leqslant \frac{\Delta^3}{3\Delta^2 - 3\Delta + 1} = \varrho_{\Delta+2}$$

This ratio is unbounded, but gives better results than the approximation ratio $(2\Delta - 1)/3$ given in [DEM 02], for any Δ, as well as than the 2-approximation algorithm given in [KES 04], for $\Delta = 3$ and $\Delta = 4$. In fact, $\varrho_{\Delta+2}$ becomes $27/19 \simeq 1.42$ for $\Delta = 3$ and $64/37 \simeq 1.73$ for $\Delta = 4$. Note, however, that an 1.17-approximation algorithm is known [WER 04] for bipartite graphs with $\Delta = 3$.

The behavior of algorithm $\mathcal{A}(G)$ for bipartite graphs of $\Delta \geqslant 4$, is improved if we allow the use of algorithms $A_{\Delta+3}$, $A_{\Delta+4}$ and so on. In fact, for $k = 3$ we have:

LEMMA 11.5.– *Algorithm $A_{\Delta+3}$ returns a solution of cost $W_{\Delta+3} \leqslant \varrho_{\Delta+2} \cdot (w_1^* + w_2^* + w_3^*) + \Delta \cdot w_4^*$.*

Proof. Similarly to the proof of Lemma 11.4, in some iteration of algorithm $A_{\Delta+3}$, the maximum edge in G_2 is equal to w_4^* and the edges of G_1 are a subset of the edges in the three heaviest matchings of the optimal solution. As $\Delta(G_1) = 3$, the algorithm $\mathcal{A}(G_1)$ returns a $\varrho_{\Delta+2}$ approximate solution for the graph G_1. ∎

Working as above, algorithm $\mathcal{A}(G)$, for bipartite graphs with $\Delta \geqslant 4$, leads to an approximation ratio:

$$\varrho_{\Delta+3} = \frac{19\Delta^4}{76\Delta^3 - 138\Delta^2 + 100\Delta - 27}$$

This ratio improves the result for $\Delta = 4$, where the ratio becomes 1.61 from 1.73. Moreover, for $\Delta = 5$ this ratio becomes 1.82.

In general, the algorithm $\mathcal{A}(G)$ gives a better approximation ratio than 2 for bipartite graphs with $\Delta(G) \leqslant 7$. An explicit formula for the ratios achieved by algorithm $\mathcal{A}(G)$ for any Δ can be found in [ESC 06]. The following table summarizes the best approximation ratio achieved by this algorithm and the previous best known ratio for different values of Δ.

Δ	3	4	5	6	7	8
our ratio	1.42	1.61	1.75	1.86	1.95	2.03
previous ratio	1.17	2	2	2	2	2

In general, the complexity of algorithm $\mathcal{A}(G)$ is dominated by the complexity of $A_{2\Delta(G)-1}$, which recursively runs at most $|E|$ times algorithm $\mathcal{A}(G_1)$, where $\Delta(G_1) \leqslant \Delta(G) - 1$. The recursion depth is at most $\Delta - 2$, since the recursion stops in $A_{\Delta+2}$. Each iteration of each $A_{\Delta+k}$ algorithm, $0 \leqslant k \leqslant \Delta - 1$, runs algorithm A_Δ, which runs in polynomial time. Thus, algorithm A_Δ is called $O(|E|^{\Delta(G)-2})$ times, in total, by algorithm $\mathcal{A}(G)$.

REMARK 11.2.– Concerning general graphs, note that an edge coloring using $(\Delta + 1)$ matchings can be found in polynomial time. Thus, by modifying algorithm A_Δ to find such a coloring of $\Delta + 1$ (instead of Δ) matchings, algorithm $\mathcal{A}(G)$ works also for general graphs. Furthermore, it beats the 2-approximation algorithm in [KES 04] for graphs of $\Delta = 3$ and $\Delta = 4$, achieving ratios 1.73 and 1.93, respectively.

11.8. Trees

As the complexity of the MEC problem on trees remains open, in this section is presented first a polynomial algorithm for a related decision problem called Feasible k-Coloring, given in [LUC 07]. This algorithm is then used to derive a polynomial algorithm for the MEC problem on trees of bounded degree. Then, a polynomial algorithm for stars of chains is presented [LUC 07].

11.8.1. *Feasible k-Coloring and bounded degree trees*

The Feasible k-Coloring problem is formally defined as follows. An analog problem is also defined and solved in [PEM 05] for the MVC problem on trees.

Feasible k-Coloring:
Instance: A tree $T(V, E)$, a weight function $w(e) : E \to \mathbb{N}$ and a sequence of k

integer weights $a_1, a_2, \ldots, a_k$, such that $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_k$.
Question: Does there exist a partition of E into exactly k matchings M_1, M_2, ..., M_k, such that $w_j \leqslant a_j,\ 1 \leqslant j \leqslant k$?

Consider the tree T rooted at an arbitrary vertex, r. For each edge $e = (v, u)$ we define u to be the most distant from r endpoint of e and $T(e)$ to be the subtree of T rooted at u. Denote by $S(e) \subseteq \{M_1, M_2, \ldots, M_k\}$ to be the set of matchings in which edge e can belong in order the subtree $T(e) \cup \{e\}$ to be feasibly colorable.

The algorithm presented in [LUC 07] initializes the sets $S(e)$ for each leaf edge e to contain the matchings that are heavier than its weight $w(e)$. Moreover, a fictitious edge e_0 of weight $w(e_0) = 0$ is connected to the root of the tree, as in Figure 11.3a, in order to treat the root of the tree as the remaining vertices. The feasible k-coloring algorithm, denoted by FKC, follows.

```
 - FKC
1. Initialization:
     Leaf edges: S(e) = {M_j | 1 ⩽ j ⩽ k and w(e) ⩽ a_j}
     Rest edges: S(e) = {}
     Add a fictitious vertex r',
     a fictitious edge e_0 = (r', r) with w(e_0) = 0
     and a fictitious matching M_0 with w_0 = a_0 = 0
2. For each e ∈ E ∪ {e_0} in post-order do
3.   For each matching M_j such that w(e) ⩽ a_j do
4.     If there is coloring of T(e) ∪ {e} such that e ∈ M_j then
5.       S(e) = S(e) ∪ {M_j}
6.   If S(e) = { } then return
7. Create a feasible coloring using the S(e)'s
```

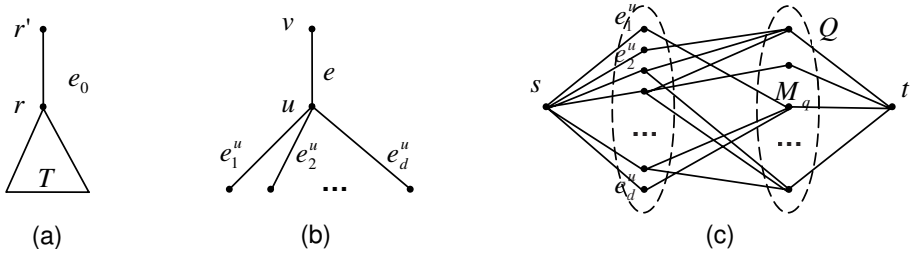


Figure 11.3. *(a) The fictitious edge e_0; (b) an instance of the graph in line 4 of algorithm FKC; (c) the flow network constructed in line 4 of algorithm FKC. All edges have weight equal to 1*

In line 4, algorithm FKC decides if a feasible coloring for the subtree $T(e) \cup \{e\}$ exists (see Figure 11.3b). For each edge $e = (v, u)$, we define $E_u = \{e_1^u, e_2^u, \ldots, e_d^u\}$ to be the set of edges from u to its children (recall that u is the most distant from the root of the tree endpoint of e). Each edge e_i^u can belong in one of the matchings in its $S(e_i^u)$. Let Q be the union of the sets $S(e_i^u)$, but the matching M_j edge e is assigned to, i.e. $Q = \bigcup_{i=1}^{d} S(e_i^u) - \{M_j\}$. A bipartite graph $B(E_u, Q; S)$ is created, where there is an edge between $e_i^u \in E_u$ and $M_q \in Q$ iff $M_q \in S(e_i^u)$. Then, a flow network F is created by joining a source vertex s to each vertex in E_u and a terminal vertex t to each vertex in Q, as in Figure 11.3c. A weight equal to 1 is assigned to all the edges in F. Then, it follows that there is coloring of $T(e) \cup \{e\}$ such that $e \in M_j$ iff there is in F an $s - t$ flow of value d.

In line 7, algorithm FKC creates a partition of the edges of T into matchings. This can be done by considering the edges of T in pre-order and assigning an edge e in an arbitrary matching in its set $S(e)$.

Algorithm FKC performs $k \cdot |E|$ iterations and in each one of them runs a maximum flow algorithm of O(poly) time. Thus, the next theorem follows.

THEOREM 11.7.– *There is a polynomial time algorithm for the Feasible k-Coloring problem.*

Algorithm FKC can be used to solve the MEC problem on trees, as following.

```
 - BOUNDED_DEGREE
1. For k = Δ to Δ' − 1 do
2.   For all (|E| choose k) combinations of edge weights,
     such that w_1 ⩾ w_2 ⩾ ... ⩾ w_k and w_1 = max{w(e)|e ∈ E}) do
3.     Run algorithm FKC
4. Return the best of the solutions found
```

Line 3 of algorithm BOUNDED_DEGREE is repeated $O(\Delta \cdot |E|^{\Delta})$ times, and therefore it is polynomial only for trees of polynomially bounded degree.

THEOREM 11.8.– *There is a polynomial time algorithm for the MEC problem in trees with polynomially bounded degree.*

11.8.2. *Stars of chains*

A star consists of m edges $e_1, e_2, \ldots, e_m$ sharing a common endpoint. Obviously, the optimal solution to the MEC problem for such a weighted star is of cost $\text{opt} = \sum_{i=1}^{m} w(e_i)$ and consists of exactly $\Delta = m$ matchings.

A star of chains consists of p chains $C_1, C_2, \ldots, C_p$ all starting from a common vertex, for example u. Consider each chain $C_i,\ 1 \leqslant i \leqslant p$, starting from u with an edge e_i^u which we call start edge. Assume, also, w.l.o.g. that $w(e_1^u) \geqslant w(e_2^u) \geqslant \ldots \geqslant w(e_p^u)$. In this section we present a polynomial algorithm for the MEC problem in stars of chains, given in [LUC 07].

LEMMA 11.6.– *For an optimal solution of the MEC problem on a star of chains the following hold:*
i) The number of matchings s^* is equal to either p or $p+1$.
ii) Only $k \leqslant 3$ matchings have cardinality $|M_j| > 1$.
iii) At least the $k-1$ heaviest start edges appear in these k matchings.

Proof. i) According to Proposition 11.1, $\Delta \leqslant s^* \leqslant \Delta' - 1$. Here, $\Delta = p$ and $\Delta' = p+2$.

ii) Assume that an optimal solution has more than three matchings of cardinality $|M_j| > 1$. Consider those matchings sorted in non-increasing order of their weights. Each non-start edge e has at most 2 neighboring edges. So, such an edge e can be moved in one of the three first heaviest matchings, since its neighboring edges can belong in at most two different matchings.

iii) Consider first that $k = 2$. Assume that in the optimal solution the heaviest start edge e_1^u does not belong to either of two matchings of cardinality $|M_j| > 1$. Then e_1^u can be either inserted in one of those two matchings (if this does not contain another start edge) or e_1^u can replace an existing start edge. In both cases the cost of the optimal solution decreases or remains the same.

Assume next that $k = 3$. As in the previous case, e_1^u can be inserted in one of the three matchings of cardinality $|M_j| > 1$. Similarly, e_2^u can be inserted in one of the remaining two of those matchings. ∎

The following two cases which depend on the possible number of matchings in an optimal solution, i.e. $p+1$ or p, are distinguished.

If an optimal solution consists of $p+1$ matchings, then it contains exactly one matching without any start edge. Algorithm `STAR`, given in what follows, finds such an optimal schedule with $p+1$ matchings.

Note that it is possible that algorithm `STAR` returns $p-1$ matchings of $|M_i| = 1$, in the case where one of the three matchings of $S^*(H)$ found in line 2 consists of a single edge. Taking into account Lemma 11.6, it follows that algorithm `STAR` returns the optimal solution of $p+1$ matchings since: (i) the $p-(k-1)$ matchings of cardinality $|M_i| = 1$ contain the $p-(k-1)$ lightest start edges (one per each matching) and (ii) the cost of k matchings is optimal.

```
 - STAR
1. Remove from the star the p − 2 lightest start edges
   (this creates a graph H consisting of p − 1 chains)
2. Find an optimal solution S*(H) for the graph H,
   using Theorem 11.3
3. If there are 3 non-empty matchings in S*(H) then
4.   Return the solution consisting of these 3 matchings of
S*(H)
     plus p − 2 matchings each one containing one of
     the removed p − 2 lightest start edges
```

The complexity of algorithm STAR is dominated by line 2 and by Theorem 11.3. Therefore, it is $O(|E|^2)$.

If an optimal solution consists of p matchings, then each of them contains a start edge. Algorithm STAR2 returns such an optimal schedule with p matchings.

```
 - STAR2
1. For i = 3 to p do
2.   Remove p − 3 start edges e_3^u, e_4^u, ..., e_{i−1}^u, e_{i+1}^u, ..., e_p^u
     (this creates a star T of 3 chains and a graph H
     of p − 3 chains)
3.   Find the optimal solution S*(T) using Theorem 11.8
4.   If there are exactly 3 matchings in S*(T) then
5.     Find the optimal solution S*(H) using Theorem 11.3
6.     Combine the solutions S*(T) and S*(H) into exactly
       3 matchings
7.     Find a solution for the initial star consisting of these
       3 matchings plus p − 3 matchings each one containing
       one of the removed p − 3 start edges
8. Return the best solution found
```

Algorithm BOUNDED_DEGREE is used in line 3 since T is a bounded degree tree with $\Delta = 3$ and it returns an optimal solution $S^*(T)$ of at least three matchings. In line 6, a 3-matchings optimal solution for the edges in T and H can be obtained by considering the matchings in both solutions in non-increasing order of their weights and merging the matchings of each solution having the same rank, since T and H are vertex-disjoint. The optimality of the solution calculated by algorithm STAR2 follows from Lemma 11.6 using the same arguments as for algorithm STAR.

The complexity of the algorithm is dominated by line 3 which takes polynomial time (see section 11.8.1) and is executed $\Delta - 2$ times.

The optimal solution to the MEC problem on stars of chains is the best between the solution found by algorithm STAR (optimal with $p + 1$ matchings) and that found by algorithm STAR2 (optimal with p matchings), since, according to Lemma 11.6, the optimal solution consists of either $p + 1$ or p matchings. Thus, the following theorem holds.

THEOREM 11.9.– *The MEC problem on stars of chains can be solved optimally in polynomial time.*

11.9. Conclusions

The recent activity on the MEC problem has led to significant progress in understanding its complexity and approximability though we are still far from resolving the issue. Many open questions can be picked up by looking at the entries of Table 11.1. These questions can be classified into two directions:

(i) further identification of the boundary between polynomial and **NP**-hard variants of the problems with respect to the underlying graph. For example, the complexity of the MEC problem on trees is an open question. Moreover, we do not know anything about the complexity of the MEC problem on interval graphs. However, the MEC problem on interval graphs includes the question for complete graphs as well as the open question about the complexity of the edge coloring problem on interval graphs of even maximum degree [BOJ 01];

(ii) decrease (or closing) the gap between the best known approximation ratios and inapproximability results, either by obtaining new approximation algorithms of lower approximation ratios or by showing higher inapproximability results. For example, a 2-approximation algorithm and a 7/6-inapproximability result is known for the MEC problem on bipartite graphs. It would be nice to decrease this gap by improving either of these results.

11.10. Bibliography

[AFR 05] AFRATI F. N., ASLANIDIS T., BAMPIS E., MILIS I., "Scheduling in Switching Networks with Set-Up Delays", *Journal of Combinatorial Optimization*, vol. 9, num. 1, p. 49–57, 2005.

[BOJ 01] BOJARSHINOV V. A., "Edge and total coloring of interval graphs", *Discrete Applied Mathematics*, vol. 114, p. 23–28, 2001.

[BON 81] BONGIOVANNI G., COPPERSMITH D., WONG C. K., "An Optimum Time Slot Assignment Algorithm for an SS/TDMA System with Variable Number of Transponders", *IEEE Transactions On Communications*, vol. 29, num. 5, p. 721–726, 1981.

[BRU 98] BRUCKER P., GLADKY A., HOOGEVEEN H., KOYALYOV M., POTTS C., TAUTENHAM T., VAN DE VELDE S., "Scheduling a Batching Machine", *Journal of Scheduling*, vol. 1, num. 1, p. 31–54, 1998.

[CRE 01] CRESCENZI P., DENG X., PAPADIMITRIOU C. H., "On Approximating a Scheduling Problem", *Journal of Combinatorial Optimization*, vol. 5, num. 3, p. 287–297, 2001.

[DEM 02] DEMANGE M., DE WERRA D., MONNOT J., PASCHOS V. T., "Weighted Node Coloring: When Stable Sets Are Expensive", *Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, p. 114–125, 2002.

[ESC 06] ESCOFFIER B., MONNOT J., PASCHOS V. T., "Weighted Coloring: further complexity and approximability results", *Information Processing Letters*, vol. 97, num. 3, p. 98–103, 2006.

[FIN 04] FINKE G., JOST V., QUEYRANNE M., SEBŐ A., Batch processing with interval graph compatibilities between tasks, Report , Cahiers du laboratoire Leibniz; available at http://www-leibniz.imag.fr/NEWLEIBNIZ/LesCahiers/index.xhtml, 2004.

[GOP 85] GOPAL I. S., WONG C., "Minimizing the Number of Switchings in a SS/TDMA System", *IEEE Transactions On Communications*, vol. 33, num. 6, p. 497–501, 1985.

[HOL 81] HOLYER I., "The NP-Completeness of Edge-Coloring", *SIAM Journal on Computing*, vol. 10, num. 4, p. 718–720, 1981.

[INU 79] INUKAI T., "An Efficient SS/TDMA Time Slot Assignment Algorithm", *IEEE Transactions On Communications*, vol. 27, num. 10, p. 1449–1455, 1979.

[KES 04] KESSELMAN A., KOGAN K., "Non-preemptive scheduling of optical switches", *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 3, p. 1840–1844, 2004.

[KON 16] KONIG D., "Uber Graphen und ihre Anwendung auf Determinantentheorie und Mengenlehre", *Mathematische Annalen*, vol. 77, p. 453–465, 1916.

[LAW 78] LAWLER E. L., LABETOULLE J., "On Preemptive scheduling of unrelated parallel processors by linear programming", *Journal of the Association for Computing Machinery*, vol. 25, num. 4, p. 612–619, 1978.

[LUC 07] LUCARELLI G., MILIS I., PASCHOS V. T., On the max-weight edge coloring problem, Report num. 248, Cahiers du laboratoire LAMSADE; available at http://www.lamsade.dauphine.fr/cahiers.html, 2007.

[MAR 05] MARX D., "NP-completeness of list coloring and precoloring extension on the edges of planar graphs", *Journal of Graph Theory*, vol. 49, num. 4, p. 313–324, 2005.

[PEM 04] PEMMARAJU S. V., RAMAN R., VARADARAJAN K. R., "Buffer minimization using max-coloring", *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, p. 562–571, 2004.

[PEM 05] PEMMARAJU S. V., RAMAN R., "Approximation Algorithms for the Max-coloring Problem", *International Colloquium on Automata, Languages and Programming (ICALP)*, p. 1064–1075, 2005.

[PRA 00] PRAIS M., RIBEIRO C. C., "Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment", *INFORMS Journal on Computing*, vol. 12, num. 3, p. 164–176, 2000.

[REN 85] RENDL F., "On the complexity of decomposing matrices arising in satellite communication", *Operations Research Letters*, vol. 4, num. 1, p. 5–8, 1985.

[RIB 89] RIBEIRO C. C., MINOUX M., PENNA M. C., "An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment", *European Journal of Operational Research*, vol. 41, p. 232–239, 1989.

[VIZ 64] VIZING V. G., "On an estimate of the chromatic class of a p-graph", *Diskret. Analiz.*, vol. 3, p. 25–30, 1964.

[WER 04] DE WERRA D., DEMANGE M., ESCOFFIER B., MONNOT J., PASCHOS V. T., "Weighted Coloring on Planar, Bipartite and Split Graphs: Complexity and Improved Approximation", *International Symposium on Algorithms and Computation (ISAAC)*, p. 896–907, 2004.

[YU 06] YU C. W., "On the Complexity of the Max-Edge-Coloring Problem with Its Variant", *Proceedings of the 23rd Workshop on Combinatorial Mathematics and Computation Theory*, p. 321–330, 2006.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 12

An Extensive Comparison of 0-1 Linear Programs for the Daily Satellite Mission Planning

12.1. Introduction

The mission planning problem of a low Earth orbit satellite is the problem of choosing a set of photographs to be taken by the satellite during a planning horizon in order to satisfy a maximal part of the requested images. The mission planning problem of SPOT5 satellite has been the subject of numerous studies [BEN 96, BEN 99, VAS 01, GAB 03, VAS 03]. Different approximate methods [BEN 96, VAS 01] have been proposed and applied to the set of instances proposed in [BEN 99]. To assess the quality of the obtained approximate solutions, we need to calculate tight upper bounds: in [BEN 96], the planning problem is formulated like a 0-1 linear program; we have proposed in [GAB 03] another 0-1 linear formulation, using decomposition and flow formulation. In [VAS 03], a specific algorithm is proposed to calculate very good upper bounds. In this chapter, we explain why our 0-1 linear formulation presents a much smaller integrality gap than the formulation proposed in [BEN 96]. We calculate upper bounds for all the instances proposed in [BEN 99]. These bounds are greater than those found in [VAS 03] but our results can easily be strengthened with some additional improvements.

Chapter written by Virginie GABREL.

12.2. Different formulations for the daily satellite mission planning problem

12.2.1. *The daily satellite mission planning problem*

The SPOT satellite constitutes a family of Earth observation satellites, which are developped by the CNES (Centre National d'Etudes Spatiales).

SPOT5 was launched in 2002. It is equipped with three viewing instruments (front, middle and rear) which can take two types of images: the monoimages need one of the three instruments and the stereo images need the front and the rear instruments.

The daily satellite mission planning problem of the SPOT5 satellite consists of deciding each day which images will be taken the next day in order to satisfy a maximal number of clients.

The problem can be described as follows:

– The set of candidate images is known.

– A weight is associated with each image.

– The starting time and the completion time for taking each image are known.

– On an instrument, the main constraints concern the non-overlapping and the respect of a minimal transition time between two successive images.

– The total number of images is limited (due to limitations of the on-board recording capacity).

– Some images cannot be taken simultaneously on different instruments (this is due to limitation of the instantaneous data flow through the satellite telemetry).

Each day, a set of images to be taken must be chosen in order to maximize the sum of the weights of selected images and to satisfy all constraints.

12.2.2. *The "natural" model*

In [BEN 96], a 0-1 linear program is proposed to modelize the daily mission planning problem of SPOT5. The model is quite natural: each image i ($i = 1, \ldots, n$) that can be taken by an instrument c (with $c \in \{1, 2, 3\}$) is represented in the model by a 0-1 variable x_i^c.

The proposed linear program, denoted by $\mathrm{S5_N}$, is:

$$(\mathrm{S5_N}) \begin{cases} \max \sum_{(i,c)\in S} w_i x_i^c & & \\ \sum_{c:(i,c)\in S} x_i^c \leqslant 1 & \forall i \in M & (1_\mathrm{N}) \\ x_i^1 - x_i^3 = 0 & \forall i \in T & (2_\mathrm{N}) \\ x_i^c + x_j^c \leqslant 1 & \forall \{(i,c),(j,c)\} \in \Omega & (3_\mathrm{N}) \\ \sum_{(i,c)\in \Lambda^k} x_i^c \leqslant \delta^k & k = 1,\ldots,K & (4_\mathrm{N}) \\ \sum_{(i,c)\in S} z_i^c x_i^c \leqslant z & & (5_\mathrm{N}) \\ x_i^c \in \{0,1\} & \forall (i,c) \in S & \end{cases}$$

where

– S denotes the set of all feasible couples (i,c), and w_i the weight of the image i,

– linear inequalities (1_N) correspond to the fact that each monoimage i (M denotes the monoset) must be taken by at most one camera,

– linear equalities (2_N) correspond to the fact that each stereo image i (T denotes the stereo set) must either be simultaneously taken by the front camera (numbered by 1) and the rear one (numbered by 3) or not be taken,

– linear inequalities (3_N) correspond to constraints induced by non-overlapping and minimal transition time between two images taken on a same instrument, where Ω denotes the set of two couples $\{(i,c),(j,c)\}$ linked by these constraints,

– constraints (4_N) correspond to the constraints induced by limitation between several instruments, where Λ^k denotes the set of two or three couples linked by constraint k and δ^k equals $\mid \Lambda^k \mid -1$,

– constraint (5_N) corresponds the limitation on recording capacity where z_i^c is the memory size necessary for recording on board image i taken by camera c and z the total memory size available.

The optimal solution value of the linear relaxation of $\mathrm{S5_N}$, denoted by $v(\overline{\mathrm{S5}}_\mathrm{N})$, is an obvious upper bound for $v(\mathrm{S5_N})$. But, as shown in section 12.4, the integrality gap between $\mathrm{S5_N}$ and $\overline{\mathrm{S5}}_\mathrm{N}$ is huge. Thus, we propose another strengthened 0-1 linear formulation.

12.2.3. *The flow formulation*

In [GAB 03], we propose another model based on a decomposition of the initial problem into three subproblems, one per camera. Sub-problem on camera c ($c = 1, 2, 3$) concerns the choice of images to be taken by camera c considering only

constraints related to non-overlapping and respect of a minimal transition time between two successive photographs. We denote by S^c the set of images that can be taken by camera c and we define two binary relations on $S^c \times S^c$.

DEFINITION 12.1.– *For $(i,j) \in S^c \times S^c$, (i,j) belongs to the incompatibility relation, denoted by $\mathcal{I}^c$, if and only if i and j cannot be taken together because they use the camera c at the same moment or the transition time between the two shots is not sufficient.*

Let us note that constraints 3_{N} in $\mathrm{S5_N}$ describe the incompatibility relation on each set S^c.

DEFINITION 12.2.– *For $(i,j) \in S^c \times S^c$, (i,j) belongs to the enchainability relation, denoted by $\mathcal{E}^c$, if and only if j is taken after i with the instrument c.*

The incompatibility relation is symmetric while the enchainability relation is clearly antisymmetric. Moreover, the enchainability relation is not necessarily transitive (it depends on satellite capacities).

Considering the enchainability relation, the sub-problem of choosing a set of images to be taken by an instrument c can be represented by a directed graph, namely the enchainability graph denoted by $G^c_{\mathcal{E}} = (V^c, E^c)$, which is defined as follows:

– V^c represents the set S^c (with $\mid S^c \mid = n^c$) of images that can be taken by camera c (with the following convention: a couple (i,c) is represented by a vertex numbered by i in V^c), plus two fictitious vertices numbered by 0 and n^c+1 respectively the root and the anti-root of $G^c_{\mathcal{E}}$,

– each vertex i in V^c is associated with the weight w_i of image i (w_0 and w_{n^c+1} are equal to 0),

– E^c represents the enchainability relation defined on $S^c \times S^c$, plus arcs $(0,i)$ and (i, n^c+1) for all $i = 1, \ldots, n^c$, and $(0, n^c+1)$.

It appears that $G^c_{\mathcal{E}}$ is circuit-free since any arbitrary circuit $(i_1, i_2, \ldots, i_q, i_1)$ in $G^c_{\mathcal{E}}$ leads to the following contradiction: i_1 is taken before i_k and i_k is taken before i_1.

Any feasible sequence of images taken on camera c can be represented by vertices belonging to a path $\mu[0, n^c+1] = \{i_1 = 0, i_2, \ldots, i_q = n^c+1\}$ from 0 to n^c+1 in $G^c_{\mathcal{E}}$. Respectively, any path μ from 0 to n^c+1 in $G^c_{\mathcal{E}}$ represents a sequence of images taken on camera c. This sequence respects all the constraints related to non-overlapping and respect of a minimal transition time between images if and only if $\mathcal{E}^c$ is transitive, otherwise some of these binary constraints may be violated. Indeed, if $\mathcal{E}^c$ is transitive, all vertices i_j and i_{j+k} belonging to μ, with k a strictly positive integer, are such that $(i_j, i_{j+k}) \in \mathcal{E}^c$. Otherwise, there may be some vertices i_j and i_{j+k} belonging to μ (with $k \geqslant 2$) such that $(i_j, i_{j+k}) \notin \mathcal{E}^c$. Consequently, associated

constraints $x^c_{i_j} + x^c_{i_{j+k}} \leqslant 1$ are not respected by the sequence of images corresponding to μ. In fact, the part of $3_{\rm N}$ constraints excluded from $G^c_{\mathcal{E}}$ is represented by arcs not in E^c but belonging to the transitivity closure of $G^c_{\mathcal{E}}$, which is denoted by $\hat{G}^c_{\mathcal{E}} = (V^c, \hat{E}^c)$ (obviously $\hat{E}^c = E^c$ when $G^c_{\mathcal{E}}$ is transitive).

So, the daily mission planning problem can be rewritten in terms of paths from 0 to $n^c + 1$ in $G^c_{\mathcal{E}}$, with $c = 1, 2, 3$, using the traditional vertex-arc formulation as follows:

$$
(\mathrm{S5_{VA}}) \left\{
\begin{array}{ll}
\max \displaystyle\sum_{i=1}^{n} \sum_{c=1}^{3} w_i \sum_{j \in \Gamma_{G^c_{\mathcal{E}}}(i)} \varphi^c_{ij} & \\
\displaystyle\sum_{c=1}^{3} \sum_{j \in \Gamma_{G^c_{\mathcal{E}}}(i)} \varphi^c_{ij} \leqslant 1 \ \ \forall i \in M & (1_{\rm VA}) \\
\displaystyle\sum_{j \in \Gamma_{G^1_{\mathcal{E}}}(i)} \varphi^1_{ij} - \sum_{j \in \Gamma_{G^3_{\mathcal{E}}}(i)} \varphi^3_{ij} = 0 \ \ \forall i \in T & (2_{\rm VA}) \\
A^c \varphi^c = b^c \ \ \forall c = 1, 2, 3 & (3_{\rm VA}) \\
\sum_{j \in \Gamma_{G^c_{\mathcal{E}}}(i)} \varphi^c_{ij} + \sum_{k \in \Gamma_{G^c_{\mathcal{E}}}(jk)} \varphi^c_{jk} \leqslant 1 \ \forall (i,j) \in \hat{E}^c \setminus E^c & (3^+_{\rm VA}) \\
\displaystyle\sum_{(i,c) \in \Lambda^k} \sum_{l \in \Gamma_{G^c_{\mathcal{E}}}(i)} \varphi^c_{il} \leqslant \delta^k \ \ \forall k = 1, \ldots, K & (4_{\rm VA}) \\
\displaystyle\sum_{c=1}^{3} \sum_{i=1}^{n} z^c_i \sum_{j \in \Gamma_{G^c_{\mathcal{E}}}(i)} \varphi^c_{ij} \leqslant z & (5_{\rm VA}) \\
\varphi^c_{ij} \in \{0, 1\} \ \ \forall (i,j) \in E^c \ \ \forall c = 1, 2, 3. &
\end{array}
\right.
$$

where each binary variable φ^c_{ij} is associated with arc $(i,j) \in E^c$, $\Gamma_{G^c_{\mathcal{E}}}(i)$ is the set of nodes adjacent to i in $G^c_{\mathcal{E}}$, A^c is the vertex-arc incidence matrix of $G^c_{\mathcal{E}}$, b^c is an $n^c + 2$ vector with all coefficients equal to 0 except the first which is equal to 1 and the last which is equal to -1.

$\mathrm{S5_{VA}}$ is defined from $\mathrm{S5_N}$ by replacing variables x^c_i by $\sum_{j \in \Gamma_{G^c_{\mathcal{E}}}(i)} \varphi^c_{ij}$ in all integer linear program excepted for 3_N constraints corresponding to arcs in E^c. Constraints 3_N corresponding to arcs in E^c are translated in terms of paths with constraints $3_{\rm VA}$.

The optimal solution value of the linear relaxation of $\mathrm{S5_{VA}}$, denoted by $v(\overline{\mathrm{S5_{VA}}})$, is an upper bound for $v(\mathrm{S5_{VA}})$ which is equal to $v(\mathrm{S5_N})$. In the next section, we show that $v(\overline{\mathrm{S5_{VA}}}) \leqslant v(\overline{\mathrm{S5_N}})$.

12.3. Model comparison

In [VAS 01], the daily mission planning problem is seen as a knapsack problem with a huge number of additional logic constraints on variables. In order to compare $\overline{\mathrm{S5_{VA}}}$ and $\overline{\mathrm{S5_{N}}}$, we prefer to consider this problem as a maximum weighted stable set problem with additional linear constraints. Indeed, on each camera c, a set of images that can be taken together can be represented by a stable set in a graph, namely the incompatibility graph, denoted by $G^c_{\mathcal{I}} = (X^c, I^c)$ which is defined as follows:

– X^c represents the set S^c (with $\mid S^c \mid = n^c$) of images that can be taken by camera c (with the same convention: a couple (i, c) is represented by a vertex numbered by i in X^c),

– each vertex i in X^c is associated with the weight w_i of image i,

– I^c represents the incompatibility relation defined on $S^c \times S^c$.

Any set of images that can be taken on camera c corresponds to a set of vertices belonging to a stable set in $G^c_{\mathcal{I}}$. Respectively, any stable set in $G^c_{\mathcal{I}}$ represents a set of images that can be taken on camera c. In the following section, we recall some well-known results about stable set polytope formulations.

12.3.1. *About the stable set polytope*

Let us consider the stable set problem in an arbitrary graph $G = (V, E)$. The stable set polytope, usually denoted by $\mathtt{STAB}(G)$, can be written as:

$$\mathtt{STAB}(G) = conv(x^S \in \{0,1\}^{|V|} : S \subseteq V \text{ is a stable set in } G)$$

where x^S is the incidence vector of S.

The stable set polytope and its LP-relaxation have been the subject of numerous studies (see, for instance, [GRÖ 88]).

The **edge constraints** of the form:

$$x_i + x_j \leqslant 1 \; \forall (ij) \in E$$

are facet-defining inequalities of $\mathtt{STAB}(G)$. And we have:

$$\mathtt{STAB}(G) = conv(x \in \{0,1\}^{|V|} : x \text{ satisfies all edge constraints})$$

In general cases, the **edge polytope** defined as follows:

$$\mathtt{ESTAB}(G) = \{x \in \mathbb{R}^{|V|}_{+} : x \leqslant 1 \text{ and satisfies all edge constraints}\}$$

is not equal to the stable set polytope (except for bipartite graphs; see, for instance, [GRÖ 88]). In other words, all edge constraints are not sufficient to describe the stable set polytope, additional facet-defining inequalities must be introduced.

Obviously, **maximal clique constraints** of the form:

$$\sum_{i \in Q} x_i \leqslant 1, \text{ with } Q \subseteq V$$

where Q is a maximal clique in G, generalize edge constraints. The **clique polytope** (also called fractional stable set polytope) is defined as follows:

$$\texttt{QSTAB}(G) = \{x \in \mathbb{R}_+^{|V|} : x \text{ satisfies all maximal clique constraints}\}$$

For an arbitrary graph, we have $\texttt{STAB}(G) \subseteq \texttt{QSTAB}(G) \subseteq \texttt{ESTAB}(G)$. But, if G is a perfect graph, it appears that $\texttt{STAB}(G) = \texttt{QSTAB}(G)$.

12.3.2. *Stable set polytope and daily mission planning problem formulations*

For a given c, constraints $x_i^c + x_j^c \leqslant 1 \ \forall (i,j) \in I^c$ are edge constraints on $G_{\mathcal{I}}^c = (X^c, I^c)$ and, the polytope $conv\{x^c \in \{0,1\}^{n^c} : x_i^c + x_j^c \leqslant 1 \ \forall (i,j) \in I^c\}$ defines the stable set polytope $\texttt{STAB}(G_{\mathcal{I}}^c)$. Consequently, in $\overline{\mathrm{S5_N}}$, this stable set polytope is replaced by the corresponding edge polytope $\texttt{ESTAB}(G_{\mathcal{I}}^c) = \{x^c \in [0,1]^{n^c} : x_i^c + x_j^c \leqslant 1 \ \forall \{(i,j)\} \in \mathcal{I}^c\}$ which does not correspond to $\texttt{STAB}(G_{\mathcal{I}}^c)$ since there is no reason for $G_{\mathcal{I}}^c$ to be a bipartite graph. This difference between the two polytopes partially explains the huge integrality gap between $\mathrm{S5_N}$ and $\overline{\mathrm{S5_N}}$. In fact, the difference between $\overline{\mathrm{S5_N}}$ and $\overline{\mathrm{S5_{VA}}}$ is related to $\texttt{STAB}(G_{\mathcal{I}}^c)$ formulations.

THEOREM 12.1.– $v(\overline{\mathrm{S5_{VA}}}) \leqslant v(\overline{\mathrm{S5_N}})$.

Proof. Let us first consider the transitivity of the enchainability relation. In this case, any stable set in $G_{\mathcal{I}}^c$ corresponds to a path from 0 to $n^c + 1$ in $G_{\mathcal{E}}^c$ and conversely. In fact, $G_{\mathcal{E}}^c$ can be easily deduced from $G_{\mathcal{I}}^c$ as follows: define the complementary graph of $G_{\mathcal{I}}^c$, replace each edge between i and j by an arc from i to j if image i is taken before image j and by an arc from j to i otherwise. The obtained directed graph is exactly $G_{\mathcal{E}}^c$. Let us note that a graph such that its complementary graph has a transitive orientation is a well-known perfect graph, namely a co-comparability graph. In [BAR 03], it is shown that the stable set polytope in co-comparability graph can be exactly described by the path-based formulation. Consequently, the polytope $\Phi = \{\varphi_{ij}^c \in [0,1]^{|E^c|} : A^c \varphi^c = b^c\}$ is equal to $\texttt{STAB}(G_{\mathcal{I}}^c)$ for all $c = \{1,2,3\}$, and included in the edge polytope $\texttt{ESTAB}(G_{\mathcal{I}}^c)$. Thus, $v(\overline{\mathrm{S5_{VA}}}) \leqslant v(\overline{\mathrm{S5_N}})$.

In the non-transitivity case, any path from 0 to n^c+1 in $G_{\mathcal{E}}^c$ corresponds to a stable set in a subgraph $\tilde{G}_{\mathcal{I}}^c$ of $G_{\mathcal{I}}^c$. $\tilde{G}_{\mathcal{I}}^c$ is obtained from $G_{\mathcal{I}}^c$ by removing all edges (i,j) such

that either (i,j) or (j,i) belongs to $\hat{E}^c \setminus E^c$. $\tilde{G}^c_{\mathcal{I}}$ is a co-comparability graph and the polytope Φ corresponds to the stable set polytope of $\texttt{STAB}(\tilde{G}^c_{\mathcal{I}})$ which is included in $\texttt{ESTAB}(\tilde{G}^c_{\mathcal{I}})$. Since $\texttt{STAB}(G^c_{\mathcal{I}}) = \texttt{STAB}(G^c_{\mathcal{I}} \setminus \tilde{G}^c_{\mathcal{I}}) \cap \texttt{STAB}(\tilde{G}^c_{\mathcal{I}})$ and, in $\overline{\mathrm{S5_N}}$, $\texttt{STAB}(\tilde{G}^c_{\mathcal{I}})$ is replaced by $\texttt{ESTAB}(\tilde{G}^c_{\mathcal{I}})$ while, in $\overline{\mathrm{S5_{VA}}}$, $\texttt{STAB}(\tilde{G}^c_{\mathcal{I}})$ is exactly described by Φ, we have $v(\overline{\mathrm{S5_{VA}}}) \leqslant v(\overline{\mathrm{S5_N}})$. ∎

We calculate $v(\overline{\mathrm{S5_N}})$ and $v(\overline{\mathrm{S5_{VA}}})$ on a set of realistic benchmark instances. Results are presented in the following section.

12.4. Experiments and results

Experiments are carried out on a set of realistic instances described in [BEN 99]. There are two classes of instances:

– the first class includes small size instances without recording capacity constraint,

– the second class includes bigger size instances with recording capacity constraint.

For the first class instances, optimal solution values are known. For the second class instances, approximate solutions have been determined using tabu search (for details see [VAS 01]) and, for some of them, optimality has been proved in [VAS 03].

Each instance is only described by a list of all binary and ternary constraints. So, when two images i and j are not linked by a constraint, we do not know if i is taken before j or j before i. Thus, we are not able to define $G^c_{\mathcal{E}}$ from the instance description. Consequently, we propose to construct $G^c_{\mathcal{E}}$ with the polynomial time algorithm presented in [GIL 64] for determining a transitive orientation of a comparability graph. When this algorithm is applied on an arbitrary graph, it defines a circuit-free orientation of edges. Thus, applied to the complementary graph of $G^c_{\mathcal{I}}$, this algorithm defines a circuit-free directed graph (transitive when $G^c_{\mathcal{I}}$ is a co-comparability graph) that can be used instead of $G^c_{\mathcal{E}}$.

For each instance, we calculate with Cplex 8.0 the optimal solution value of $\overline{\mathrm{S5}}_{\mathrm{N}}$ and $\overline{\mathrm{S5_{VA}}}$. Tables 12.1 and 12.2 present the obtained results.

Upper bounds provided by $\overline{\mathrm{S5_{VA}}}$ are logically closer to the optimal solution value of $\mathrm{S5_N}$ than those provided by $\overline{\mathrm{S5_N}}$. Let us note that $G^c_{\mathcal{I}}$ is a co-comparability graph only for instances 54, 42, 1502 and, in these cases, $v(\overline{\mathrm{S5_{VA}}})$ is very close to optimal solution value. Moreover, upper bounds provided by $\overline{\mathrm{S5_{VA}}}$ are more interesting for difficult big size instances with one knapsack constraint 5_{N}.

Some improvements would be possible with another instance description. First of all, the algorithm we use to define an orientation of the complementary graph of $G^c_{\mathcal{I}}$

Instance	Number of images	$v(\mathrm{S5_N})$	$v(\overline{\mathrm{S5_N}})$	$v(\overline{\mathrm{S5_{VA}}})$	$\frac{v(\overline{\mathrm{S5_N}})-v(\mathrm{S5_N})}{v(\overline{\mathrm{S5_N}})}\%$	$\frac{v(\overline{\mathrm{S5_{VA}}})-v(\mathrm{S5_N})}{v(\overline{\mathrm{S5_{VA}}})}\%$
54	67	70	83	71	15	1
42	190	108067	190567	108067	43	0
28	230	56053	221090	67053	74	16
5	309	114	315	151	63	24
404	100	49	96	49	48	0
408	200	3082	5188	3094	40	0.3
412	300	16102	31323	23580	48	31
503	143	9096	12637	9601	28	5
505	240	13100	22236	15803	41	17
507	311	15137	27361	25278	44	40
509	348	19125	36394	25684	47	25

Table 12.1. *Integrality gaps for small size SPOT5 instances*

Instance	Number of images	$\hat{v}(\mathrm{S5_N})$	$v(\overline{\mathrm{S5_N}})$	$v(\overline{\mathrm{S5_{VA}}})$	$\frac{v(\overline{\mathrm{S5_N}})-v(\mathrm{S5_N})}{v(\overline{\mathrm{S5_N}})}\%$	$\frac{v(\overline{\mathrm{S5_{VA}}})-v(\mathrm{S5_N})}{v(\overline{\mathrm{S5_{VA}}})}\%$
1401	488	176056	300000	188370	41	6
1403	665	176140	300149	188467	41	6
1405	855	176179	300207	188551	41	6
1502	209	61158	64160	64155	4	4
1504	605	124243	191279	139585	35	11
1506	940	168247	276863	183978	39	8

Table 12.2. *Integrality gaps for big size SPOT5 instances*

selects in an arbitrary way a co-comparability subgraph of $G^c_{\mathcal{I}}$; it is obvious that $S5_{\mathrm{VA}}$ would be strengthened if it was based upon the real enchainability graph $G^c_{\mathcal{E}}$.

Moreover, $\mathrm{S5_{VA}}$ can be strengthened in rewriting contraints 4_N. These constraints are due to the limitation of the instantaneous data flow through the satellite telemetry. A much better formulation is to write cumulative constraints of that type: at each time, the sum of the data flows used by images must be lower or equal to a certain limit.

12.5. Conclusion

The daily satellite mission planning problem can be seen as a stable set problem with additional linear constraints. In this chapter, we compare linear relaxations of two 0-1 linear formulations. In the first one, stable set polytope is represented by edge polytope while, in the second one, stable set polytope is exactly described. The theoretical differences between these two formulations are also illustrated by numerical experiments on benchmark instances. It is shown that the 0-1 linear programming is of real interest for modeling and solving satellite mission planning.

12.6. Bibliography

[BAR 03] BARCIA P., CERDEIRA J., "Node packings on co-comparability graphs", *Operations Research Letters*, vol. 31, p. 341-342, 2003.

[BEN 96] BENSANNA E., VERFAILLIE G., AGNÈSE J.-C., BLUMSTEIN N. B. D., "Exact and approximate methods for the daily management of an earth observation satellite", *Proc. of the* 4th *Int. Symposium on Space Mission Operations and Ground Data Systems, Munich, Germany*, 1996.

[BEN 99] BENSANNA E., LEMAÎTRE M., VERFAILLIE G., "Earth observation satellite management", *Constraints: an International Journal*, vol. 4, num. 3, p. 293-299, 1999.

[GAB 03] GABREL V., MURAT C., "Mathematical programming for Earth observation satellite mission planning", CIRIANI T., FASANO G., GLIOZZI S., TADEI R., Eds., *Operations Research in Space and Air*, p. 103-122, Kluwer Academics Publishers, 2003.

[GIL 64] GILMORE P., HOFFMAN A., "A characterization of comparability graphs and of interval graphs", *Can. Journal Math.*, vol. 16, p. 539-548, 1964.

[GRÖ 88] GRÖTSCHEL M., LOVÀSZ L., SCHRIJVER A., *Geometric Algorithms and Combinatorial Optimization*, Springer, 1988.

[VAS 01] VASQUEZ M., HAO J.-K., "A Logic-Constrained Knapsack formulation and a Tabu Algorithm for the Daily Photograph Scheduling of an Earth Observation satelite", *Journal of Computational Optimization and Applications*, vol. 20, num. 2, p. 137-157, 2001.

[VAS 03] VASQUEZ M., HAO J.-K., "Upper Bounds for the SPOT5 Daily Photograph Scheduling Problem", *Journal of Combinatorial Optimization*, vol. 7, p. 87-103, 2003.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 13

Dantzig-Wolfe Decomposition for Linearly Constrained Stable Set Problem

13.1. Introduction

Large-scale 0-1 linear programs often present a strong structure on which a decomposition scheme can be applied. One of the most famous decomposition schemes in linear programming is the Dantzig-Wolfe decomposition proposed in [DAN 60]. This decomposition scheme can be applied to 0-1 linear programs [BAR 98]. It leads us to reformulate the initial program into another one, called the master program, only containing a subset of constraints on some original variables plus additional variables representing all the 0-1 solutions of the decomposed constraints. When the polytope defined by the decomposed constraints does not present the integrality property, it is well-known that the LP-relaxation of the master program is a strengthened formulation of the initial program continuous relaxation. But, in this case, the LP-relaxation of the master program is difficult to solve since the variables involved in an optimal solution, which represent some 0-1 solutions of the decomposed constraints, cannot be determined in polynomial time (the problem of determining a particular 0-1 solution of the decomposed constraints is called the auxiliary problem).

In this chapter, we focus on a particular class of 0-1 linear programs in which a subset of constraints defines a stable set polytope. We study the difficulty and the interest of applying the Dantzig-Wolfe decomposition scheme on the stable set constraints. When the decomposed constraints represent a stable set problem in a perfect graph, the auxiliary problem can be solved in polynomial time and the optimal solution value

Chapter written by Virginie GABREL.

of the master program LP-relaxation can be obtained by column generation procedure. For some particular classes of perfect graphs (like interval or co-comparability graphs), another formulation – more compact – of the master program can be proposed and directly solved without using column generation procedure. For general graphs, the auxiliary problem is **NP**-hard. In this case, an alternative approach is to select some stable set constraints which define a stable set problem in a perfect subgraph and, to decompose only on these selected constraints. In this way, the master program can be solved in polynomial time whereas it is a strengthened formulation of the traditional continuous relaxation. The selection of a subset of constraints may be difficult but, for mathematical programs associated with some particular application, the structure induced by the application itself may be a sufficient guide.

In section 13.2, we recall the Dantzig-Wolfe decomposition scheme in 0-1 linear programming. Then, in section 13.3, we present the considered problem, namely the stable set problem with additional linear constraints. In section 13.4, we study the gap between the traditional continuous relaxation and the linear relaxation of the master problem, obtained by decomposing on stable set constraints with regards to various stable set polytope representations for general and perfect graphs. For general graphs, an alternative approach based on the decomposition on some subsets of the stable set constraints is briefly introduced.

13.2. The Dantzig-Wolfe decomposition in 0-1 linear programming

Consider the 0-1 linear program P of the form:

$$(P)\begin{cases} \max cx & \\ Ax \leqslant a & m_1 \text{ constraints} \\ Bx \leqslant b & m_2 \text{ constraints} \\ x \in \{0,1\}^n & \end{cases}$$

where $A \in \mathbb{N}^{m_1 \times n}$ and $B \in \mathbb{N}^{m_2 \times n}$ are integer matrices, $c \in \mathbb{N}^n$, $a \in \mathbb{N}^{m_1}$ and $b \in \mathbb{N}^{m_2}$ are integer vectors. The set $X_I^B = X^B \cap \{0,1\}^n$, with $X^B = \{x \in \mathbb{R}_+^n : Bx \leqslant b, x \leqslant 1\}$, contains a finite set of vectors, denoted by $\{x^1, x^2, \ldots, x^q\}$, and any point $x \in X_I^B$ can be represented as:

$$x = \sum_{k=1}^{q} x^k \lambda_k$$

with $\sum_{k=1}^{q} \lambda_k = 1$ and $\lambda_k \in \{0,1\}\ \forall k = 1, \ldots, q$.

The main idea of the Dantzig-Wolfe decomposition in integer programming [BAR 98] is to rewrite P using the fact that any solution of P is necessarily a solution of X_I^B, as follows:

$$(M)\begin{cases} \max \sum_{k=1}^{q} cx^k \lambda_k \\ \sum_{k=1}^{q} Ax^k \lambda_k \leqslant a \\ \sum_{k=1}^{q} \lambda_k = 1 \\ \lambda_k \in \{0,1\} \forall k = 1, \ldots, q \end{cases}$$

M is called the master problem.

If we now consider the linear relaxations of P and M, denoted by $\bar{P}$ and $\bar{M}$ respectively, it is well known that (see, for example, [BAR 98]):

$$v(\bar{P}) \geqslant v(\bar{M}) \geqslant v(M) = v(P)$$

Indeed, it is easy to see that:

$$(\bar{M})\begin{cases} \max \sum_{k=1}^{q} cx^k \lambda_k \\ \sum_{k=1}^{q} Ax^k \lambda_k \leqslant a \\ \sum_{k=1}^{q} \lambda_k = 1 \\ \lambda_k \geqslant 0 \ \forall k = 1, \ldots, q \end{cases}$$

is equivalent to:

$$\begin{cases} \max cx \\ Ax \leqslant a \\ x \in conv(X_I^B) \end{cases}$$

where $conv(X_I^B)$ denotes the convex hull of X_I^B. Thus, $v(\bar{P}) \geqslant v(\bar{M})$ since $v(\bar{P}) = \{\max cx : Ax \leqslant a,\ x \in X^B\}$ and $conv(X_I^B) \subseteq X^B$.

Consequently, when the polyhedron X^B presents the integrality property, $X^B = conv(X_I^B)$, we have: $v(\bar{P}) = v(\bar{M})$. Moreover, when $conv(X) = \{Ax \leqslant a\} \cap conv(X_I^B)$, where X is the set of all P feasible solutions, we have: $v(P) = v(\bar{M})$.

These two previous cases are particular and, in general, we have:

$$v(\bar{P}) > v(\bar{M}) > v(M) = v(P)$$

These inequalities are very important since we need to calculate a tight upper bound either for exactly solving a large-scale linear integer program in a branch-and-bound scheme, or for evaluating the quality of approximate solutions. In this context, the M formulation and the calculation of $v(\bar{M})$ can be really interesting.

Since the number of variables in $\bar{M}$ can be huge (because it is equal to the cardinality of X_I^B), a column generation procedure must be applied to calculate $v(\bar{M})$. It consists of generating only a subset of variables: those which may participate to an optimal solution. Starting from a subset J of some initial variables (completed with some artificial variables if necessary), we have to solve at each iteration of the column generation procedure the following linear problem:

$$(\bar{M}_J) \begin{cases} \max \sum_{k \in J} cx^k \lambda_k \\ \sum_{k \in J} Ax^k \lambda_k \leqslant a & (1) \\ \sum_{k \in J} \lambda_k = 1 & (2) \\ \lambda_k \geqslant 0 \ \forall k \in J \end{cases}$$

and to determine among all remaining variables that which has the greatest reduced cost. The reduced cost of a variable k is defined as follows:

$$\bar{c}_k = c_k - \pi A x_k - \pi_0$$

where π denotes the dual variables associated with constraint (1) and π_0 the dual variable associated with constraint (2). The variable with the greatest strictly positive reduced cost (or several variables with strictly positive reduced cost; see [BAR 98] for details concerning column generation procedures) must be included in J and $\bar{M}_J$ must be reoptimized. The procedure stops when all the remaining variables have negative or zero reduced cost and the optimal solution of $\bar{M}_J$ is thus an optimal solution of $\bar{M}$.

The problem of determining a variable, or equivalently a solution of X_I^B, with the greatest reduced cost, called the auxiliary problem denoted by Aux, can be formulated as follows:

$$(Aux) \begin{cases} \max(c - \pi A)x - \pi_0 \\ Bx \leqslant b \\ x \in \{0,1\}^n \end{cases}$$

In what follows, we focus on a particular class of $conv(X_I^B)$ polyhedron: the stable set polytope.

13.3. The stable set problem with additional linear constraints

In this section, we suppose that the m_2 constraints $Bx \leqslant b$ are of the form:

$$\sum_{i \in Q_k} x_i \leqslant 1 \text{ with } Q_k \subseteq \{1, \ldots, n\} \text{ and } |Q_k| \geqslant 2 \ \forall k = 1, \ldots, m_2$$

These constraints can also be represented by a graph $G = (V, E)$ defined as follows:

– each variable x_i, $i = 1, \ldots, n$, is represented by a vertex $i \in V$ (thus $|V| = n$),

– each constraint k, $k = 1, \ldots, m_2$ is represented by one or several edges (ij) of E for all i and j (with $i < j$) belonging to Q_k.

A solution x belonging to X_I^B can be seen as the incidence vector of a set $S \subseteq V$ of vertices (with $x_i = 1$ if $i \in S$ and 0 otherwise) such that: $\forall (i, j) \in S \times S, (ij) \notin E$. Consequently, S is a stable set in G and $conv(X_I^B)$ is exactly the stable set polytope, which is usually denoted by $\texttt{STAB}(G)$:

$$\texttt{STAB}(G) = conv(x^S \in \{0,1\}^{|V|} : S \subseteq V \text{ is a stable set in } G)$$

In this context, the considered program P corresponds to the maximum weighted stable set problem with additional linear constraints (the first m_1 constraints).

The stable set polytope and its LP-relaxation has been the subject of numerous studies (see, for instance, [GRÖ 88]).

Considering an arbitrary graph $G = (V, E)$, the **edge constraints** of the form:

$$x_i + x_j \leqslant 1 \ \forall (ij) \in E$$

are facet-defining inequalities of $\texttt{STAB}(G)$. And we have:

$$\texttt{STAB}(G) = conv(x \in \{0,1\}^{|V|} : x \text{ satisfies all edge constraints})$$

In general cases, the **edge polytope** defined as follows:

$$\texttt{ESTAB}(G) = \{x \in \mathbb{R}_+^{|V|} : x \leqslant 1 \text{ and satisfies all edge constraints}\}$$

is not equal to the stable set polytope (except for bipartite graphs, see, for instance, [GRÖ 88]). In other words, all edge constraints are not sufficient to describe the stable set polytope and additional facet-defining inequalities must be introduced.

Obviously, **maximal clique constraints** of the form:

$$\sum_{i \in Q} x_i \leqslant 1, \text{ with } Q \subseteq V$$

where Q is a maximal clique in G, generalize edge constraints. The **clique polytope** (also called fractional stable set polytope) is defined as follows:

$$\texttt{QSTAB}(G) = \{x \in \mathbb{R}_+^{|V|} : x \text{ satisfies all maximal clique constraints}\}$$

For an arbitrary graph, we have $\texttt{STAB}(G) \subseteq \texttt{QSTAB}(G) \subseteq \texttt{ESTAB}(G)$ (see, for instance, [GRÖ 88]). But, if G is a perfect graph, it appears that $\texttt{STAB}(G) = \texttt{QSTAB}(G)$.

In the next section, we apply the Dantzig-Wolfe decomposition scheme on stable set constraints and we compare linear relaxations of master and initial programs with regards to various stable set polytope representations.

13.4. Dantzig-Wolfe decomposition on stable set constraints: strengthening the LP-relaxation

13.4.1. *Gap between LP-relaxations of master and initial problems*

In section 13.2, we recalled that the LP-relaxation of the master problem in a Dantzig-Wolfe decomposition scheme is equivalent to:

$$\begin{cases} \max cx \\ Ax \leqslant a \\ x \in conv(X_I^B) \end{cases}$$

When $conv(X_I^B)$ exactly corresponds to a stable set polytope, $\bar{M}$ can be reformulated as follows:

$$\begin{cases} \max cx \\ Ax \leqslant a \\ x \in \texttt{STAB}(G) \end{cases}$$

Consequently, if G is an arbitrary graph, $v(\bar{M}) \leqslant v(\bar{P})$ since $\texttt{STAB}(G) \subseteq X^B$. Otherwise, if G is a perfect graph or the complementary graph of a perfect graph, $\texttt{STAB}(G)$ can be replaced by $\texttt{QSTAB}(G)$ (since in this case $\texttt{STAB}(G) = \texttt{QSTAB}(G)$) and we have:

– if $\texttt{QSTAB}(G) \subset X^B$, $v(\bar{M}) \leqslant v(\bar{P})$,
– if $\texttt{QSTAB}(G) = X^B$, $v(\bar{M}) = v(\bar{P})$.

Consequently, when X^B does not exactly describe the clique polytope, the LP-relaxation of the master problem in a Dantzig-Wolfe decomposition scheme may give a tighter upper bound than the LP-relaxation of the initial problem. The gap comes from the difference between the two polytopes X^B and $\texttt{QSTAB}(G)$.

Moreover, the auxiliairy problem Aux can be reformulated as follows:

$$\begin{cases} \max(c - \pi A)x - \pi_0 \\ x \in \texttt{STAB}(G) \end{cases}$$

Thus, Aux is the maximal weight stable set problem. If G is an arbitrary graph, the problem is **NP**-hard ; otherwise, if G is a perfect graph, this problem can be solved in polynomial time.

In this context, when $\texttt{STAB}(G) \subset X^B$, an obvious way for strengthening $\bar{P}$ is to add in X^B some additional facet-defining inequalities for $\texttt{STAB}(G)$ (this is done by a solver like Cplex when it determines the so-called clique cut; see [ILO 02]). When G is a perfect graph, replacing X^B by $\texttt{QSTAB}(G)$, the calculation of $v(\bar{M})$ remains hard. In this case, the difficulty comes from the possibly huge number of maximal clique constraints. Indeed, except for some particular graphs (interval graph for example) for which the number of maximal cliques is bounded ($O(n)$ for interval graph, see for example [GIL 64]), there are perfect graphs with an exponential number of maximal cliques. However, this problem can be avoided using either constraint generation techniques or, if possible, a more compact reformulation. In the following section, we focus on a particular class of graphs, namely co-comparability graphs, and we show how to obtain a more compact formulation for $\bar{M}$.

13.4.2. *The case of co-comparability graphs*

A simple graph is a comparability graph if it has a transitive orientation, i.e., an orientation of edges such that (i, j) and (j, k) implies (i, k). In [GIL 64, PNU 71], polynomial time algorithms are proposed for determining if a graph is a comparability graph, and for such graphs, for defining a transitive orientation in polynomial time. A co-comparability graph is the complement of a comparability graph. A co-comparability graph is a perfect graph but it can have an exponential number of maximal cliques. However, there is a more compact formulation of the stable set polytope using flow variables (as also suggested in [BAR 03]).

Let us consider $G = (X, U)$ a co-comparability graph. A stable set in G corresponds to a clique in its complement graph, which is denoted by $G_C = (X, U_C)$ with $|U_C| = m$. Since G_C is a comparability graph, we can define $\overrightarrow{G_C} = (X, \overrightarrow{U_C})$ as a digraph obtained by choosing a transitive orientation of U_C. Let us add into $\overrightarrow{G_C}$ two fictitious vertices e and s and the following arcs in $\overrightarrow{U_C}$: (e, s), (e, i) and (i, s) for all $i = 1, \ldots, n$. Thus, any stable set in G defines a path between e and s in $\overrightarrow{G_C}$ and conversely. Consequently, it is possible to represent $\texttt{STAB}(G)$, or equivalently $\texttt{QSTAB}(G)$, by a traditional vertex-arc formulation of paths.

In the vertex-arc formulation of the stable set polytope, each variable ϕ_{jk} represents an arc (j, k) in $\overrightarrow{U_C}$ and $\phi = (\phi_{jk})_{(j,k)\in\overrightarrow{U_C}}$ is an incidence vector of a path between e and s in $\overrightarrow{G_C}$ if it satisfies the following constraints:

$$\begin{cases} \sum_{k\in\Gamma^+(j)} \phi_{jk} - \sum_{k\in\Gamma^-(j)} \phi_{kj} = 0 & \forall j = \{1, \ldots, n\} \\ \sum_{k\in\Gamma^+(e)} \phi_{ek} = 1 \\ -\sum_{k\in\Gamma^-(s)} \phi_{ks} = -1 \\ \phi \in \{0, 1\}^m \end{cases}$$

The number of variables is equal to the number of arcs in $\overrightarrow{U_C}$, i.e. $m + 2n + 1$ or equivalently $O(n^2)$, while the number of contraints equals $n + 2 + m_1$ or equivalently $O(n^2)$. Moreover, it is well-known that this polyhedron presents the integrality property (since the constraint matrix is a vertex-arcs incidence matrix).

As we showed in the previous section, a way to strengthen $\bar{P}$ (and for perfect graph to reach the bound provided by the Dantzig-Wolfe decomposition scheme) is to replace X^B by $\texttt{QSTAB}(G)$. For a co-comparability graph, a way to describe $\texttt{QSTAB}(G)$ with a polynomial number of variables and constraints is to use the vertex-arc formulation as follows:

$$\begin{cases} \max \sum_{j=1}^{n} c_j \sum_{k\in\Gamma^+(j)} \phi_{jk} \\ \sum_{j=1}^{n} a_{ij} \sum_{k\in\Gamma^+(j)} \phi_{jk} \leqslant a_i & \forall i = 1, \ldots, m_1 \\ \sum_{k\in\Gamma^+(j)} \phi_{jk} - \sum_{k\in\Gamma^-(j)} \phi_{kj} = 0 & \forall j = \{1, \ldots, n\} \\ \sum_{k\in\Gamma^+(e)} \phi_{ek} = 1 \\ -\sum_{k\in\Gamma^-(s)} \phi_{ks} = -1 \\ 0 \leqslant \phi_{jk} \leqslant 1 \; \forall (j, k) \in \overrightarrow{U_C} \end{cases}$$

where x and ϕ variables are linked that way:

$$x_j = \sum_{k\in\Gamma^+(j)} \phi_{jk} \; \forall j = 1, \ldots, n.$$

Consequently, this new formulation must be used to calculate $v(\bar{M})$.

13.4.3. *A decomposition scheme for general graphs*

For a general graph G, the LP-relaxation of the master program obtained by applying the Dantzig-Wolfe decomposition scheme on all the stable set constraints cannot be solved in polynomial time because the auxiliary problem is **NP**-hard. An alternative approach is to identify one or several perfect subgraphs of G and to decompose only on the constraints describing these subgraphs as follows:

1) determine one or several perfect subgraphs $G^k = (V^k, U^k)$ with $k = 1, \ldots, K$ ($K \geqslant 1$) of G,

2) separate the constraints set $Bx \leqslant b$ into $K + 1$ subsets: for $k = 1, \ldots, K$, $B^k x \leqslant b^k$ are all the stable set constraints represented in G^k, $\bar{B}x \leqslant \bar{b}$ are all the remaining constraints,

3) apply the Dantzig-Wolfe decomposition on $B^k x \leqslant b^k$, with $k = 1, \ldots, K$.

The obtained master program denoted by M' is equivalent to:

$$\begin{cases} \max cx \\ Ax \leqslant a \\ \bar{B}x \leqslant \bar{b} \\ x \in \texttt{STAB}(G^k) \quad \forall k = 1, \ldots, K \end{cases}$$

Thus, $v(\bar{M'})$ is tractable and we have : $v(\bar{P}) \geqslant v(\bar{M'}) \geqslant v(\bar{M})$. In order to calculate $v(\bar{M'})$, several identical methods can be used: a direct calculation with column generation procedure, a constraint generation procedure applied to the clique polytope formulation of $\texttt{STAB}(G^k)$ and, when $\texttt{QSTAB}(G^k)$ can be described with a "polynomial" number of clique constraints for all $k = 1, \ldots, K$, a direct resolution of the clique polytope formulation.

In this approach, the determination of one or several perfect subgraphs may be very difficult for a general graph. But, for a graph associated with a real application, the structure of the problem may be a powerful guide for identifying some subgraphs with "nice" properties.

13.5. Conclusion

When solving exactly or approximately large-scale integer linear programs (ILP), very important information comes from the calculation of the linear relaxation of ILP. It is well-known that, when applying the Dantzig-Wolfe decomposition on ILP, the linear relaxation of the master problem $\bar{M}$ may present a smaller integrality gap.

We consider here a particular class of ILP, i.e., those which contain a subset of constraints defining a stable set polytope. In this case, when the decomposition is performed on the stable set constraints, the difference between LP-relaxations of ILP and M can be partially explained by the gap between the stable set polytope and its corresponding LP-relaxation. In the case of a perfect graph, we show that the LP-relaxation of M is equivalent to another formulation obtained by replacing in ILP the inequalities defining the stable set polytope by the constraints defining the clique polytope. A more compact formulation is then given for co-comparability graphs, based on vertex-arc formulation of paths. In the case of a general graph, $\bar{M}$ cannot be calculated in polynomial time. Thus, an alternative approach is possible in which stable set constraints are separated in order to determine some subsets with "nice" property. This is the subject of further research.

13.6. Bibliography

[BAR 98] BARNHART C., JOHNSON E., NEMHAUSER G., SAVELSBERGH M., VANCE P., "Branch-and-Price: Column Generation for Solving Huge Integer Programs", *Operations Research*, vol. 46, num. 3, p. 316-329, 1998.

[BAR 03] BARCIA P., CERDEIRA J., "Node packings on co-comparability graphs", *Operations Research Letters*, vol. 31, p. 341-342, 2003.

[DAN 60] DANTZIG G., WOLFE P., "Decomposition Principle for Linear programs", *Operations Research*, vol. 8, p. 101-111, 1960.

[GIL 64] GILMORE P., HOFFMAN A., "A characterization of comparability graphs and of interval graphs", *Can. Journal Math.*, vol. 16, p. 539-548, 1964.

[GRÖ 88] GRÖTSCHEL M., LOVÀSZ L., SCHRIJVER A., *Geometric Algorithms and Combinatorial Optimization*, Springer, 1988.

[ILO 02] ILOG, User's manual of Cplex 8.0, 2002.

[PNU 71] PNUELI A., LEMPEL A., EVEN S., "Transitive orientation of graphs and identification of permutation graphs", *Can. Journal Math.*, vol. 23, num. 1, p. 160-175, 1971.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 14

Algorithmic Games

The origins of the concepts related to game theory go back to the Babylonian *Talmud*, where we can find problems like the following[1]:

> If a man was married to three women and died, and if the amount of the Ketubah[2] was 100 zuz for the first, 200 zuz for the second, and 300 zuz for the third, and if his estate only contains 100 zuz, they divide it equally. If the estate contains 200 zuz, the one with the 100-zuz Ketubah gets 50, and the other two get 75 each. If the estate contains 300 zuz, the one with the 100-zuz Ketubah gets 50, the one with the 200-zuz Ketubah gets 100, and the one with the 300-zuz Ketubah gets 150. And if three people [similarly] contributed to a fund, and it is lost or gained, this is how they divide things.

In modern terms, this problem can be modeled as a cooperative game and the suggested solution corresponds to the optimality concept of *nucleolus*.

In contrast with this, no reference of any concept related to games can be found in the part of Greek or Arab literature known to date; in newer times, the origins of development of game theory's basic notions can be found in the *Problèmes plaisants et délectables, qui se font par des nombres*, edited in 1612 by Claude Casper Bachet de Mezirac, but also in a letter from Pascal to Pierre de Fermat (in this letter, Pascal deals with the problem of the fair partition of the stakes in a card game between two

---

Chapter written by Aristotelis GIANNAKOS, Vangelis Th. PASCHOS and Olivier POTTIÉ.

1. Mishnah (Ketubot, Chapter 10, Mishnah 3).

2. Amount of money specified in the marriage contract (Ketubah), that can be collected by the widow from her husband's estate after his death or after a divorce has been pronounced.

equivalent adversaries, whenever the game is not yet finished but one player has got some advantage over the other[3]) and in the book of R. Monmort[4] on games of chance.

Being nourished by various sources like the first attempts in the 19th century - by Pierre Laplace - of establishing a theory of probabilities deserving that name, the works of A. Cournot in mathematical economics [COU 1838] and the systematization of the war theory by Clausewitz[5], the idea of creating a mathematical theory of conflicts was in the air at the beginning of the 20th century, as it can be attested from the papers [BOR 21, BOR 24, BOR 27, BOU 02, MOO 09, STE 60], and above all [ZER 13]. We should also mention [LAS 19] which is considered as a philosopical forerunner of game theory.

This period ends with the seminal talk of J. von Neumann at the Mathematical Society of Göttingen, published in [NEU 28], that sets the foundations of a formal treatment of game theory (for the diverse "paternity" issues that have been raised, see [KUH 58] and for von Neumann's contribution to the formalization of game theory, see [KUH 58]). It has been followed by the classic monography by von Neumann and Morgenstern [NEU 53b], published in 1944, with which game theory has definitively entered to its modern era.

14.1. Preliminaries

14.1.1. *Basic notions on games*

Throughout this chapter, we focus on algorithmic aspects of non-cooperative game theory. The presentation of the basic notions and the notation follow in rough lines the exposé of [VOR 98].

DEFINITION 14.1.– *Let I be a set, $\{X_i\}_{i\in I}$ a set system indexed by I and $\{R_i\}_{i\in I}$ a system of binary relations $R_i \subseteq \prod_{i\in I} X_i \times \prod_{i\in I} X_i$. The triplet:*

$$\Gamma = \left\langle I, \{X_i\}_{i\in I}, \{R_i\}_{i\in I} \right\rangle \qquad [14.1]$$

is called a non-cooperative game of players I with strategy set X_i and preference relation R_i for every player $i \in I$. Relation $P_i = R_i \setminus R_i^{-1}$ is called strict preference

3. Œuvres de Fermat, ed. P. Tannery, C. Henri, vol. II, Paris, 1894; the problem has been reported to young Pascal by the Chevalier de Méré.

4. R. Monmort, *Essai d'analyse sur les jeux de hasard* (2nd revised edition, completed with several unpublished letters), Quillan, Paris 1713 (reedited by Chelsea, New York, 1980).

5. C. von Clausewitz, Vom Kriege, Dümmlers Verlag, Berlin, 1832.

relation of player i. The elements of a set of strategy X_i are called the choices of i in the game Γ and the combinations of choices $\mathbf{x} \in \mathbf{X} = \prod_{i \in I} X_i$ are called the situations of Γ.

In what follows, we deal exclusively with non-cooperative games of a finite number of players, each one having a finite strategy set.

It is often the case that preference relations R_i of Definition 14.1 are represented by *payoff* functions $H_i : \mathbf{X} \longrightarrow \mathbb{R}$, such that:

$$\mathbf{x} R_i \mathbf{y} \Leftrightarrow H_i(\mathbf{x}) \geqslant H_i(\mathbf{y})$$

In this case, a game Γ will be defined by its player set I and, for each player i, his strategies X_i and his payoff function H_i:

$$\Gamma = \left\langle I, \{X_i\}_{i \in I}, \{H_i\}_{i \in I} \right\rangle \qquad [14.2]$$

The modeling of preferences by using functions is the object of study of the *utility theory*, that goes beyond the scope of this chapter. However, it can be noted that if preferences R_i are partial orders, then there is a "fine" representation of them in terms of H_is, i.e., such that:

$$\mathbf{x} P_i \mathbf{y} \Leftrightarrow H_i(\mathbf{x}) > H_i(\mathbf{y})$$

The definition of non-cooperative games introduces in a natural manner the issue of a relation of "collective" preference over a game's situations. Indeed, every attempt to interpret the latter as decisions of the players raises the question of the existence of situations that could be considered as better than others, or even optimal. If, in the case of a traditional optimization problem – that can also be considered as a degenerated game of one player – an optimal solution is usually the value (the choice of strategy/situation) minimizing or maximizing a real function (the payoff function), in the case of a non-cooperative game of more than one player, an optimality concept ceases to be that evident. More generally, this concept can be defined as follows:

DEFINITION 14.2.– *An optimality concept is a function φ associating with every game a subset of its situations set: $\varphi\Gamma \subseteq \mathbf{X}$.*

However, Definition 14.2 is quite general such that "intuitive" optimality concepts as well as "artificial" or trivial ones can be understood. So, it is rather better to follow a reverse approach: define a preference relation over the situations of a game and then determine $\varphi\Gamma$ to be the set of situations such that there is no situation preferable to any of them. Thus, an important class of optimality concepts can be formulated as follows.

Let $K \subseteq I$ be a *coalition* and $\mathbf{x} = (x_i)_{i \in I}$ a situation of Γ. A situation $\mathbf{y} = (y_i)_{i \in I}$ is *attainable* from $\mathbf{x}$ *for* K if and only if $\forall i \in I, y_i \neq x_i \Rightarrow i \in K$. It is easy to see that this attainability relation is an equivalence relation; the equivalence class of situations which are attainable from $\mathbf{x}$ for K will be denoted by $\mathcal{E}_\Gamma(\mathbf{x}, K)$.

For any $K \subseteq I$ and a set of indices S_K, a family of couples of subsets of K indexed by S_K can be chosen:

$$\mathcal{L}(K) = \{\langle L_P^s, L_R^s \rangle : s \in S_K\}$$

Then, a preference relation over $\mathbf{X}$ (denoted $R_{\mathcal{L}(K)}$), can be defined:

$$\mathbf{x} R_{\mathcal{L}(K)} \mathbf{y} \Leftrightarrow \bigvee_{s \in S_K} \left(\left(\bigwedge_{i \in L_P^s} \mathbf{x} P_i \mathbf{y} \right) \bigwedge \left(\bigwedge_{i \in L_R^s} \mathbf{x} R_i \mathbf{y} \right) \right)$$

Provided that $\mathbf{x} P_i \mathbf{y}$ implies $\mathbf{x} R_i \mathbf{y}$, it can be considered without loss of generality that $L_P^s \cap L_R^s = \{\}$.

A situation $\mathbf{x}$ of Γ is called $\mathcal{L}(K)$-optimal whenever there is no situation $\mathbf{y} \in \mathcal{E}_\Gamma(\mathbf{x}, K)$ such that $\mathbf{y} R_{\mathcal{L}(K)} \mathbf{x}$.

Finally, let $\mathcal{K}(K)$ be a set of coalitions $K \subseteq I$ and let $\mathcal{L}(\mathcal{K}) = \{\mathcal{L}(K) : K \in \mathcal{K}\}$ be the set of families of couples corresponding to the coalitions in $\mathcal{K}$. A situation (x) of Γ is then called $\mathcal{L}(\mathcal{K})$-optimal if it is $\mathcal{L}(K)$-optimal for all $K \in \mathcal{K}$. The optimality concept $\varphi_{\mathcal{L}(\mathcal{K})}$ associating with Γ the set $\varphi_{\mathcal{L}(\mathcal{K})}\Gamma$ of its $\mathcal{L}(K)$-optimal situations is called *$\mathcal{K}$-optimality concept*.

The quasi-totality of the optimality concepts explored in non-cooperative game theory are $\mathcal{K}$-optimality concepts. Among the most studied, we can find the following:

- *Pareto $\mathcal{K}$-optimality*: for all $K \in \mathcal{K}$, it is $S_K = K$, $L_P^i = \{i : i \in K\}$, $L_R^i = K \setminus \{i : i \in K\}$:
  - if $\mathcal{K}$ consists of a single coalition K, the concept is called *Pareto K-stability*;
  - if $\mathcal{K} = \{I\}$, the concept is called simply *Pareto stability*[6];
  - Pareto 2^I-optimality is also called *stability in the strong sense*;
- *$\mathcal{K}$-equilibrium*: for all $K \in \mathcal{K}$, it is $S_K = \{\alpha\}$, $L_P^\alpha = K, L_R^\alpha = \{\}$:
  - 2^I-equilibrium is also called *equilibrium in the strong sense*;
  - if $\mathcal{K} = \{\{i\} : i \in I\}$, the corresponding $\mathcal{K}$-equilibrium is called *Nash equilibrium*.

6. Many authors still use for this concept the term *Pareto optimality* instead of "Pareto stability".

EXAMPLE 14.1.– *Prisoner's dilemma*. Consider the game:

$$\langle \{A, B\}, X_A = X_B = \{d, n\}, H_A, H_B \rangle$$

with:

$$\begin{aligned} H_A(d,d) &= H_B(d,d) = -8 \\ H_A(n,n) &= H_B(n,n) = -1 \\ H_A(d,n) &= H_B(n,d) = 0 \\ H_A(n,d) &= H_B(d,n) = -10 \end{aligned}$$

To put a narrative on this game, imagine that A and B are jailed and isolated from each other. They are accused of the same crime but no solid proof can yet be established for the charges. Both A and B can either admit the charge and denounce the other for complicity (choice d), or deny any implication to the crime (choice n). If A and B denounce each other, then they will be both sentenced to 8 years in prison. If both of them deny the charge, then they will be judged on the basis of existing proofs, and (probably) they will be condemned to a sentence of 1 year in prison for minor crimes. If one confesses/denounces the other, while the latter denies his participation in the crime, then the one who confessed will be released, but the other will be designated as the sole guilty, and consequently he will be sentenced to 10 years in prison. It is easy to see that in this game, situation (d, d) is a Nash equilibrium, while the remaining three $(d, n), (n, d), (n, n)$ are Pareto stable (optima)[7].

14.1.2. *Complexity classes mentioned throughout the chapter*

For the sake of completeness of presentation, we give in this section some short definitions of complexity classes that will be mentioned later on. For more details, see [PAP 94a] and also [ANG 05, GAR 79, PAS 04, PAS 06]. For basic notions of formal language theory like "alphabet", "Turing machine", etc., see [LEW 81].

Let Σ be a finite alphabet. We note by **NP** (resp. **NEXP**) the class of languages $L \subseteq \Sigma^*$ (where Σ^* is the Kleene's star closure of Σ) for which, given a string x of characters from Σ, it can be checked in polynomial time whether $x \in L$. We note by **P** (resp. **EXP**) the class of languages L for which an $x \in L$ (or the confirmation that L is empty) can be calculated in polynomial (resp., exponential) time.

Let $R \subseteq \Sigma^* \times \Sigma^*$ be a binary relation over the strings of characters in Σ. R is called *polynomial-time decidable* if there is a deterministic Turing machine which can

7. The first detailed analysis of this type of games is published in [LUC 57]; the monography [RAP 65] is devoted to Prisoner's dilemma.

decide in polynomial time; R is called polynomially balanced if there is $k \leqslant 1$ such that, for all $(x, y) \in R$: $|y| \leqslant |x|^k$.

For R polynomial-time decidable and polynomially balanced, the *counting problem* associated with R is the following: "given $x \in \Sigma^*$, calculate the number of y such that $(x, y) \in R$". Class $\sharp$**P** is the set of counting problems associated with relations R of this type.

A necessary and sufficient condition for a language $L \subseteq \Sigma^*$ to belong to **NP** is the existence of a polynomial-time decidable and polynomially balanced relation R_L over strings, such that $L = \{x : \exists y, (x, y) \in R_L\}$ [PAP 94a]. We can define then, for all languages L in **NP**, a *function problem associated with* L as the following problem. "Let R_L be a relation satisfying the previous condition. Given $x \in \Sigma^*$, find y such that $(x, y) \in R_L$ or return "no" if there is no such y". Whenever it can be shown that R_L is such that, for all $x \in \Sigma^*$, there is always a y such that $(x, y) \in R_L$, the associated function problem is then called *total function problem associated with* L. The class of all function problems associated with some language in **NP** is denoted by **FNP**. The class of total function problems is denoted by **TF**. The subclass of total function problems associated with some language in **NP** is denoted by **TFNP**.

An important class of "total" problems, for which we search for a solution but the existence of at least one is guaranteed, is the class of total problems that can be reduced to the following problem, called END OF LINE and defined as follows: "let P and C be two logic circuits with n input bits and n output bits, such that $P(0, \ldots, 0) = (0, \ldots, 0) \neq C(0, \ldots, 0)$; find an input $x \in \{0, 1\}^n$ such that $P(C(x)) \neq x$ or $C(P(x)) \neq x \neq (0, \ldots, 0)$".

Intuitively, the problem END OF LINE is illustrated by a directed graph $G(V, A)$ whose vertex set is $V = \{0, 1\}^n$ and directed edge set is $A = \{(x, y) : P(y = C(x)) = x\}$. The in- and out-degree of G is at most one, and G has at least one source, $(0, \ldots, 0)$. We search for another source or a sink. Class **PPAD** includes then all total problems whose totality can be established by the combinatorial argument: "if a directed graph has a vertex with non-zero difference between in- and out-degree, then it must have another one" [DAS 05].

Given an optimization problem Π, a problem of searching for *local* optima can be defined by associating with the set of feasible solutions of Π a neighborhood structure, and then searching for a solution which is optimal in comparison with its neighbors (see also [MON 03]). The class **PLS** is the set of local research problems for which there exist three polynomial algorithms $\mathtt{A}_\Pi$, $\mathtt{B}_\Pi$ and $\mathtt{C}_\Pi$ such that:

- given an input x, algorithm $\mathtt{A}_\Pi$ decides whether x is an instance of Π and, if this is the case, calculates a feasible solution for this instance;

- given an instance x of Π and a candidate solution s for x, algorithm $\mathtt{B}_\Pi$ decides if s is a feasible solution for x and, if this is the case, calculates its value;

- given an instance x of Π and a feasible solution s for x, algorithm $\mathtt{C}_\Pi$ decides if s is a local optimum (under the neighborhood structure associated with Π); otherwise, calculates a neighbor solution s' of strictly better value than the value of s.

14.2. Nash equilibria

A. Cournot, in Chapter 7 of his famous *Recherches*, had already formulated an equilibrium concept for some specific two-player games, the direct generalisation of which is the Nash equilibrium [LEO 94]. A major part of contemporary research on the theory of non-cooperative games, above all on algorithmics of non-cooperative games, is devoted to this optimality concept and its variants. Most of the results presented in the present chapter deal with Nash equilibria.

In its current form, this concept has been introduced by J. Nash in [NAS 50b]. In this note, the author shows that, for a quite broad family of games, the set of Nash equilibria is guaranteed to be non-empty. Based on this fundamental propriety, he outlines later [NAS 50a] a vast program of reduction of cooperative to non-cooperative games.

Intuitively, a situation is a Nash equilibrium if no player can get profit if he modifies in a "unilateral" fashion his choice (i.e., while the other players stay in their choices that define this situation).

In order to introduce the principal results of Nash, let us recall some mathematical definitions.

DEFINITION 14.3.– *Let $\mathbf{Z}$ be a linear space. A set $Z \subseteq \mathbf{Z}$ is* convex *if, for all $z', z'' \in Z$ and $\lambda \in [0,1]$, it is $\lambda z' + (1-\lambda)z'' \in Z$.*

DEFINITION 14.4.– *Let $\mathbf{Z}$ be a linear space. A function $f : \mathbf{Z} \longrightarrow \mathbb{R}$ is called* concave (resp., convex) *if, for all $z', z'' \in \mathbf{Z}$ and $\lambda \in [0,1]$, it is:*

$$\lambda f(z') + (1-\lambda) f(z'') \leqslant (\textit{resp.},\ \geqslant) f(\lambda z' + (1-\lambda)z'') \qquad [14.3]$$

Function f is strictly concave (strictly convex) if inequality [14.3] is strict for $z' \neq z''$ and $0 < \lambda < 1$. Function f is quasi-concave (resp., quasi-convex) if the set $\{z : f(z) \geqslant (\textit{resp.},\ \leqslant)\alpha\}$ is convex for all $\alpha \in \mathbb{R}$.

A game in form [14.2] is called *concave* (resp., *quasi-concave*) if, for each player i, his set of strategies X_i is a convex subset of a linear space and if, whenever we fix the choice of the other players $I \setminus \{i\}$ in an arbitrary way, his payoff function H_i

becomes a concave (resp., quasi-concave) function $X_i \longrightarrow \mathbb{R}$. We must note here that it is not necessary to have all of the H_is concave or quasi-concave.

Now Theorem 14.1 can be stated.

THEOREM 14.1.– *Let a game Γ (in form [14.2]) be quasi-concave. If strategy sets X_i of all players are compact (in some topology) and if payoff functions H_i are continuous (in the corresponding topology) over the situations, then Γ has a Nash equilibrium.*

Proof. Let $\mathbf{x}$ be an arbitrary situation. The situation obtained if in $\mathbf{x}$ the choice of i is replaced by x_i will be denoted by $\mathbf{x}\|x_i$. Let $\mathbf{c}_i(\mathbf{x}) = \operatorname{argmax}_{x_i} H_i(\mathbf{x}\|x_i)$, i.e., the set of choices of player i for which:

$$H_i\left(\mathbf{x}\|x_i^*\right) = \max_{x_i \in X_i} \left\{H_i\left(\mathbf{x}\|x_i\right)\right\}$$

Because of the compacity of X_i and the continuity of H_i, $\mathbf{c}_i(\mathbf{x})$ is non-empty; moreover, because of the quasi-concavity of H_i over x_i, $\mathbf{c}_i(\mathbf{x})$ is convex (and closed, by continuity of H_i).

The correspondence between $\mathbf{x}$ and $\mathbf{c}_i(\mathbf{x})$ is also closed, i.e., for all converging sequences of situations: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)} \rightarrow \mathbf{x}^{(0)}$ and all converging sequences of choice of i: i: $y_i^{(1)}, \ldots, y_i^{(m)} \rightarrow y_i^{(0)}$ such that, for $m = 1, \ldots$:

$$y_i^{(m)} \in \mathbf{c}_i\left(\mathbf{x}^{(m)}\right) \qquad [14.4]$$

the following inclusion holds:

$$y_i^{(0)} \in \mathbf{c}_i\left(\mathbf{x}^{(0)}\right) \qquad [14.5]$$

[14.5] can be shown by selecting an arbitrary x_i; expression [14.4] means that for all m:

$$H_i\left(\mathbf{x}^{(m)}\|y_i^{(m)}\right) \geqslant H_i\left(\mathbf{x}^{(m)}\|x_i\right)$$

and hence, passing to the limit:

$$H_i\left(\mathbf{x}^{(0)}\|y_i^{(0)}\right) \geqslant H_i\left(\mathbf{x}^{(0)}\|x_i\right)$$

which implies the validity of [14.5] because the latter holds for any x_i.

Now consider the application $\mathbf{c} : \mathbf{X} \longrightarrow 2^{\mathbf{X}}$ establishing the correspondence between each situation $\mathbf{x}$ and the set of situations $\mathbf{c}(\mathbf{x}) = \prod_{i \in I} \mathbf{c}_i(\mathbf{x})$. Set $\mathbf{c}(\mathbf{x})$ is a Cartesian product of non-empty, compact and convex sets - thus it is also non-empty, compact and convex; let us recall that if $\mathbf{c}_i(\mathbf{x})$ are closed, we get that $c(x)$ is also closed.

Thus, application $\mathbf{c}$ satisfies the conditions of the Kakutani Theorem [BOR 85], according to which $\mathbf{c}$ has a fixpoint, i.e., there is $\mathbf{x}^* \in \mathbf{X}$ such that $\mathbf{x}^* \in \mathbf{c}(\mathbf{x}^*)$. But, for all choice of x_i^* component of $\mathbf{x}^*$, there will be also $x_i^* \in \mathbf{c}_i(\mathbf{x}^*)$, i.e., for all players i:

$$\forall x_i \in X_i, H_i\left(\mathbf{x}^*\right) \geqslant H_i\left(\mathbf{x}^* \| x_i\right)$$

Hence, the Nash equilibria set for a game as those of Theorem 14.1, are non-empty. ∎

This result can still be obtained if the concavity condition of Γ is relaxed and if the continuity condition for the payoff functions is slightly modified, as stated in the following theorem, that we mention without the proof.

THEOREM 14.2.– *Let Γ be a concave game (in form [14.2]). If the set of players' strategies X_i are compact (in some topology) and if the payoff functions are continuous (in the correponding topology) whenever the choice of all other players except i is fixed in an arbitrary fashion and the choice of i varies freely, and if $\sum_{i \in I} H_i(\mathbf{x})$ is continuous over $\mathbf{x}$, then Γ has a Nash equilibrium.*

14.3. Mixed extension of a game and Nash equilibria

For every optimality concept φ, an interesting question is whether it is possible to transform a game Γ into a game $\mathcal{T}(\Gamma)$, in such a way that $\varphi\Gamma \subseteq \varphi\mathcal{T}(\Gamma)$. The importance of Nash equilibria comes precisely from the fact that there is an intuitive transformation $\mathcal{T}$ transforming every game Γ of finite player set and strategies into a game that satisfies the conditions of Theorem 14.1, i.e. into a game having (at least) one Nash equilibrium.

DEFINITION 14.5.– *Let $\Gamma = \langle I, \{X_i\}_{i \in I}, \{H_i\}_{i \in I} \rangle$ be a game with finite I and X_i. We call mixed extension of Γ a game $G = \langle I, \{X_i'\}_{i \in I}, \{\overline{H_i}\}_{i \in I} \rangle$ in which the X_i' are independent probability measures, each one defined by the X-algebra 2^{X_i} over X_i and $\overline{H_i}$ are expectancies of H_i. X_i' are called mixed strategies sets of Γ and their elements are called mixed choices.*

Intuitively, a mixed extension of Γ is the game that yields when players play probabilities distributions over their strategies sets rather than choices themselves; "classic" situations appear whenever every player i plays a degenerate probability distribution

over X_i, with x chosen with probability one and every other choice of i chosen with zero probability.

Thus, every situation of Γ is also a situation of G and for these situations the payoff values are the same in G as in Γ. It is easy to see that the X_i' and the $\overline{H_i}$ all satisfy the conditions of Theorem 14.1 (in fact, every X_i' is a simplex).

Theorem 14.3 expresses some basic proprieties of mixed extensions of games, related to Nash equilibria.

THEOREM 14.3.– *Every mixed extension of a game as* Γ *of definition has a Nash equilibrium, and all Nash equilibria of* Γ *are also Nash equilibria of its mixed extension.*

Proof. According to the previous comment and Definition 14.5 of G, the sets:

$$X_i' = \left\{ p_i : p_i\left(x_i\right) \geqslant 0 \text{ for } x_i \in X_i \text{ and } \sum_{x_i \in X_i} p_i\left(x_i\right) = 1 \right\}$$

are simplices in Euclidean spaces of finite dimension and the topologies in such spaces are compact. In this case, spaces are linear and hence sets X_i' are convex.

For all mixed situations $\mathbf{x}'$, it is:

$$\overline{H_i}\left(\mathbf{x}'\right) = \sum_{x_1 \in X_1} \cdots \sum_{x_{|I|} \in X_{|I|}} H_i\left(x_1, \ldots, x_{|I|}\right) p_1\left(x_1\right) \ldots p_{|I|}\left(x_{|I|}\right)$$

Hence, functions $\overline{H_i} : \mathbf{X}' \longrightarrow \mathbb{R}$ are linear in every one of their arguments and consequently they are concave functions. ∎

Nash equilibria of the mixed extension of a game Γ are often called *Nash equilibria of* Γ *in mixed strategies*; those which are not Nash equilibria of Γ are called *mixed Nash equilibria* of Γ, while those which are Nash equilibria of Γ are called *pure Nash equilibria of* Γ.

Equilibria sets of games of Definition 14.5 in mixed strategies contain a finite number of connected components.

14.4. Algorithmic problems

Finding an equilibrium in mixed strategies is a total problem: there is always one, no matter what the game is. However, in the existence proof no constructive arguments are used, at least in the "polynomial" sense of this term. It should also be mentioned

that Nash had already given an example of a three-player game whose only equilibrium is in irrationals (at the same time, no irrational numbers enter the description of this game). Therefore, any algorithm calculating Nash equilibria for general games would rather have to calculate a finite-description, rational ε-approximate equilibrium, that is a situation such that no player can improve his payoff more than an ε additive factor, being part of the algorithm's input.

The complexity of an algorithm for the problem formulated in this way would then be stated in terms of order of magnitude of a function of the game's description size and of the precision parameter ε.

14.4.1. *Games of succinct description*

In order to specify a game $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ with finite player and strategy sets, $|I| \prod_{i\in I} |X_i|$ values are needed for describing the payoffs, plus $\sum_{i\in I} |X_i|$ values for the strategy sets, plus the number of players $|I|$. In order to further specify an instance of the problem of searching a mixed equilibrium in Γ, a value for the precision factor ε is also needed. In this representation form, the lengthier part is in fact the payoff description.

However, it is often possible to obtain a succinct representation of the H_i, for instance, whenever every payoff function can be expressed by a short and polynomially computable formula, or in the case where the payoffs are in fact independent of the choice of the whole set of players except for a small part of it.

In such cases, a directed *influence graph* $G_\Gamma(V, E)$ [KEA 01] (with $V(G_\Gamma) = I$ and $E(G_\Gamma) = \{(i, j) : H_i$ depends on the strategy chosen by $j\}$) can be associated with Γ. Then, Γ is called *a game defined on the graph* G_Γ.

In general, we can always define a game on a complete directed graph; but, for the games for which a low connectivity graph is enough, this representation can be seen as a kind of factorization in smaller "local" games. If the graph is of bounded degree Δ, then, for the description of H_i, no more than $|I|m^{\Delta+1}$ values are needed, with $m = \max_{i\in I}\{|X_i|\}$.

14.4.2. *Results related to the computational complexity of a mixed equilibrium*

It is well-known that a Nash equilibrium for a game can be represented as the solution of a system of polynomial inequalities [VOR 98]. There exist algorithms solving such systems in time that is polynomial on $1/\varepsilon$ and $\log(M)(mn)^{m^2n^2}$, M being the absolute maximum of the players' payoffs, m the maximum number of choices for all players and $n = |I|$ [GRI 88].

The question of the existence of a polynomial time algorithm for calculating a mixed Nash equilibrium remains open to this day (see [PAP 05] for some recent developments on this issue). In most of the works dealing with the subject, this question is addressed in the following form: given a succinct description of a game, find an algorithm which calculates a mixed Nash ε-equilibrium of the game and is polynomial on $1/\varepsilon$, on the number of players and on the maximum number of choices for any player.

[BUB 79, VOR 98] show a transformation of any game of n players to some game of 3 players, such that there is a bijection between the equilibria of the transformed and the equilibria of the initial games, which make it possible to find the latter from the former in polynomial time. In this transformation, the sets of strategies of the three-player game are of exponential size with respect to the size of those of the initial general game.

Two other equilibria-preserving transformations of general games into three-player games have been presented in [CHE 05, DAS 05]. They are based on that shown in [GOL 05]. In this paper, it is first shown how to transform a game defined on a graph of n vertices and degree bounded by Δ into a game of Δ^2 players (a "special" player plus $\Delta^2 - 1$ players corresponding to chromatic classes of the graph obtained after having put over the original graph one directed edge (i, k) for every path $(i, j), (j, k)$ in the original graph). It is then shown how to obtain, from a general game, a game defined on a graph of degree 3, where every vertex-player has two choices. Combining these reductions yields that the problem of finding an equilibrium in a three-player game is, like in the general case, **PPAD**-complete[8]. Thus, the existence of a polynomial algorithm[9] for this problem seems highly improbable.

In [KEA 01], the authors present an algorithm that finds a mixed equilibrium of a game defined on a tree where every player is given two choices ("zero" and "one"). The algorithm makes two searches of the tree's nodes, one from the leaves to the root, and a second from the root towards the leaves. During the first search, every vertex transfer to its father a sequence of numbers defining a partition of the interval $[0, 1]$, which determines its choice for each choice of the father in every part of the partition, in order to attain a Nash equilibrium given the father's choice. This sequence of numbers – the "breakpoints" – of the $[0, 1]$ has been calculated from the data transferred by the children of the vertex. When the root is reached, it determines a choice that is transfered to the root's children; during the search towards the leaves, the vertices determine their choice depending on their sequence of numbers and the

8. This is the class of problems in **TF** that can be shown to belong to the latter by a parity argument (of the kind: if a directed graph has a vertex with a non-zero difference between its in- and out-degree, then it must have another such vertex).

9. In the sense discussed above.

choice announced by their fathers. The authors claim that this algorithm is polynomial for bounded degree trees[10]. In [VIC 02] the algorithm is generalized for other graph classes.

In the case of zero sum, two-player games (i.e., whenever for all $(x_1, x_2) \in \mathbf{X}$, $H_1(x_1, x_2) + H_2(x_1, x_2) = 0$), the problem of searching a mixed Nash equilibrium can be formulated as a linear program, hence it is polynomial-time solvable.

The question of general two-player games remains open [KOL 92]. A solution method has been proposed in [LEM 64]; however, this method can be exponential even for some zero-sum games [SAV 04].

Quite recently, Chen and Deng [CHE 05B] have shown the searching of Nash equilibria in general two-player games to be **PPAD**-complete, as for the case of general games. In [SIN 00], some steepest-descent heuristics are also proposed for this problem.

As for negative results on the complexity issue, [GIL 89] shows some in this vein. In fact, the central part of the assertions shown in this paper is that searching a mixed Nash equilibrium with specified properties often becomes **NP**-complete, even for symmetrical two-player games[11].

In order to obtain these results, the authors of the paper have built a reduction of a CNF formula to a symmetrical two-player game, i.e., such that in any situation (x_1, x_2), the payoff of one of the players is equal to the payoff of the other in situation (x_2, x_1).

In what follows, we present the reduction given in [CON 03] which is based on that in [GIL 89].

Let ϕ be a Boolean formula in conjunctive normal form. Let V be its set of variables (with $|V| = n$), L the set of literals corresponding to elements of V and let C be the set of its clauses. The function $v : L \longrightarrow V$ returns the variable that corresponds to a literal, i.e., $v(z_i) = v(-z_i) = z_i$. $\Gamma(\phi)$ is defined as the following symmetrical game of two players ($I = \{1, 2\}$): $X = X_1 = X_2 = L \cup V \cup C \cup \{f\}$; the payoff

10. However, a more careful analysis in [ELK 06] shows that this algorithm becomes polynomial under more powerful constraints.

11. Games for which $H_1(x_1, x_2) = H_2(x_2, x_1)$, for all $x_1 \in X_1$, $x_2 \in X_2$.

functions H_1, H_2 are defined as:

$$\begin{cases} H_1\left(l^1, l^2\right) = H_2\left(l^2, l^1\right) = 1 & \forall l^1, l^2 \in L \text{ with } l^1 \neq -l^2 \\ H_1\left(l, -l\right) = H_2(-l, l) = -2 & \forall l \in L \\ H_1(l, z) = H_2(z, l) = -2 & \forall l \in L, z \in X \setminus L \\ H_1(v, l) = H_2(l, z) = 2 & \forall v \in V, l \in L \text{ with } v(l) \neq v \\ H_1(v, l) = H_2(l, v) = 2 - n & \forall v \in V, l \in L \text{ with } v(l) = v \\ H_1(v, z) = H_2(z, v) = -2 & \forall v \in z \in X \setminus L \\ H_1(c, l) = H_2(l, c) = 2 & \forall c \in C, l \in L \text{ with } l \notin c \\ H_1(c, l) = H_2(l, c) = 2 - n & \forall c \in C, l \in L \text{ with } l \in c \\ H_1(c, z) = H_2(z, c) = -2 & \forall c \in C, z \in X \setminus L \\ H_1(f, f) = H_2(f, f) = 0 & \\ H_1(f, z) = H_2(z, f) = 1 & \forall z \in X \setminus \{f\} \end{cases}$$

Now the main theorem can be stated as follows:

THEOREM 14.4.– *The game $\Gamma(\phi)$ has a mixed Nash equilibrium in which both players play l_i with probability $1/n$ of getting payoff 1 if and only if $(l_1, l_2, \ldots, l_n)$ (with $v(l_i) = z_i$) satisfies ϕ. Moreover, there is only one more Nash equilibrium (in mixed strategies), when both players play f, each one receiving payoff 0.*

Proof. We start by showing that the situations mentioned in the theorem are indeed Nash equilibria. If (with $v(l_i) = z_i$) satisfies ϕ and player 2 plays l_i with probability $1/n$, then player 1 can choose one of these l_i and be paid 1. Notice also that:

- if 1 plays the negation of one of these l_i, then he gets as payoff:

$$\frac{1}{n} \times (-2) + \frac{n-1}{n} \times 1 < 1$$

- given that for some of the l_i that 2 plays sometimes, it is $v(l_i) = v$, if 1 plays a variable v he will be given:

$$\frac{1}{n} \times (2 - n) + \frac{n-1}{n} \times 2 = 1$$

- if 1 plays a clause c he will get a payoff at most:

$$\frac{1}{n} \times (2 - n) + \frac{n-1}{n} \times 2 = 1$$

because one of the l_i played sometimes by 2 is in clause c, according to the fact that l_i satisfies ϕ;

- finally, if 1 plays f he will be handed 1.

It follows that playing any of the l_i played sometimes by 2 is an optimal response for 1; hence if both players play each one of these l_i with probability $1/n$, a Nash

equilibrium is reached. Clearly, the situation where both players play f is also a Nash equilibrium, given the situation with one of the players playing anything but f whenever the other plays f, will give him a payoff equal to -2.

It remains to show that there is no other Nash equilibrium: if the other player plays always f, the only "best response" for the player is to play f because any other choice would give payoff -2. In the opposite case, given a mixed strategy of the other player with zero probability of playing f, we consider the payoff of the first player. If this payoff is less than 1, then he should rather play f (which gives a payoff equal to 1 when the other does not play at all f, which is a choice that is better than his former choice whenever the first player plays f). Hence, these situations cannot be Nash equilibria.

In the case where a player plays f while the other does not play f, there is no Nash equilibrium; so, suppose that both players play f with probability less than 1. Let us consider the "expected social welfare" $\overline{H_1 + H_2}$ when no player plays f. It is easy to see that no such situation can give a social welfare greater than 2. Notice also that every situation in which one of the players plays an element of V or C implies a social welfare which is strictly less than 2. It follows that if one of the players plays an element of V or C, and no player plays f, then the expected social welfare is strictly less than 2.

By linearity of expectancy, it follows that the expected payoff of at least one of the players is strictly less than 1 when no player plays f and, according to the previous reasoning, this player could improve his payoff by playing f rather than any probability distribution over choices which are different from f. This suffices to conclude that no element of V or C is played in a Nash equilibrium.

Thus, we can suppose that both players put non-zero probabilities only over elements of $L \cup \{f\}$. If the other player puts positive probabilities over f, then it is better for the first player to choose f than any element of L (because both players will be given payoff 1 if the other player plays an element of L, while playing f is a best response if the other plays f). It follows that the only equilibrium in which one of the players plays f is the situation in which both players play f.

Now we can suppose that both players put positive probabilities over elements of L exclusively. Suppose that, for a particular player, the probability of playing l or $-l$ is less than $1/n$. In that case the expected payoff of the other player when he plays $v(l)$ is strictly less than:

$$\frac{1}{n} \times (2 - n) + \frac{n-1}{n} \times 2 = 1$$

so this cannot be a Nash equilibrium, yielding that in a Nash equilibrium, for all $l \in L$, the probability for a player to play l or $-l$ is exactly $1/n$.

If player 1 puts positive probabilities over an element of L and player 2 puts positive probabilities over the negation of the same element, both players get an expected payoff less than 1 and they would be better off by playing f. Thus, in a Nash equilibrium, if player 1 plays l with some probability, then player 2 has to play l with probability $1/n$. This implies that for each variable there exists exactly one of its literals which is played by both players with probability $1/n$. It follows that, in every Nash equilibrium (other than the situation where both players play f), the literals played sometimes correspond to a truth assignment.

Finally, we show that if this assignment does not satisfy ϕ, then the corresponding situation is not a Nash equilibrium. Let $c \in C$ be a clause that is not satisfied by the assignment, i.e., none of its literals is ever played. Then, playing c would yield a payoff equal to 2 and both players should play it. ∎

According to Theorem 14.4 shown above, there is a Nash equilibrium in $\Gamma(\phi)$ in which every player gets 1 as payoff if and only if ϕ is satisfiable; the only other equilibrium is when both players play f and they get 0 as payoff. However, every "sensible" definition of "welfare optimization" should rather push to prefer equilibria of the first type. In this sense, deciding if there is a "good" equilibrium is a hard problem. The following corollaries develop a spectrum of similar aspects of the problem (each corollary is directly deduced by Theorem 14.4).

COROLLARY 14.1.– Even in symmetrical two-player games, it is **NP**-hard to decide the existence of a Nash equilibrium $\mathbf{x}$ yielding an expected social welfare $\sum_{i \in |I|} \bar{H}_i(\mathbf{x})$ which is at least equal to k, even if k is the maximum attainable value for the social welfare in the game.

COROLLARY 14.2.– Even in symmetrical two-player games, it is **NP**-hard to decide if:

1) there is a Nash equilibrium in which all players get payoffs at least k, even if k is the greatest value for which there is a probability distribution over the situations of the game such that all players have a total expected payoff of at least k;

2) there is a Nash equilibrium that is also Pareto-stable;

3) there is a Nash equilibrium where one of the players gets a payoff of at least k;

4) there is more than one Nash equilibrium;

5) there is a Nash equilibrium in which one of the players sometimes plays a particular choice $x \in X_1$;

6) there is a Nash equilibrium in which one player never plays a particular choice $x \in X_1$.

Corollary 14.2 shows that it is hard to obtain specific information about Nash equilibria in a game. Point 4 and softer versions of 1, 5 and 6 have been first shown in [GIL 89].

As a conclusion to this section, we mention a remarkable result of [DAS 05] concerning Nash equilibria in mixed strategies for games defined on a d-dimensional grid. It is shown that it can be decided in polynomial time whether a game defined on a 1-dimensional grid has a pure equilibrium, while the same problem for $d \geqslant 2$ becomes **NEXP**-complete.

14.4.3. *Counting the number of equilibria in a mixed strategies game*

As it is already shown (page 348), the number of connected components[12] of the Nash equilibria set in a mixed game is finite. The question of enumerating these components was first studied from the algorithmic complexity point of view in [MCL 97, MCL 02]. Theorem 14.4 can be directly used to show that deciding this number for a given game is hard.

PROPOSITION 14.1.– *Even in symmetrical two-player games, counting the number of Nash equilibria is ♯**P**-hard.*

Proof. The number of Nash equilibria in the game $\Gamma(\phi)$ of Theorem 14.4 is the number of satisfying assignments of ϕ plus one. But it is well-known that counting the number of satisfying assignments of a formula in CNF is ♯**P**-hard [VOR 98]. ∎

Every Nash equilibrium of $\Gamma(\phi)$ is a connected set, hence the number of connected components is equal to the number of satisfying assignments of ϕ plus one. Thus, the following Corollary 14.3 holds:

COROLLARY 14.3.– Even in symmetrical two-player games, counting the number of maximal connected subsets of the Nash equilibria set is ♯**P**-hard.

It is worth mentioning here that the most interesting results on ♯**P**-hardness are those whose corresponding decision and searching questions are easy (such as, for instance, the ♯**P**-hardness of counting perfect matchings in a bipartite graph, which is reduced to the calculation of the permanent of a matrix).

14.5. Potential games

The representation problem discussed in section 14.4 is simplified to a great extent in the case of *potential games* introduced in this section. This class of games models a broad spectrum of optimization problems; they have always pure Nash equilibria coinciding with the local optima of their potential function.

12. Of the maximal connected subsets.

14.5.1. *Definitions*

Let $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ be a game of finite sets of players and strategies. For a set $K \subseteq I$, we denote by $-K$ the complementary set of K with respect to I and by X_K the Cartesian product $\prod_{i\in K} X_i$. For singletons $\{i\}$, $X_{\{-i\}}$ will be denoted by X_{-i}. Let us assume a situation $\mathbf{x} = (x_1, \ldots, x_n)$. The situation:

$$(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_n) = \mathbf{x}\|z$$

is noted by (x_{-i}, z).

A function $P : \mathbf{X} \longrightarrow \mathbb{R}$ is an *ordinal potential* for Γ if, for all $i \in I$ and for all $x_{-i} \in X_{-i}$:

$$H_i(x_{-i}, x) - H_i(x_{-i}, z) > 0 \Longleftrightarrow P(a_{-i}, x) - P(a_{-i}, z) > 0, \quad \forall x, z \in X_i$$

DEFINITION 14.6.– Γ *is an ordinal potential game if it admits an ordinal potential (i.e., it is possible to define an ordinal potential for* Γ*).*

Let $w = (\omega_i)_{i\in I}$ be a vector of positive numbers that will be called the *weights*. A function $P : \mathbf{X} \longrightarrow \mathbb{R}$ is a *ω-potential* for Γ if, for all $i \in I$ and all $s_{-i} \in X_{-i}$, it is

$$H_i(s_{-i}, x) - H_i(s_{-i}, z) = \omega_i (P(s_{-i}, x) - P(s_{-i}, z))$$

for all $x, z \in X_i$.

DEFINITION 14.7.– Γ *is a ω-potential game if it admits a ω-potential. P is also called a* weighted potential *and* Γ *a* weighted potential game.

A function $P : X \longrightarrow \mathcal{R}$ is an *exact potential* (or, simply, a *potential*) if it is a weighted potential for Γ with $\omega_i = 1$, for all $i \in I$.

DEFINITION 14.8.– Γ *is an* exact potential game *(or, simply a* potential game*) if it admits a potential.*

14.5.2. *Properties*

The existence of an ordinal potential for a game makes it possible to transform it into an equivalent "simplified" game, with the same equilibria set as the original game.

LEMMA 14.1.– *Let P be an ordinal potential for the game:*

$$\Gamma = \langle I = \{1, \ldots, n\}, \{X_1, \ldots, X_n\}, \{H_1, \ldots, H_n\}\rangle$$

The equilibria set of Γ *concides with the equilibria set of:*

$$G = \langle \{1, \ldots, n\}, \{X_1, \ldots, X_n\}, \{H_1 \equiv H_2 \ldots \equiv H_n \equiv P\} \rangle$$

i.e., the game obtained if the payoff functions of Γ *are replaced by the potential function.*

In other words, $s \in X$ is a Nash equilibrium for Γ if and only if, for all $i \in I$ such that $x \in X_i$:

$$P(s) \geqslant P(s_{-i}, x) \qquad [14.6]$$

Consequently, if P admits a maximal value over $\mathbf{X}$, then G has an equilibrium in pure strategies. But this is the case, according to the construction of P. Thus, Theorem 14.5 can be stated.

THEOREM 14.5.– *Every ordinal potential game has a pure Nash equilibrium.*

Theorem 14.5 can be further generalized, in order to completely characterize the class of games with pure Nash equilibria.

Starting from a finite game Γ, we can build a directed graph having as vertex set the situations $\mathbf{X}$ of Γ and all the directed edges $(\mathbf{s}, \mathbf{s}')$ for situations $\mathbf{s}$ and $\mathbf{s}'$ differing only in one component i and such that $H_i(\mathbf{s}') > H_i(\mathbf{s})$. If the graph has no circuit, we say that the *Nash dynamics* converge for Γ.

PROPOSITION 14.2.– *If the Nash dynamics converge for* Γ*, then* Γ *has a pure Nash equilibrium.*

Proof. The sinks of the graph are the Nash equilibria of the game. ∎

A *path* in this graph is the sequence $\gamma = (y^0, y^1, \ldots)$ such that, for all $k \geqslant 1$, there is a unique player i such that $y^k = (y^{k-1}_{-i}, x)$ for $x \neq y^{k-1}_i$ in X_i; y^0 is called the *origin point* of γ and if γ is finite, then its last element is called the *terminal point* of γ. A path $\gamma = (y^0, y^1, \ldots)$ is an *improvement path* for Γ if, for all $k \geqslant 1$, it is $H_i(y^k) > H_i(y^{k-1})$ with i being the only player changing strategy at step k. We say that Γ has the finite improvement property (FIP) if every improvement path is finite.

LEMMA 14.2.– *In every ordinal potential game the FIP holds.*

Proof. For all improvement paths $\gamma = (y^0, y^1, \ldots)$, we have, according to Lemma 14.1:

$$P(y^0) < P(y^1) < \ldots$$

X being a finite set, the sequence γ must be also finite. ∎

Clearly, saying that a game has the FIP and saying that the Nash dynamics converge for this game, are equivalent statements. In other words, every maximal improvement path must terminate to a Nash equilibrium.

At this point, an interesting comment is that having the FIP is not equivalent to admitting an ordinal potential.

A function $P : \mathbf{X} \longrightarrow \mathbb{R}$ is a *generalized ordinal potential* for Γ if, for all $i \in I$, for all $s_{-i} \in X_{-i}$ and all $x, z \in X_i$, it is:

$$H_i(s_{-i}, x) - H_i(s_{-i}, z) > 0 \Longrightarrow P(s_{-i}, x) - P(s_{-i}, z) > 0$$

THEOREM 14.6.– *A finite game Γ has the FIP if and only if Γ admits a generalized ordinal potential.*

Proof. Let Γ be a game with the FIP. Define the binary relation $\succ$ on $\mathbf{X}$ as follows: $\mathbf{x} \succ \mathbf{y}$ if and only if $\mathbf{x} \neq \mathbf{y}$ and there is no finite improvement path γ with initial point x and terminal point y. The FIP implies transitivity of the relation $\succ$.

Let $\mathbf{Z} \subseteq \mathbf{X}$. We say that $\mathbf{Z}$ is *represented* if there is a function $Q : \mathbf{Z} \longrightarrow \mathbb{R}$ such that, for all $\mathbf{x}, \mathbf{y} \in \mathbf{Z}$, $\mathbf{x} \succ \mathbf{y}$ implies that $Q(\mathbf{x}) > Q(\mathbf{y})$. Let $\mathbf{Z}$ be a represented and maximal subset of $\mathbf{X}$. We are going to show that $\mathbf{Z} = \mathbf{X}$.

Suppose that $\mathbf{x} \notin \mathbf{Z}$. If $\mathbf{x} \succ \mathbf{z}$ for all, then Q can be extended to $\mathbf{Z} \cup \{\mathbf{x}\}$, by defining $Q(\mathbf{x}) = 1 + \max_{\mathbf{z} \in \mathbf{Z}} Q(\mathbf{z})$, which contradicts the maximality of $\mathbf{Z}$. If $\mathbf{z} \succ \mathbf{x}$ for all $\mathbf{z} \in \mathbf{Z}$, then Q can be extended to $\mathbf{Z} \cup \{\mathbf{x}\}$ by defining $Q(\mathbf{x}) = \min_{\mathbf{z} \in \mathbf{Z}} Q(\mathbf{z}) - 1$, which also contradicts the maximality of $\mathbf{Z}$.

Finally, if neither $\mathbf{x} \succ \mathbf{z}$ nor $\mathbf{z} \succ \mathbf{x}$, then Q can be extended and the maximality of $\mathbf{Z}$ is contradicted by defining $Q(\mathbf{x}) = (a + b)/2$ with $a = \max\{Q(\mathbf{z}) : \mathbf{z} \in \mathbf{Z}, \mathbf{x} \succ \mathbf{z}\}$ and $b = \min\{Q(\mathbf{z}) \in \mathbf{Z}, \mathbf{z} \succ \mathbf{x}\}$. Thus, $\mathbf{X}$ is represented (so it admits a generalized ordinal potential Q). ∎

COROLLARY 14.4.– Let Γ be a finite game with the FIP. Suppose also that, for all $i \in I$ and for all $s_{-i} \in X_{-i}$, $H_i(s_{-i}, x) \neq H_i(s_{-i}, z), \forall x \neq z \in X_i$. Then Γ has an ordinal potential.

Indeed, it is enough to notice that the supplementary condition for Γ implies that every generalized ordinal potential for Γ is also an ordinal potential for Γ. Hence, the corollary is straightforward from Lemma 14.6.

The rest of this section is devoted to a remarkable propriety of exact potential games, i.e., they can be "decomposed" with respect to a "basis" of even simpler games.

DEFINITION 14.9.– *A game* $G = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ *is:*

- a coordination game *if there is a function* $U : \mathbf{X} \longrightarrow \mathbb{R}$ *such that* $H_i \equiv U$*, for all* $i \in I$*;*

- a fictitious game *if, for all* $i \in I$ *and every* $x_{-i} \in X_{-i}$*, there is a* $k \in \mathbb{R}$ *such that* $H_i(x_i, x_{-i}) = k$ *for all* $x_i \in X_i$*.*

In a coordination game, the players seek the same aim and this is modeled by the fact that all the payoff functions are identical. In a fictitious game, the payoff of a player does not depend on its own strategy.

THEOREM 14.7.– *A game* $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ *is an exact potential game if and only if there exist two families of functions* $\{C_i\}_{i\in I}$ *and* $\{D_i\}_{i\in I}$ *such that:*

- $H_i = C_i + D_i$*, for all* $i \in I$*;*

- $\langle I, \{X_i\}_{i\in I}, \{C_i\}_{i\in I}\rangle$ *is a coordination game;*

- $\langle I, \{X_i\}_{i\in I}, \{D_i\}_{i\in I}\rangle$ *is a fictitious game.*

Proof. The direct part is trivial: the payoff function of the coordination game is an exact potential function for Γ.

In order to show the reverse, we define P as an exact potential for Γ. Then, for all $i \in I$, $H_i = P + (H_i - P)$. Clearly, $\langle I, \{X_i\}_{i\in I}, \{P_i \equiv P\}_{i\in I}\rangle$ is a coordination game. Let $i \in I$, $x_{-i} \in X_{-i}$ and $x_i, \chi_i \in X_i$. Then:

$$H_i\left(x_i, X_{-i}\right) - H_i\left(\chi_i, X_{-i}\right) = P\left(x_i, X_{-i}\right) - P\left(\chi_i, X_{-i}\right)$$

implies that:

$$H_i\left(x_i, X_{-i}\right) - P\left(x_i, X_{-i}\right) = H_i\left(\chi_i, X_{-i}\right) - P\left(\chi_i, X_{-i}\right)$$

whence it is directly deduced that $\langle I, \{X_i\}_{i\in I}, \{H_i - P\}_{i\in I}\rangle$ is indeed a fictitious game. ∎

The ordinal potential games may admit multiple ordinal potentials. In the exact potential games, all the possible potentials are related according to proposition 14.3.

PROPOSITION 14.3.– *Let* P_1 *and* P_2 *be two potentials for the game* Γ*. Then there is a constant* c *such that* $P_1(\mathbf{x}) - P_2(\mathbf{x}) = c$ *for all* $\mathbf{x} \in \mathbf{X}$*.*

Proposition 14.3 implies that the set of strategies maximizing a potential function of an exact potential game does not depend on any particular potential function.

Maximizing potential strategies has been used for proving the existence of pure Nash equilibria in the ordinal potential games.

The "potential maximizer" defined formally for a potential game Γ as:

$$\mathrm{PM}(\Gamma) = \left\{ \mathbf{x} \in \mathbf{X} : \mathbf{x} \in \underset{\chi \in \mathbf{X}}{\operatorname{argmax}} \{P(\chi)\} \text{ for a potential } P \text{ of } \Gamma \right\}$$

can then be used as an "equilibria refinement tool" as suggested in [MON 96].

14.6. Congestion games

14.6.1. *Rosenthal's model*

In a congestion model, players use multiple resources selected from a common "pool". The costs or benefits that a player gets by using a resource are determined by the number of players using the same resource. This section aims at defining Rosenthal's congestion model.

In this model, every player selects a subset of resources. The allocation function associated with every resource is a function depending only on the number of players using the resource. In a situation, a player's payoff is the sum of the allocation values associated with the resources he selects.

The existence of a pure Nash equilibrium can be established for this kind of game, by constructing an exact potential function (see below for a formal definition of games derived from Rosenthal's model).

In [MON 96] it is shown that every exact potential game is isomorphic to a congestion game. Their proof is somewhat complicated, but in [VOO 99] a shorter and more intuitive proof is given, based on Theorem 14.7.

A *congestion model* is defined by:

- a finite set of players I;

- a finite set of resources F;

- for every player $i \in I$, its strategies X_i is a set of subsets of F;

- for every resource $f \in F$, $w_f : \{1, \ldots, n\} \to \mathbb{R}$ is the allocation function of the resources, with $w_f(r), r \in \{1, \ldots, n\}$ as the allocation function value for each user of resource f and the total number of its users being r.

This model makes it possible to define a *congestion game* where I is the player set, X_i the set of choices of player $i \in I$ and $i \in I$ and $H_i : \mathbf{X} \to \mathbb{R}$ is defined as

follows; for all $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{X}$ and every $f \in F$:

$$n_f(\mathbf{x}) = |\{i \in I : f \in x_i\}|$$

is the number of users of resource f if the players choose $\mathbf{X}$. Then, $H_i(\mathbf{x}) = \sum_{f \in x_i} w_f(n_f(\mathbf{x}))$.

This definition implies that every player's payoff depends only on the resources he uses and the number of the players using the resources. Notice that the allocation functions can yield negative values, thus representing the cost of using a resource.

The main result of [ROS 73], stated in terms of exact potential, is given in Proposition 14.4. The trivial proof is omitted.

PROPOSITION 14.4.– *Let Γ be a congestion game defined as described above. Then Γ is an exact potential game. A potential function is given by $P : \mathbf{X} \to \mathbb{R}$ defined for all $\mathbf{x} = (x_i)_{i \in I} \in \mathbf{X}$ as follows:*

$$P(\mathbf{x}) = \sum_{f \in \bigcup_{i \in I} X_i} \sum_{l=1}^{n_f(\mathbf{x})} w_f(l)$$

Given that $\mathbf{X}$ is a finite set, the game has a Nash equilibrium in pure strategies.

Let $\Gamma = \langle I, \{X_i\}_{i \in I}, \{H_i\}_{i \in I} \rangle$ and $\Lambda = \langle I, (\chi_i)_{i \in I}, (U_i)_{i \in I} \rangle$ be two games with the same player set. Games Γ and Λ are *isomorphic* if, for all $i \in I$, there is a bijection $\varphi_i : X_i \to \chi_i$ such that for all $(x_1, \ldots, x_n) \in \mathbf{X}$:

$$H_i(x_1, \ldots, x_n) = U_i(\varphi_1(x_1), \ldots, \varphi_n(x_n))$$

Clearly, any congestion game whose resources have non-zero allocations only in the case where all its players use them, is a coordination game. Furthermore, every coordination game can be formulated this way, as shown in the proof of Theorem 14.8.

THEOREM 14.8.– *Every coordination game is isomorphic to some congestion game.*

Proof. Let $\Gamma = \langle I, \{X_i\}_{i \in I}, \{H_i \equiv U\}_{i \in I} \rangle$ be a coordination game of n players (in which every player has the same payoff function U).

For each $\mathbf{x} \in \mathbf{X}$, there is a distinct resource $f(\mathbf{x})$. Let us define the following congestion model $\langle I, F, \{\Xi_i\}_{i \in I}, \{W_f\}_{f \in F} \rangle$:

- $F = \cup_{\mathbf{x} \in \mathbf{X}} \{f(\mathbf{x})\}$;

- for all $i \in I$, $\Xi_i = \{g_i(x_i) : x_i \in X_i\}$, where function $g_i(x_i)$ is defined by $g_i(x_i) = \cup_{x_{-i} \in X_{-i}} \{f(x_i, x_{-i})\}$;

- for all $f(\mathbf{x}) \in F$:

$$W_{f(\mathbf{x})}(r) = \begin{cases} U(\mathbf{x}) & \text{if } r = |I| \\ 0 & \text{otherwise} \end{cases}$$

- for all $(x_1, \ldots, x_{|I|}) = \mathbf{x} \in \mathbf{X} : \bigcap_{i \in I} g_i(x_i) = \{f(\mathbf{x})\}$.

Thus, the game corresponding to this congestion model is isomorphic to Γ (the isomorphism relates x_i to $g_i(x_i)$). ∎

Now, let us consider a congestion game wherein the allocations for some resource are non-zero only in the case where this resource is used by exactly one player.

If for each player, given the strategy choice of the others, one resource is chosen and kept without bothering for the player's own choice, then a fictitious game results.

THEOREM 14.9.– *Every fictitious game is isomorphic to some congestion game.*

Proof. Let $\Gamma = \langle I, \{X_i\}_{i \in I}, \{H_i\}_{i \in I} \rangle$ be a fictitious game. For each $i \in I$ and each $x_{-i} \in X_{-i}$, we have a distinct resource $f(x_{-i})$. We define a congestion model $\langle I, M, \{\Xi_i\}_{i \in I}, \{W_f\}_{j \in F} \rangle$ with $F = \cup_{i \in I} \cup_{x_{-i} \in X_{-i}} \{f(x_{-i})\}$ and for all $i \in I : \Xi_i = \{h_i(x_i) : x_i \in X_i\}$: where:

$$\begin{aligned} h_i(x_i) &= \{f(X_{-i}) : x_{-i} \in X_{-i}\} \\ &\quad \cup \{f(\chi_{-j}) : j \in I \setminus \{i\} \text{ and } \chi_{-j} \in X_{-j} \text{ is such that } \chi_i \neq x_i\} \end{aligned}$$

and for all $f(x_{-i}) \in M$:

$$W_{f(x_{-i})}(r) = \begin{cases} H_i(x_i, x_{-i}) & \text{if } r = 1 \text{ (with arbitrary } x_i \in X_i) \\ 0 & \text{otherwise} \end{cases}$$

For all $i \in I, \bar{x}_{-i} \in X_{-i}$ and $\bar{x}_i \in X_i$: i is the unique user of $f(\bar{x}_{-i})$ in $(h_j(\bar{x}_j))_{j \in I}$ and all the remaining resources in $h_i(\bar{x}_i)$ have more than one user.

Indeed, let $i \in I$, $\bar{x}_{-i} \in X_{-i}$ and $\bar{x}_i \in X_i$. Then $f(\bar{x}_{-i}) \in h_i(\bar{x}_i)$ and, for all $j \in I \setminus \{i\} : f(\bar{x}_{-i}) \notin h_j(\bar{x}_j)$; thus i is the only user of $f(\bar{x}_{-i})$ in $(h_j(\bar{x}_j))_{j \in I}$. Let $f \in h_i(\bar{x}_i), f \neq f(\bar{x}_{-i})$. Then:

- if $f = f(\chi_{-i})$ for $\chi_{-i} \in X_{-i}$, then $\chi_{-i} \neq \bar{x}_{-i}$ implies that $\chi_j \neq \bar{x}_j$ for $j \in I \setminus \{i\}$ and hence $f = f(\chi_{-i}) \in h_j(\bar{x}_j)$;

- if $f = f(\chi_{-j})$ for $j \in I \setminus \{i\}$ and $\chi_{-j} \in X_{-j}$ with $\chi_i \neq \bar{x}_i$, then $f = f(\chi_{-j}) \in h_j(\bar{x}_j)$.

In both cases, f has more than one user. Thus, the game that corresponds to this congestion model is isomorphic to Γ (the isomorphism associates x_i with $h_i(x_i)$). ∎

In the two preceding theorems, it is shown that fictitious games and coordination games are congestion games. By using the decomposition of Theorem 14.7, Theorem 14.10 yields.

THEOREM 14.10.– *Every exact potential game is isomorphic to a congestion game.*

Proof. Let $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ be an exact potential game. Γ can be decomposed into a coordination game and a fictitious game as shown in Theorem 14.7 and their corresponding isomorphic congestion games can be obtained as shown in Theorems 14.8 and 14.9, respectively. Without loss of generality, we can take their resource sets to be disjoint. Then a congestion game isomorphic to Γ can be constructed by taking the union of the resource sets of the previous games, the allocation functions as in Theorems 14.8 and 14.9 and the strategies $\Xi_i = \{g_i(x_i) \cup h_i(x_i) : x_i \in X_i\}$. ∎

In a *network congestion game*, the strategies X_i are implicitly identified to paths in a network. Consider a network $G(v, E)$[13] two nodes $a_i, b_i \in V$ for each player $i \in I$ and an allocation function, with the edges in the role of resources. The set of subsets of F – the strategies for i –, is the set of edges of all paths from a_i to b_i.

A network congestion game is *symmetric* if all the players have the same starting and terminal points a and b. Usually the allocation functions on such congestion games are called *delay functions*. In what follows, we mention the complexity results for network congestion games obtained in [FAB 04].

14.6.2. *Complexity of congestion games (Rosenthal's model)*

The following result characterizes the complexity of congestion games of Rosenthal's model.

THEOREM 14.11.– *There is a polynomial algorithm finding a pure Nash equilibrium in symmetric network congestion games.*

Proof. The algorithm calculates an optimum of the function $P(\mathbf{x})$ given by Theorem 14.4.

Provided that the optimum is also a local optimum, the situation $\mathbf{x}'$ attained is a pure Nash equilibrium. The algorithm is in fact a reduction to the problem of minimum-cost flow.

13. This network is in fact a graph; the term "network" is used to avoid confusion with the games defined on graphs that have been presented above.

Given a network $N = \langle V, E, a, b\rangle$ and a delay function $(w_e)_{e\in E}$, every edge e in N is replaced by $n = |I|$ parallel edges between the same nodes, each one of capacity equal to 1 and of cost $w_e(1), \ldots, w_e(n)$, respectively.

It is easy to verify that every (integer) min-cost flow of the new network corresponds to a situation minimizing $P(\mathbf{x})$. ∎

Theorem 14.12 is about the complexity of searching a pure Nash equilibrium in congestion games.

THEOREM 14.12.– *Finding a pure Nash equilibrium is* **PLS***-complete for congestion games of the following types:*

1) general congestion games;

2) symmetric congestion games;

3) asymmetric network congestion games.

The complete and rather technical proof of Theorem 14.12 can be found in [FAB 04]. We give here an outline:

- to show 1), the authors of [FAB 04] build from an instance of POSNAE3FLIP[14] a congestion game such that its Nash equilibria are the local optima of the instance;

- the proof 2) is a reduction from the non-symmetric to the symmetric case;

- finally, to show 3), which is more complicated, the authors modify the problem POSNAE3FLIP and get a new one that they call WITNESSED XPNAE3FLIP; then they show that:

- there is a **PLS**-reduction from WITNESSED XPNAE3FLIP towards NETWORK CONGESTION GAME;

- WITNESSED XPNAE3FLIP is **PLS**-complete.

14.6.3. *Other models*

The games introduced in [KON 97, MIL 04, QUI 94] are similar to those seen previously, in the sense that the players' payoffs are characterized by a congestion effect. The diverse game classes we deal with in this section are identified by properties that

14. POSNAE3FLIP denotes the following problem: "given an instance of NOT-ALL-EQUAL-3SAT (i.e., a CNF formula for which we search for an 0-1 assignment to its variables satisfying it and such that every clause in the formula has at least one literal set to 1 and at least a literal set to 0) with weights over the clauses and containing only positive literals, find an assignment with satisfied clauses whose total weight cannot increase by flipping one variable"; it is known [SHÄ 91] that this problem is **PLS**-complete.

have to do with "strategies interaction". In particular, the authors of [KON 97] make the following assumptions, for a game $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$.

HYPOTHESIS 14.1.– There is a finite set F such that $X_i = F$ for every player $i \in I$.

This set F is called "resource set" and a choice for a player i is to select an element of F.

HYPOTHESIS 14.2.– For all situations $\mathbf{x} \in \mathbf{X}$ and all couples of players $i, j \in I$: if $x_i \neq x_j$ and $x'_j \in X_j$ is such that $x_i \neq x'_j$, then $H_i(x_j, x_{-j}) = H_i(x'_j, x_{-j})$.

The authors of [KON 97] call this hypothesis *independence of irrelevant choices*: for every player $i \in I$ and every situation, the payoff of i shall not change if the set of players choosing the same resource as i is not modified.

Let $\mathbf{x} \in \mathbf{X}, f \in F$. We note by $n_f(X)$ the number of players having selected resource f in the situation $\mathbf{x}$. The third assumption is then formulated as follows.

HYPOTHESIS 14.3.– For all players $i \in I$ and all situations $\mathbf{x}$, $\chi \in X$ with $x_i = \chi_i$: if $n_f(\mathbf{x}) = n_f(\chi)$ for all $f \in F$, then $H_i(\mathbf{x}) = H_i(\chi)$.

Hypothesis 14.3 can be seen as a kind of "anonymity condition". It reflects the idea that the payoff for player i depends rather on the number of the players that have selected the same resource as him than on their identity.

A fourth hypothesis, called *partial rivalry*, establishes that any player i does not regret that other players have selected the same resource with him, by selecting a different resource.

HYPOTHESIS 14.4.– For all players $i \in I$, all situations $\mathbf{x} \in \mathbf{X}$, all players $j \neq i$ such that $x_j = x_i$ and all $x'_j \neq x_i$: $H_i(x_j, x_{-j}) \leqslant H_i(x'_j, x_{-j})$.

Although in [MIL 04] this model is introduced in a slightly different manner, the class of games obtained is the same. More precisely, [MIL 04] makes Hypotheses 14.1, 14.4 and also 14.5.

HYPOTHESIS 14.5.– For all players $i \in I$ and all couple of situations $(\mathbf{x}, \chi)$ with $x_i = \chi_i = f$, if $n_f(\mathbf{x}) = n_f(\chi)$, then $H_i(\mathbf{x}) = H_i(\chi)$.

In other words, the payoff for player i depends only on the number of players having selected the same resource as him. It can be shown that making Hypothesis 14.1 implies directly Hypotheses 14.2 and 14.3. The reverse also holds.

LEMMA 14.3.– *Every game* $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ *satisfying Hypotheses 14.1, 14.2 and 14.3 also satisfies Hypothesis 14.5.*

Proof. Let $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ be a game satisfying Hypotheses 14.1, 14.2 and 14.3. Let $i \in I$, $\mathbf{x}, \chi \in \mathbf{X}$ such that $x_i = \chi_i = f$; suppose also that $n_f(\mathbf{x}) = n_f(\chi)$. Clearly, if $|F| = 1$, then Hypothesis 14.5 holds. Otherwise, by using repeatedly Hypothesis 14.2, we can assert that for a fixed γ such that $\gamma \neq x_i$, relations $H_i(x_i, x_{-i}) = H_i(x_i, x'_{-i})$ hold, where, for every $j \in I \setminus \{i\}$:

$$x'_j = \begin{cases} x_i & \text{if } x_j = x_i \\ \gamma & \text{otherwise} \end{cases}$$

and relations $H_i(x_i, \chi_{-i}) = H_i(x_i, \chi'_{-i})$ also hold, where, $j \in I \setminus \{i\}$:

$$\chi'_j = \begin{cases} x_i & \text{if } \chi_j = x_i \\ \gamma & \text{otherwise} \end{cases}$$

Notice that, for all $h \in F$, $n_h(x_i, x'_{-i}) = n_h(x_i, \chi'_{-i})$. Thus Hypothesis 14.3 implies that: $H_i(x_i, x'_{-i}) = H_i(x_i, \chi'_{-i})$. Consequently, $H_i(x_i, x_{-i}) = H_i(x_i, x'_{-i}) = H_i(x_i, \chi'_{-i}) = H_i(\chi_i, \chi_{-i})$. ∎

The authors of [KON 97, MIL 04] have shown independently the following theorem.

THEOREM 14.13.– *Every game $\Gamma = \langle I, \{X_i\}_{i\in I}, \{H_i\}_{i\in I}\rangle$ statisfying Hypotheses 14.1, 14.2, 14.3 and 14.4 has a Nash equilibrium.*

According to the preceding definitions, a situation $\mathbf{x}$ of Γ is an equilibrium in the strong sense (or a strong equilibrium) if, for all $K \subseteq I$ and all choices $\chi_K \in X_K$, there is at least one player $i \in K$ such that $H_i(\chi_K, X_{-K}) \leqslant H_i(X)$. We note by $\mathrm{SE}(\Gamma)$ the set of equilibria in the strong sense of game Γ. Generally, the existence of a strong equilibrium is not guaranteed; however, the authors of [KON 97] show Theorem 14.14.

THEOREM 14.14.– *Every game satisfying Hypotheses 14.1, 14.2, 14.3 and 14.4 has non-empty strong equilibria set.*

The authors of [QUI 94] propose a model in which the assumption that all players have the same resource set (formulated by Hypothesis 14.1) is replaced by the following hypothesis.

HYPOTHESIS 14.6.– There is a finite set F such that for all $i \in I,\ X_i \subseteq F$.

Assuming Hypothesis 14.6, it is simple to verify that Hypothesis 14.5 implies Hypotheses 14.2 and 14.3. However, the reverse does not hold, unlike the case in Lemma 14.3.

EXAMPLE 14.2.– Let $I = \{1, 2, 3\}$, $F = \{a, b, c\}$ and the strategy sets $X_1 = \{a, b\}, X_2 = \{a\}, X_3 = \{a, c\}$. This game satisfies Hypothesis 14.6. Hypothesis 14.3 does not impose any more constraints and to meet Hypothesis 14.2 $H_1(b, a, a) = H_1(b, a, c)$ and $H_3(a, a, c) = H_3(b, a, c)$ must hold. But this does not imply that $H_2(a, a, c) = H_2(b, a, a)$, which is required by Hypothesis 14.5.

THEOREM 14.15.– *[QUI 94] Every game satisfying Hypotheses 14.6, 14.5 and 14.4 has a pure Nash equilibrium.*

The games that belong to the above-defined classes may not admit a potential function. Let us consider a "cross symmetry" condition, establishing the fact that the payoffs over some resources are independent of players' identities, provided that their number does not change. This condition is formulated as follows.

HYPOTHESIS 14.7.– For all couples of situations $(\mathbf{x}, \chi) \in \mathbf{X}^2$ and all couples of players $(i, j) \in I^2$: if $x_i = \chi_j = f$ and $n_f(\mathbf{x}) = n_f(\chi)$, then $H_i(\mathbf{x}) = H_j(\chi)$.

Notice that Hypotheses 14.1 and 14.7 imply Hypothesis 14.5 and thus also Hypotheses 14.2 and 14.3. Moreover, Hypotheses 14.1 and 14.7 do guarantee the existence of an exact potential.

THEOREM 14.16.– *Every game satisfying Hypotheses 14.1 and 14.7 is an exact potential game.*

Proof. Let $\Gamma = \langle I, \{X_i\}_{i \in I}, \{H_i\}_{i \in I} \rangle$ be a game satisfying Hypotheses 14.1 and 14.7. For all $f \in F$ and $\mathbf{x}, \chi \in \mathbf{X}$ such that $n_f(\mathbf{x}) = n_f(\chi)$, according to Hypothesis 14.7 if there exist $i, j \in I$ such that $x_i = \chi_j = f$, then $H_i(\mathbf{x}) = H_j(\chi)$. This shows that for all $f \in F$, there is an allocation function $W_f : \{1, \ldots, n\} \to \mathbb{R}$ such that, for all $\mathbf{x} \in \mathbf{X}$, if $x_i = f$, then $H_i(\mathbf{x}) = W_f(n_f(\mathbf{x}))$. Thus, Γ becomes a congestion game like those defined in section 14.6.1[15]. Hence, the result is directly derived from Proposition 14.4. ∎

Theorem 14.16 also holds if Hypothesis 14.1 is replaced by Hypothesis 14.6. Proposition 14.4 means also that the allocation function $(W_f)_{f \in F}$ induces the potential: $P : \mathbf{X} \longrightarrow \sum_{f \in \cup_{i \in I} \{X_i\}} \sum_{l=1}^{n_f(\mathbf{x})} W_f(l)$.

The authors of [VOO 99] define a class $\mathcal{C}$ of congestion games satisfying Hypotheses 14.1, 14.7 but also 14.4, i.e.:

$$\mathcal{C} = \left\{ \Gamma = \left\langle I, \{X_i\}_{i \in I}, \{H_i\}_{i \in I} \right\rangle : \Gamma \text{ satisfies Hypotheses 14.1, 14.4 and 14.7} \right\}$$

15. Where the choice of a resource $f \in F$ is associated with the choice of a set of resources $\{f\} \subseteq F$.

THEOREM 14.17.– *For every game that belongs to class $\mathcal{C}$, SE = NE = PM*[16].

Indeed, for every game Γ, $\mathrm{SE}(\Gamma) \subseteq \mathrm{NE}(\Gamma)$ and for every exact potential game, $\mathrm{PM}(\Gamma) \subseteq \mathrm{NE}(\Gamma)$. In order to show Theorem 14.17, it suffices to show the following result.

PROPOSITION 14.5.– *For every game $\Gamma \in \mathcal{C}$:*

- *$NE(\Gamma) \subseteq PM(\Gamma)$;*
- *$NE(\Gamma) \subseteq SE(\Gamma)$.*

Finally, we note that the largest family of games, for which the existence of pure Nash equilibria has been established, is the family of congestion games. Moreover, the games of the model presented in [KON 97, MIL 04] have at least one pure Nash equilibrium; despite this, we know of no potential function making it possible to establish this result.

14.7. Final note

The concept of non-cooperative games as formulated by Definition 14.1 is general enough to represent a really vast family of games which, at first sight, do not enter in this formulation. Indeed, several game models, like repeated games, multiple step games, games played by automata or games with incomplete information, can be reduced to this form with suitable definitions of player set, strategies and preferences.

Non-cooperative game theory has been developed in most cases through the study of games of special forms; for instance, there are numerous works on two-player games or on repeated games, to name a few such forms. Among them, only families[17] for which advances in algorithmic questions are most significant have been discussed here.

Finally, we have not dealt with "inverse" problems, namely mechanism design problems and auctions; this category constitutes by itself too broad a subject, with its own algorithmic issues, that go beyond the scope of this chapter.

14.8. Bibliography

[ANG 05] ANGEL E., CHRISTOPOULOS P., ZISSIMOPOULOS V., "Recherche locale : complexité et approximation", PASCHOS V. T., Ed., *Optimisation combinatoire : concepts avancés*, vol. 2, Chapter 4, p. 139-177, Hermès, Paris, 2005.

16. SE=*Strong Equilibria set*, NE=*Nash equilibria set*, PM=*Potential Maximizer set*.

17. Games defined on a graph, potential games and congestion games.

[AUM 85] AUMAN R. J., MASCHLER M., "Game-theoretic analysis of a bankuptcy problem from the Talmud", *Journal of Economic Theory*, vol. 36, num. 195-213, 1985.

[BOR 21] BOREL E., "La théorie du game et les équations intégrales à noyau symétrique", *C.R. Acad. Sci. Paris*, vol. 173, p. 1304-1308, 1921.

[BOR 24] BOREL E., "Sur les games de hasard où interviennent l'hasard et l'habilité des joueurs", BOREL E., Ed., *Théorie des probabilités*, p. 204-224, J. Hermann, Paris, France, 1924.

[BOR 27] BOREL E., "Sur les systèmes de formes linéaires à déterminant symétrique gauche et la théorie générale du game", *C.R. Acad. Sci. Paris*, vol. 184, p. 52-53, 1927.

[BOR 85] BORDER K. C., *Fixed Point Theorems with Applications to Economics and Game Theory*, Cambridge University Press, Cambridge, 1985.

[BOU 02] BOUTON C. L., "Nim, a game with a complete mathematical theory", *Annals of Mathematics (Princeton)*, vol. 2, num. 3, p. 35-39, 1902.

[BUB 79] BUBYALIS V. S., "On equilibria in finite games", *Int. J. of Game theory*, vol. 8, num. 2, p. 65-79, 1979.

[CHE 05] CHEN X., DENG X., "3-NASH is PPAD-complete", *Electronic Colloquium in Computational Complexity*, p. TR05-134, 2005.

[CHE 05B] CHEN X., DENG X., "Settling the Complexity of 2-Player Nash-Equilibrium", *Electronic Colloquium in Computational Complexity*, p. TR05-140, 2005.

[CON 03] CONITZER V., SANDHOLM T., "Complexity results about Nash equilibria", *Proc. Int. Joint Conf. of Artificial Intelligence, IJCAI'03*, p. 765-771, 2003.

[COU 1838] COURNOT A., *Recherches sur les principes mathématiques de la théorie des richesses*, Paris, 1838.

[DAS 05] DASKALAKIS K., PAPADIMITRIOU C. H., "Three-player games are hard", *Electronic Colloquium in Computational Complexity*, p. TR-05-139, 2005.

[ELK 06] ELKIND E., GOLDBERG L. A., GOLDBERG P., "Nash equilibria in graphical games on trees revisited", *Proc. ACM conf. on electronic commerce*, p. 100-109, 2006.

[FAB 04] FABRIKANT A., PAPADIMITRIOU C. H., TALWAR K., "The complexity of pure Nash equilibria", *Proc. STOC'04*, p. 604-612, 2004.

[FRÉ 53] FRÉCHET M., "Emile Borel, initiator of the theory of psychological games and its application", *Econometrica*, vol. 21, num. 1, p. 95-96, 1953.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and Intractability. A Guide to the Theory of NP-completeness*, W. H. Freeman, San Francisco, 1979.

[GIL 89] GILBOA I., ZEMEL E., "Nash and correlated equilibria: some complexity considerations", *Games and Economic Behavior*, vol. 1, p. 80-93, 1989.

[GOL 05] GOLDBERG P. W., PAPADIMITRIOU C. H., "Reducibility among equilibrium problems", *Electronic Colloquium in Computational Complexity*, p. TR-05-090, 2005.

[GRI 88] GRIGORIEV D. Y., VOROBIEV N. N., "Solving systems of polynomial inequalities in subexponential time", *J. of Symb. Computing*, vol. 5, p. 37-64, 1988.

[KEA 01] KEARNS M., LITTMAN M., SINGH S., "Graphical models for game theory", *Proc. Conf. on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, p. 253-260, 2001.

[KOL 92] KOLLER D., MEGIDDO N., "The complexity of two-person zero-sum games in extensive form", *Games and Economic Behavior*, vol. 4, p. 528-552, 1992.

[KON 97] KONISHI H., BRETON M. L., WEBER S., "Equilibria in a model with partial rivalry", *Journal of Economic Theory*, vol. 72, p. 225-237, 1997.

[KUH 58] KUHN H. W., TUCKER A. W., "John von Neumann's work in the theory of games and mathematical economics", *Bull. Amer. Math. Soc.*, vol. 64, num. 3, p. 100-122, 1958.

[LAS 19] LASKER E., *Die philosophie des unvollendbaren*, Vereinigung wissenschaftlicher Verleger, Berlin, 1919.

[LEM 64] LEMKE C. E., HOWSON J. T., "Equilibrium points of bimatrix games", *J. of the SIAM*, vol. 12, p. 413-423, 1964.

[LEO 94] LEONARD R. J., "Reading Cournot, reading Nash: the creation and stabilisation of Nash equilibrium", *The Economic Journal*, vol. 104, num. 424, p. 492-511, 1994.

[LEW 81] LEWIS H. R., PAPADIMITRIOU C. H., *Elements of the Theory of Computation*, Prentice-Hall, Englewood Cliffs, 1981.

[LUC 57] LUCE R. D., RAIFFA H., *Games and Decisions. Introduction and Critical Survey*, John Wiley and Sons, New York, 1957.

[MCL 97] MCLENNAN A., PARK I.-U., "Generic 4x4 two person games have at most 15 Nash equilibria", Available at `http://www.citeseer.ist.psu.edu/mclennan97generic.html`, 1997.

[MCL 02] MCLENNAN A., "On the expected number of Nash equilibria of a normal form game", Available at `http://www.citeseer.ist.psu.edu/mclennan02expected.html`, 2002.

[MIL 04] MILGROM P., *Putting Auction Theory to Work*, Cambridge University Press, Cambridge, 2004.

[MON 96] MONDERER D., SHAPLEY L. S., "Potential games", *Games and Economic Behavior*, vol. 14, p. 124-143, 1996.

[MON 03] MONNOT J., PASCHOS V. T., TOULOUSE S., *Approximation polynomiale des problèmes NP-difficiles : optima locaux et rapport différentiel*, Hermès, Paris, 2003.

[MOO 09] MOORE E. H., "A generalization of the game called Nim", *Annals of Mathematics (Princeton)*, vol. 2, num. 11, p. 93-94, 1909.

[NAS 50a] NASH J. F., "The bargaining problem", *Econometrica*, vol. 18, p. 155-162, 1950.

[NAS 50b] NASH J. F., "Equilibrium points in N-person games", *Proc. Nat. Acad. Sci. USA*, vol. 36, p. 48-49, 1950.

[NAS 51] NASH J. F., "Non-cooperative games", *Annals of Mathematics (Princeton)*, vol. 54, num. 2, p. 286-295, 1951.

[NEU 28] VON NEUMANN J., "Zur theorie der gesellschaftsspiele", *Mathematische Annalen*, vol. 100, p. 295-320, 1928.

[NEU 53a] VON NEUMANN J., "Communication on the Borel notes", *Econometrica*, vol. 21, num. 1, p. 124-125, 1953.

[NEU 53b] VON NEUMANN J., MORGENSTERN O., *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, 1953, 3rd edition revised and completed with the axiomatic utility theory (1st edition appeared in 1944).

[PAP 94a] PAPADIMITRIOU C. H., *Computational complexity*, Addison-Wesley, Reading, 1994.

[PAP 94b] PAPADIMITRIOU C. H., "On the complexity of the parity argument and other inefficient proofs of existence", *J. Comput. System Sci.*, vol. 48, num. 3, p. 498-532, 1994.

[PAP 05] PAPADIMITRIOU C. H., ROUGHGARDEN T., "Computing equilibria in multi-player games", *Proc. Symposium on Discrete Algorithms, SODA'05*, p. 82-91, 2005.

[PAS 04] PASCHOS V. T., *Complexité et approximation polynomiale*, Hermès, Paris, 2004.

[PAS 06] PASCHOS V. T., "Concepts de base de l'algorithmique et de la théorie de la complexité", PASCHOS V. T., Ed., *Optimisation combinatoire : concepts fondamentaux*, vol. 1, Chapter 1, p. 23-41, Hermès, Paris, 2006.

[POT 05] POTTIÉ O., Etude des équilibres de Nash, jeux de potentiel et jeux de congestion, Annales num. 4-5, p. 124-143, LAMSADE, Paris-Dauphine University, 2005.

[QUI 94] QUINT T., SHUBIK M., A model of migration, Working paper, Cowles Foundation, Yale University, 1994.

[RAP 65] RAPAPORT A., CHAMMAH A. M., *Prisoner's Dilemma. A Study in Conflict and Cooperation*, University of Michigan Press, Ann Arbor, 1965.

[ROS 73] ROSENTHAL R. W., "A class of games possessing pure-strategy Nash equilibria", *Int. J. of Game theory*, vol. 2, p. 65-67, 1973.

[SAV 04] SAVANI R., VON STENGEL B., "Exponentially many steps for finding a Nash equilibrium in a bimatrix game", *Proc. FOCS'04*, p. 258-267, 2004.

[SHÄ 91] SHÄFFER A. A., YANNAKAKIS M., "Simple local search problems that are hard to solve", *SIAM J. Comput.*, vol. 20, num. 1, p. 56-87, 1991.

[SIN 00] SINGH S., KEARNS M., MANSOUR Y., "Nash convergence of gradient dynamics in general-sum games", *Proc. Conf. on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, p. 541-548, 2000.

[STE 60] STEINHAUS H., "Definitions for a theory of games and pursuit", *Naval Res. Logist. Quart.*, vol. 7, num. 2, p. 105-107, 1960, Translation of the chapter which appeared in Polish at 1925 in Mysl Akademicka, Lwow, 1 (1), 13-14.

[VIC 02] VICKREY D., KOLLER D., "Multi-agent algorithms for solving graphical games", *Proc. American Nat. Conference on Artificial Intelligence*, p. 345-351, 2002.

[VOO 99] VOORNEVELD M., BORM P., MEGEN F. V., TIJS S., "Congestion games and potentials reconsidered", *Int. Game Theory Review*, vol. 1, num. 3-4, p. 283-299, 1999.

[VOR 98] VOROBIEV N. N., *Mathematical Foundations of Game Theory*, Birkhäuser, Boston, 1998.

[ZER 13] ZERMELO E., "Uber eine anwendung der mengenlehre auf die theorie des schachspiels", *Proc. International Congress of Mathematicians, Cambridge*, p. 501-504, 1913.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 15

Flows!

15.1. Introduction

Some problems are simplified by a "divide and conquer" approach, which usually leads to efficient algorithms (even often optimal). For instance, to sort an array, we may split it into two parts, sort the left part, then the right part and merge the two sorted parts. Some other problems do not seem to naturally allow such an approach: for instance, when searching for the shortest path between two vertices of a graph, it is not clear at all that it may be obtained in such a way. Neither the image pattern recognition nor problems like automatic translation, database management or flow problems seem to allow a natural "divide and conquer" approach.

Actually these problems can be solved thanks to a "divide and conquer" approach. The goal of this chapter is to fully detail one of them, namely the flowing problem.

In order to solve a problem thanks to a "divide and conquer" algorithm, the data is usually split into smaller parts (usually halved) whose solutions are merged together.

Let us return to the example of sorting an array. It is possible to use a quicksort, a mergesort, a heapsort, etc. The heapsort, which is not a "divide and conquer" algorithm, is optimal in the average and worst case but is averagely less efficient than the quicksort. The quicksort proceeds by choosing an element of the array (for example, the middle one called "pivot"), puts at its right the elements that are greater than or equal to the pivot, at its left the elements that are lower than or equal to the pivot, and sorts the left and right parts. The step of dealing with the two parts is performed

Chapter written by Michel KOSKAS and Cécile MURAT.

at first and then the independent calculations are performed. The mergesort consists of sorting the left and right halves of the array and then deals with the two parts by merging two sorted subarrays. A dichotomic search has at each step only to deal with one half of the array and therefore does not necessitate a "deal with two parts" step. The algorithm presented in this paper uses an algorithm to find the shortest paths in an unweighed graph that performs several times the "deal with two parts" step and the "merge" (or "check solutions") part.

We use an extra data structure where the method applies. This means that the first step of these algorithms consists of building the extra data structure and, fortunately, this step is performed only once and may be considered as part of the data representation. For instance, when dealing with graphs for the shortest path problem, for example, the extra data structure is calculated only once and is used for any couple of vertices.

All these algorithms lie on the use of radix trees, which are an efficient structure for storing data and for performing hierarchical calculations on it.

The chapter is organized as follows. After giving the basic definitions and notation, we present the common knowledge about radix trees followed by some applications in various domains. Section 15.4 is devoted to the study of the shortest path problem and its refinements, which in turn leads to an efficient improvement of the well known Ford-Fulkerson Algorithm for flows.

15.2. Definitions and notations

A graph G is a triple (V, A, C) where V is a finite set of *vertices*, A is a subset of $V \times V$ (the *arcs* of the graph) and C a function from A to $\mathbb{R}^+$ (the *capacities*).

The sets A and C may be considered as an adjacency matrix $M = (m_{i,j})_{1 \leqslant i,j \leqslant n}$ where $v = \#(V)$ and:

$$\forall i, j,\ m_{i,j} = \left\{ \begin{array}{ll} 0 & \text{if } (i,j) \notin A \\ C((i,j)) & \text{otherwise} \end{array} \right\}$$

For any vertices i and j, the arc (i, j) is an outgoing arc of i and an incoming arc of j. For any vertex i, we denote by $\mathrm{Out}(i)$ the set of its outgoing arcs and $\mathrm{Inc}(i)$ its incoming vertices.

The flow problem is the following. Among the vertices of G, we choose a vertex called Source (denoted by S from now on), and another one called Well (denoted by W from now on). The goal is to find a function φ over A such that $\forall a \in A\, \varphi(a) \leqslant C(a)$

and for all vertices (except S and W) of G, the Kirchhoff's laws apply, which means that:

$$\sum_{a \in Inc(v)} \varphi(a) = \sum_{a \in Out(v)} \varphi(a)$$

The value of the flow φ is:

$$\sum_{a \in Out(S)} \varphi(a) = \sum_{a \in Inc(W)} \varphi(a)$$

The problem is to find a flow maximizing this value.

The algorithm usually used to solve this problem is the well known Ford-Fulkerson algorithm, which is explained in section 15.5.

15.3. Presentation of radix trees

Radix trees are trees which make it possible to store data in a hierarchical way. Let us suppose for instance that we want to store a set of words over an alphabet $\{a, b, c\}$. Then the edges are labeled with the letters of the alphabet (or an empty letter) and the words are obtained by reading the path between the root of the tree and any of its leaves. For instance, the set $\{a, ab, aba, abc, bab, bac\}$ may be stored as (see Figure 15.1):

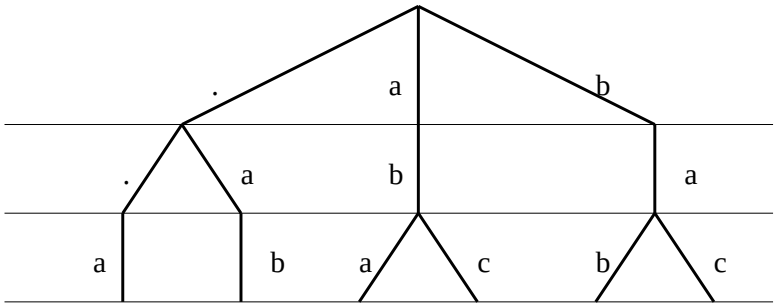


Figure 15.1. *The set $\{a, ab, aba, abc, bab, bac\}$ stored in a radix tree*

The efficiency of the radix tree structure is revealed by the calculations it allows. For instance, a set of integers may be stored as a set of words by writing them in a given basis (and adding 0 at the left of shorter integers). Then a set of couples of integers may also be stored in the same way. Let us take an example: the set $S = \{(0,1),(0,2),(1,3),(0,4),(3,1),(2,2)\}$ it may be written in base 2 as:

$$S = \{(000,001),(000,010),(001,011),(000,100),(011,001),(010,010)\}$$

and, using the alphabet $\{00, 01, 10, 11\}$ it may be rewritten as $S = \{00.00.01,$-$00.01.00, 00.01.11, 01.00.00, 00.10.11, 00.11.00\}$ (the i-th digit of an element is made up of the catenation of the i-th digits of the corresponding couple). This may finally be stored in the set (see Figure 15.2):

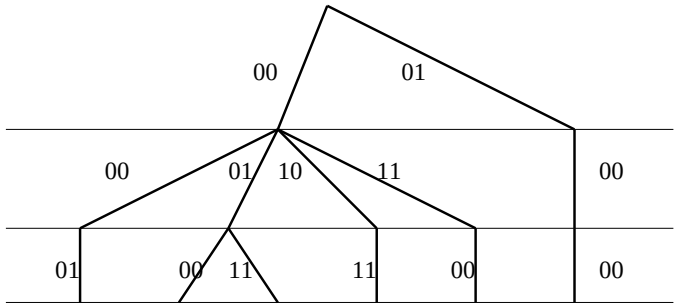


Figure 15.2. *The set $S = \{(0,1),(0,2),(1,3),(0,4),(3,1),(2,2)\}$ stored in a radix tree*

These sets allow efficient operations such as intersection, union, complementation (for set operations) and when the elements stored are integers or couples of integers we may also perform a translation of the whole set asymptotically faster than by adding the constant to all the elements of the set. Let us furthermore notice that the elements are stored sorted.

It is also possible to advantageously store a graph in a radix tree, by thickening the graph.

DEFINITION 15.1.– *Let $G = (V, A)$ be a graph where $V = \{v_0, \ldots, v_{v-1}\}$. Let $G' = (V', A')$ be an unweighed graph where $V' = \{v'_0, \ldots, v'_{v'-1}\}$ is a set of $v' = \lceil \frac{v}{2} \rceil$ vertices and such that:*

$$\forall i, j \leqslant v', (v'_i, v'_j) \in A' \Leftrightarrow \exists a \in \{2i, 2i+1\}, b \in \{2j, 2j+1\}, (v_a, v_b) \in A$$

Such a graph G' is said to be a thickening of G or, equivalently, G is a refinement of G'.

REMARK 15.1.– Any equivalence relation may be used to define the thickening of a graph (in the definition two vertices are in the same equivalence class if and only if they have the same quotient of 2).

In other words, the vertices of V are grouped by two and there exists an arc between any two couples if and only if there exists an arc between an element of the first couple and an element of the second couple. A graph may be recursively thickened as long as the obtained graph has a number of arcs a and of vertices v verifying

$P(\frac{a}{s^2}) \geqslant \frac{1}{2}$ where $P(X) = 1 - \frac{3}{2}X + X^2 - \frac{1}{4}X^3$. Finally, a graph may be stored in a radix tree. Let us consider for instance an unweighed graph by its adjacency matrix. The adjacency matrix is then "summarized" as in Figure 15.3.

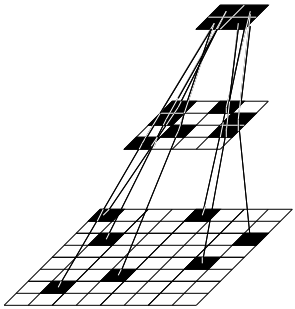


Figure 15.3. *A graph thickened*

In the next sections we shall explain how this data structure may be used to profile "divide and conquer" algorithms on a number of problems.

15.3.1. *Database management*

A relational database is made up of several relations linked by sets of primary keys and foreign keys. A primary key of a relation T is an attribute or a set of attributes such that T may not contain two lines whose values are equal on these attributes. Another way to say it is that the value of a primary key identifies a single line of T. Any relation has at least a primary key.

A foreign key of a relation T is an attribute or a set of attributes targeting a primary key of a relation T'. In other words, a foreign key is a recall of a primary key. A foreign key replaces all the attributes of T' for the line identified by its value. It is a cheap storage of all the values of T' (only the values of the primary key of T' are stored instead of all its attributes). For instance, if a relation contains the data of a set of clients, each command should not contain all the data of the client but only the value of the primary key of the relation storing the client data.

Given a relation T, we build the matching "expansion relation", the thesauruses and radix trees and answer SQL queries (see below). For a full description of this algorithm, see [KOS 04]

Expansion relations

We give a recursive definition of an expansion relation. It is a denormalized version of the database. The foreign keys are replaced by the attributes of the targeted relations.

DEFINITION 15.2.– *The expansion of a foreign key belonging to a relation is the replacement of the attributes of this foreign key by the attributes of the targeted relation. Let T be a relation. The related expansion relation $E(T)$ is the relation obtained by expanding recursively all its foreign keys.*

Building indexes

Once the expansion tables are built, the database is made up of relations which are no longer linked because the jointures have been formally expanded in each relation. Then we build the thesaurus of each attribute and store for each element of the thesaurus the set of the line indices it appears at. For example, let us consider the attribute in Table 15.1.

0	1	2	3	4	5	6	7	8	9	10
Male	Female	Female	Male	Female	Male	Male	Female	Female	Male	Male

Table 15.1. *An example of simple relation*

Then its thesaurus is $\{$Female, Male$\}$ and the indices where Female appears (resp., Male) are $\{1, 2, 4, 7, 8\}$ (resp. $\{0, 3, 5, 6, 9, 10\}$). The radix trees associated with the words of the thesaurus are shown in Figure 15.4).

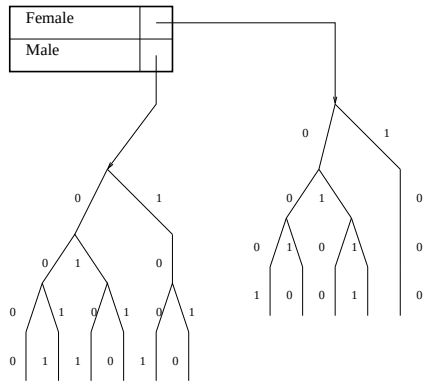


Figure 15.4. *The indices of an attribute*

Solving SQL requests

We briefly give indications on how to solve SQL requests. For more details see [KOS 04].

The main part of this work is to solve the `where` clause, which is composed of equalities or inequalities separated by logical operators (and, or) and joint clauses etc.

This `where` clause returns a radix tree, which contains the line indexes answering the clause. The joint clauses are irrelevant here because of the use of expansion relations.

An equality between an attribute and a constant is the simplest case because we simply have to read the corresponding radix tree. An inequality may be calculated by several readings of radix trees and the calculation of an "or". An "and" (resp., "or") clause is calculated by performing the "and" (resp., "or") of the corresponding radix trees.

All the other operations may be performed in a similar way.

15.3.2. *Image pattern recognition*

An image may be considered as a mapping from $[\![0, L-1]\!] \times [\![0, H-1]\!]$ to a finite set (of colors), with l being the width and L the height of the image. Each color appears at a set of coordinates that we may store in a radix tree. For instance, the picture (see Figure 15.5) contains three colors: black, grey and white. The black pixels are at indexes $B = \{(3,0),(4,0),(2,1),(3,1),(5,1),(6,1),(2,2),(6,2),$-$(0,3),(3,3),(4,3),(7,3),(0,4),(3,4),(4,4),(7,4),(2,5),(6,5),(2,6),(3,6),$-$(5,6),(6,6),(3,7),(4,7)\}$, the white ones at indexes $W = \{(1,0),(2,0),(5,0),$-$(6,0),(0,1),(7,1),(0,2),(3,2),(4,2),(7,2),(2,3),(5,3),(2,4),(5,4),(0,5),$-$(3,5),(4,5),(7,5),(0,6),(7,6),(1,7),(2,7),(5,7),(6,7)\}$ and the grey ones at indexes $G = \{(0,0),(7,0),(3,1),(4,1),(2,2),(5,2),(2,3),(6,3),(2,4),(6,4),$-$(2,5),(5,5),(3,6),(4,6),(0,7),(7,7)\}$.

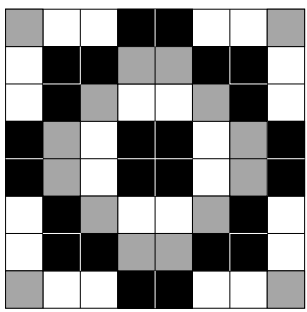


Figure 15.5. *The pixels of an image*

The set of appearance of the black pixels may be rewritten, in base 2, as $B = \{(011,000),(100,000),(010,001),(011,001),(101,001),(110,001),$-$(010,010),(110,010),(000,011),(011,011),(100,011),(111,011),(000,100),$-$(011,100),(100,100),(111,100),(010,101),(110,101),(010,110),(011,110),$-$(101,110),(110,110),(011,111),(100,111)\}$ and the three sets may be stored in radix trees as in Figure 15.6.

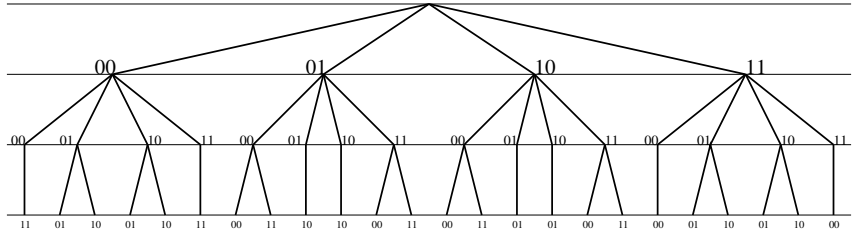


Figure 15.6. *The radix tree of the black pixels*

Such a radix tree may be translated by any vector (the calculation is easy and asymptotically faster than the translation of a whole set stored in an array, for instance). Then, by calculating $(B + (a, b)) \cap W$ we obtain the set of coordinates of the white pixels having a black pixel at relative coordinates $(-a, -b)$.

By iterating this process we find any pattern in an image (for more details on this algorithm see [KOS 01]).

15.3.3. *Automatic translation*

This section describes a work in progress. One of the problems arising to an automated translation process comes from the fact that a word has usually several senses and each of these senses has a different translation in the target language. The Prophyre tree is a tree in which all the meanings of all the words are put together (this tree is actually unachieved whatever the language) in a way which makes it possible to calculate the semantic distance between two meanings of two words. Let a sentence be made up of n words, $m_1, m_2, \ldots, m_n$ and let $m_{i,j}$ be the vertices of a graph representing the j-th meaning of m_i. Then we may build the arcs of this graph by linking $m_{i,j}$ and $m_{i+1,k}$ for any i, j, k with an arc whose weight is the semantic distance between the two involved meanings. Then, we add two vertices, e_b linked to any meaning of m_1 (with a weight 0) and e_e linked to any meaning of m_n (with a weight 0).

Then the problem of finding a pertinent sense to the sentence is to find a shortest path between e_b and e_e. Furthermore, if there are several shortest paths between these two vertices, then there are several meaningful choices for the senses of the words of the sentence (which makes it possible to find automatically ambiguous sentences).

15.4. Shortest path problem

The shortest path problem is one of the oldest and more studied of the graph area. A lot of papers have been written on this subject (see [BER 87, FLO 62, GIB 89,

WAX 88, WID 94] for instance). Some algorithms span graphs with trees or use spanning trees (see [GIL 68] for instance) and the multi-cast problem is also widely studied (see for instance [CHO 02, CRA 97, KOM 93, MOY 94, SAL 95, SAL 97, WAT 96]).

The best algorithm known up to now to solve the shortest path problem is the Dijkstra's algorithm. It runs in time $O(V \ln E)$ where V is the number of vertices of the graph and E the number of edges (see [DIJ 59, JOH 77, WEI 77]). This algorithm has been improved by many authors since 1959.

DEFINITION 15.3.– *Let $G = (V, A)$ be an unweighted graph. Let R be an equivalence relation among the vertices of G. We shall call thickening of G the graph $G'(V', A')$ for the relation R where V' is the set of equivalence relations for R and $\forall V_1, V_2 \in V'$, $(V_1, V_2) \in A' \Leftrightarrow \exists v_1 \in V_1, v_2 \in V_2$ such that $(v_1, v_2) \in A$. In such a case we shall say that G' is a thickening of G or, equivalently, that G is a refinement of G'.*

Let us denote by π the application which maps a vertex v of V onto the vertex v' of G' such that $v \in v'$. Let $v_1, v_2 \in V$ and let us suppose that there exist an integer k and a path from v_1 to v_2 of length k. Then there exists a path of length k between $\pi(v_1)$ and $\pi(v_2)$. Indeed, if $p = (v_0, v_1, \ldots, v_k)$ is a path in G of length k between v_0 and v_k, then $p' = (\pi(v_0), \pi(v_1), \ldots, \pi(v_k))$ is a path of length k in G' between $\pi(v_0)$ and $\pi(v_k)$. Then, we shall call p a refinement of p' or, equivalently, p' a thickening of p. The problem is then to find a path of minimal length in G' which may be refined in G.

Let $G_0 = G, G_1, \ldots, G_t$ be a sequence of graphs such that $\forall i \leqslant t-1$, G_{i+1} is a thickening of G_i. If we call v_i the number of vertices of G_i and a_i its number of arcs, we suppose that $\forall 0 \leqslant i \leqslant t,\ P(\frac{a_i}{v_i^2}) \geqslant \frac{1}{2}$. An algorithm to find the shortest path between two vertices of G, for example, d and a is the following.

```
ShortestPath(G[])
Input: a sequence G[] of thickenings of G_0,
       two vertices of G_0, d and a
Output: the shortest paths between d and a.
Variables: l, th: integer
l ← 0
While no path is found between d and a Do
      P[t] ← all the paths of length l in G[t] between d[t] and a[t]
      th ← t
      While (th > 0 And P[th] is not empty) Do
            Foreach path p in P[th] Do
                  refine p and if it may be refined
                        store its refinement in P[th − 1]
            EndForEach
```

```
            th ← th - 1
      EndWhile
      if P[0] is not empty
         Then the shortest paths are found: output them and exit
      Else l ← l + 1
      EndIf
EndWhile End
```

In $G[t]$ the shortest paths are calculated thanks to a refinement of the BFS algorithm. Let us detail this part.

15.4.1. *Finding paths in* $G[t]$

Let $G = (V, A)$ be an unweighed graph such that $P(\frac{a}{v^2}) < \frac{1}{2}$ with $P(X) = 1 - \frac{3}{2}X + X^2 - \frac{1}{4}X^3$. Let d and a be two of its vertices and let us calculate the shortest paths between d and a. We suppose furthermore that the adjacency vertices of any given vertex are given through a sorted list of vertices.

Let $S \subset V$ be a non-empty subset of V. The cost of the calculation of the set $S' = \{v' \in V, \exists v \in S, (v, v') \in A\}$ is averagely majored by the cardinality of S'. Indeed, the set S' may be stored in a bit vectors in which we add the vertices linked to the elements of S by a bit-or. In a similar way, the set $S'' = \{v'' \in V, \exists v \in S, (v'', v) \in A\}$ is averagely majored by the cardinality of S'' (same reason). The operation consisting of calculating S' (resp. S'') knowing S will be called an increment (resp. decrement). Furthermore, the cost of the intersection of two sorted sets is majored by the sum of their cardinals.

Let us consider the following algorithm, consisting of incrementing $\{d\}$, decrementing $\{a\}$ and intersect them until there exists a non-empty intersection. Then the shortest paths between d and a may be read.

This algorithm may be written in the following way:

```
ShortestPath(G)
Input: a graph G, two vertices of G, d and a
Output: the shortest paths between d and a.
Variables Increments[], Decrements[], inc=0, dec=0, Parity = 0, Paths[]
Increments[Inc] = {d}
Decrements[Dec] = {a}
While (Increments[Inc] ∩ Decrements[Dec] = { })
      if (Parity = 0)
```

```
         Inc = Inc + 1
         Increments[Inc] = Increment(Increments[Inc - 1])
      Else
         Dec = Dec + 1
         Decrements[Dec] = Decrement(Decrements[Dec - 1])
      EndIf
EndWhile
Paths[Inc] = Increments[Inc] ∩ Decrements[Dec]
For I = Inc - 1 Downto 0 Do
   Paths[I] = Decrement(Paths[I + 1]) ∩ Increments[I]
EndFor
For I = Dec - 1 Downto 0 Do
   Paths[Inc + Dec - I] = Increment(Paths[Inc + Dec - I - 1])
   ∩ Decrements[I]
EndFor
ReadPaths(Paths)
End
```

After this treatment, the variable Paths contains sets of vertices and a vertex v lies on a shortest path between d and a, i arcs away from d (and $l - i$ arcs away from d where l is the length of the path) if and only if v belongs to $Paths[i]$.

The ReadPaths algorithm consists of reading the shortest paths through a deep first journey of paths. This last algorithm may be written as:

```
ReadPaths(Paths[], l, Current = 0, Res[] = 0)
Input: The shortest paths given in an array of sets of vertices,
       l the paths lengths
Output: the shortest paths between Paths[0] and Paths[l].
If  (Current = l)
   Output (Res[])
Else
   Foreach vertex v in Paths[Current] Do
      Res[Current] = v
      Memory[Current + 1] = Paths[Current + 1]
      Paths[Current + 1] = Paths[Current + 1] ∩ Increment({v})
      ReadPaths(Paths, l, Current + 1, Res[])
      Paths[Current + 1] = Memory[Current + 1]
   EndForeach
End
```

The complexity of the H algorithm is bounded above by $O(v)$ where v is the number of vertices of the graph because the first steps, consisting of calculating $Paths[]$ are bounded above by $O(v)$ and the ReadPaths algorithm is also bounded above by the number of vertices involved in the shortest paths, v.

As an example, let us calculate all the shortest paths between `S` and `W` in the following graph (see Figure 15.7).

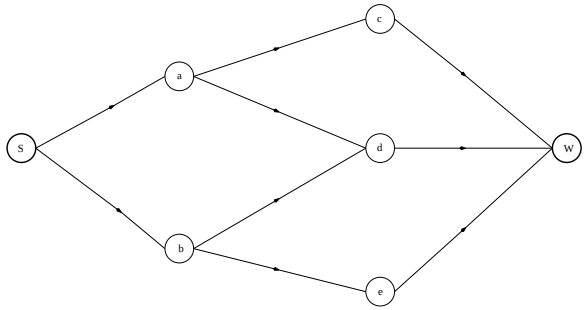


Figure 15.7. *The graph G_0 itself*

Then the vertices are gathered, for instance, two by two and we obtain the graph in Figure 15.8.

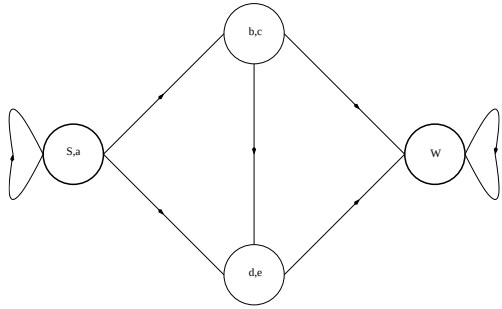


Figure 15.8. *A thickening G_1 of G_0*

By grouping again the vertices two by two we obtain the graph in Figure 15.9.

We stop here because $\frac{a_2}{a_1} \leqslant \frac{1}{2}$.

– The first step is to look for a path of length 0, which does not exist in G_0 because it does not exist in G_2.

– Now there is a path of length 1 between $\overline{S}$ and $\overline{W}$, which is $\overline{S} \rightarrow \overline{W}$. This path may not be refined in G_1 because there is no path in G_1 between $\overline{S}$ and $\overline{W}$.

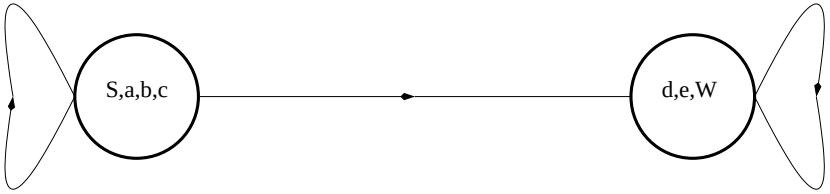


Figure 15.9. *A thickening G_2 of G_1*

– Let us look for paths of length 2 between $\overline{S}$ and $\overline{W}$ in G_2. There are two such paths, $P_1 = \overline{S} \to \overline{S} \to \overline{W}$ and $P_2 = \overline{S} \to \overline{W} \to \overline{W}$. The first of these two paths may be refined in G_1 in $P_1' = \overline{S} \to \overline{b} \to \overline{W}$ and P_2 may be refined in G_1 in $\overline{S} \to \overline{d} \to \overline{W}$. Now neither the path P_1' nor P_2' may be refined in G_0.

– Let us look for paths of length 3. There are three paths of length 3 in G_2 between $\overline{S}$ and $\overline{W}$: $P_1 = \overline{S} \to \overline{S} \to \overline{S} \to \overline{W}$, $P_2 = \overline{S} \to \overline{S} \to \overline{W} \to \overline{W}$ and $P_3 = \overline{S} \to \overline{W} \to \overline{W} \to \overline{W}$. The path P_1 may be refined in $P_1' = \overline{S} \to \overline{S} \to \overline{b} \to \overline{W}$. The path P_2 may be refined in three different paths, $P_2^1 = \overline{S} \to \overline{b} \to \overline{W} \to \overline{W}$, $P_2^2 = \overline{S} \to \overline{b} \to \overline{d} \to \overline{W}$, and $P_2^3 = \overline{S} \to \overline{S} \to \overline{d} \to \overline{W}$. The path P_3 may be refined in G_1 in $P_3' = \overline{S} \to \overline{d} \to \overline{W} \to \overline{W}$. Now let us refine these paths in G_0. The path P_1' may be refined in $P_1'' = S \to a \to c \to W$. The path P_2^1 may not be refined in G_0. The path P_2^2 may be refined in $S \to b \to d \to W$ and in $S \to b \to e \to W$. The path P_2^3 may be refined in $S \to a \to d \to W$. The path P_3' may not be refined in G_0.

So we find the shortest paths between S and W, which are: $S \to a \to c \to W$, $S \to b \to d \to W$, $S \to b \to e \to W$and $S \to a \to d \to W$.

It is clear that this algorithm can be used to find the paths of a given length between two vertices of a graph.

15.4.2. *Complexity*

In [KOS 02], the author calculates the complexity of the algorithm presented roughly in the preceding section. This complexity is $O(v)$ where v is the number of vertices of the graph. We give here the proof of this complexity in the particular case that is useful in the context of flows calculations, i.e. the unweighted graphs.

Let $G_0 = (V, A)$ be an unweighted graph and let us denote by v its number of vertices and a its number of arcs. We suppose in this section that the equivalence classes are sets of two vertices (and at most a class made of a single vertex).

LEMMA 15.1.– *Let G_1 be a thickening of G_0. Then the number of vertices of G_1 is $\lceil \frac{v}{2} \rceil$ and its number of arcs is averagely $aP(\frac{a}{v^2})$ with $P(X) = 1 - \frac{3}{2}X + X^2 - \frac{1}{4}X^3$.*

Proof. The number of vertices of a thickening of G_0 in which the vertices are grouped by two is obviously $v' = \lceil \frac{v}{2} \rceil$. There is no arc between two vertices v' and w' of G_1 if and only if there is no arc between four pairs of vertices (if $v = (v_1, v_2)$ and $w = (w_1, w_2)$, the four pairs are $(v_i, w_j)_{(1 \leqslant i,j \leqslant 2)}$ and we may suppose that the probability of any of these pairs to be an arc does not depend on the probability of the others). The number of arcs is hence $a' = \lceil \frac{v}{2} \rceil^2 (1 - (1 - \frac{a}{v^2})^4)$. So $a' = aP(\frac{a}{v^2})$ with $P(X) = 1 - \frac{3}{2}X + X^2 - \frac{1}{4}X^3$. ∎

REMARK 15.2.– The ratio $\frac{a}{v^2}$ increases in refinements.

Let us prove this remark. Let G be a graph and G' be one of its refinements in which the vertices have been gathered by two. Let us denote by a, v, a' and v' the number of arcs and vertices of G and G'.

We have $\frac{a'}{v'^2} = 4P(\frac{a}{s^2})\frac{a}{v^2}$. Since $P(x)$ decreases from 1 towards $\frac{1}{4}$ when x raises from 0 to 1, the remark is proved.

LEMMA 15.2.– *The average number of paths of length $l \in \mathbb{N}$ between two vertices in G_0 is $N(G_0, l) = \frac{1}{v}(\frac{a}{v})^l$.*

Proof. By induction on the length of the paths. The graph contains a arcs and v vertices. The formula is correct for paths of length 0 (the average number of paths of length 0 between two vertices is $\frac{1}{v}$). The average number of paths of length 1 between two vertices is hence $\frac{a}{v^2}$. If we suppose that the average number of paths of length l between two vertices is $\frac{1}{v}(\frac{a}{v})^l$, then each of these paths may averagely be prolonged in $\frac{a}{v}$ different arcs and the number of paths of length $l + 1$ between two vertices is averagely $\frac{1}{v}(\frac{a}{v})^{l+1}$. ∎

We now discuss some refinements. An algorithm to refine a path from G_{k+1} to G_k is the following:

```
RefinePath(Graph G_{k+1}, Graph G_k, Path P)
Input: a graph G_k, a refinement of G_k, G_{k+1}, a path P in G_k
Output: All the refinements of P in G_{k+1}
If the first arc of P is refinable
       Truncate P by its first arc (a,b)
       Foreach refinement (c,d) of (a,b)
               Answer.FirstElement =(c,d)
               Answer.Catenate(Refine(G_{k+1}, G_k, P)
       EndForEach
       Put back its first arc to P
EndIf
End
```

LEMMA 15.3.– *Let G be a graph containing v vertices and a arcs, and G' be one of its thickenings in which the vertices are gathered two by two. Let v_1 and v_2 be two vertices of G and let us suppose that there exists a path $w = (w'_0 = v'_1, w'_1, \ldots, w'_l = v'_2)$ of length l (we denote as usually by v'_1, resp. v'_2, the equivalence class of v_1, resp. v_2, in G'). Then there are averagely $(\frac{2}{v})^{l-1}(\frac{a}{v})^l$ paths of length l in G between v_1 and v_2 which are refinements of the path w.*

Proof. The number of possible refinements of w is 2^{l-1}. Each of these possible paths has a probability $\frac{1}{v^{l-1}}(\frac{a}{v})^l$ to exist. So the number of refinements of w is averagely $(\frac{2}{v})^{l-1}(\frac{a}{v})^l = \frac{v}{2}(\frac{2a}{v^2})^l$. ∎

The most expansive operation while refining a path from G' to G is to check whether a given arc exists in G. The complexity of the refinement is hence majored by this number of checks.

LEMMA 15.4.– *The number of checks while refining a path of length l from G to G' is $\sum_{i=0}^{l} \frac{v}{2}(\frac{2a}{v^2})^i = \frac{v}{2}\frac{1-(\frac{2a}{v^2})^{l+1}}{1-\frac{2a}{v^2}}$.*

Proof. When we have refined paths of length $i-1$, we have only to extend the found paths by one arc. The number of checks is hence the sum of the number of paths of all the lengths from 0 to l. ∎

Let us detail an hierarchical algorithm (H-algorithm):

```
ShortestPath(Graph G[0..M], int o, int e)
Input: a sequence of graphs, one being a thickening
         of the preceding, all verifying P(a/v²) ⩾ 1/2,
       an origin vertex index o
       an extremity vertex index e
Output: the shortest paths between v_o and v_e.
Variables:
          l: Path's length
l ← 0
While the shortest paths qre not found Do
   Find the shortest paths P_i in G_M thanks to the refined BFS
   Foreach shortest path P
      For k = M - 1 Downto 0
         P[i, k-1] = Refine(G[k+1], G[K], P[i, k])
      EndFor
      If there exists a path in G[0] output the paths and exit
      EndIf
   EndForEach
EndWhile
```

THEOREM 15.1.– *The mean complexity of the calculation of the shortest path between two vertices of a graph by using the H-algorithm is $C(G) \leqslant v$.*

Proof. The mean complexity of the refinements of paths of length l from a graph made of a single vertex to the initial graph G_0 is:

$$\begin{aligned} C(l) &= \sum_{i=0}^{\ln v} \frac{v_i}{2} \left(\frac{2a_i}{v_i^2}\right)^l \\ &\leqslant \sum_{i=0}^{\ln v} \frac{v_i}{2} \\ &\leqslant \sum_{i=0}^{\ln v} \frac{v_0}{2^{i+1}} \\ &< v_0 \end{aligned}$$

The complexity of the shortest path calculation is majored by the complexity of the calculations of the refinements of paths of length l. ∎

15.5. The flow problem

We now present a new algorithm for maximizing the flow over a network.

15.5.1. *The Ford-Fulkerson algorithm*

The celebrated Ford-Fulkerson algorithm can be specified as follows:

```
MaximumFlow(G)
input: a graph G, two vertices s and t
output: a flow Φ = (φ_{a_1}, ..., φ_{a_|A|})
(0) Initialisation: φ_(a) = 0, ∀a ∈ A
                    v(Φ) = 0
                    i=0
                    G_R^0 = G
(1) Find a shortest path from s to t in the residual graph G_R^i
(2) If there is no such path then END
(3)          Else μ ←a shortest path
                     (consisting of the fewest number of arcs)
                     from s to t in G_R^i
(4)               cap(μ)= the minimum capacity of residual
                     capacities of μ in G_R^i
(5)               v(Φ) ← v(Φ)+ cap(μ)
(6)               ∀a ∈ μ ∩ A, φ_(a) ← φ_(a) + cap(μ)
(7)               ∀a ∈ (μ ∉ μ ∩ A), φ_(a) ← φ_(a) − cap(μ)
(8)               i ← i + 1
(9)               Build the new residual graph G_R^i
```

The initial algorithm of Ford and Fulkerson, in [FOR 56], runs in pseudo-polynomial time. The implementation proposed here is due to Edmonds and Karp, [EDM 72] and runs in $O(|V|^2|A|)$ time. There are other implementations of the Ford-Fulkerson algorithm, which are detailed in [AHU 93].

Radix trees and shortest paths

The main step of the preceding algorithm is step 1, which may be performed thanks to the H-algorithm. The complexity of the Ford-Fulkerson algorithm is $O(n^3)$ when this step is done by the use of the Dijkstra's algorithm, whose complexity is $O(n^2)$. the use of the H-algorithm allows a mean complexity in $O(n^2)$.

As an example, let us consider the graph of Figure 15.10 in which we wish to optimize the flow between the source S and the well W.

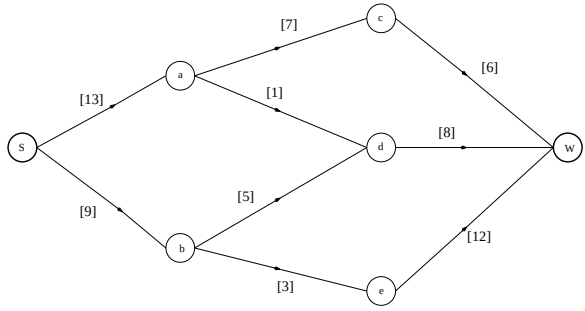


Figure 15.10. *An optimization problem*

The shortest paths in terms of number of arcs are $P_1 = S \rightarrow a \rightarrow c \rightarrow W$, $P_2 = S \rightarrow a \rightarrow d \rightarrow W$, $P_3 = S \rightarrow b \rightarrow d \rightarrow W$ and $P_4 = S \rightarrow b \rightarrow e \rightarrow W$.

After the path P_1 (of flow 6) is considered as a part of the flow, the residual graph is that of Figure 15.11.

After that paths P_2 (capacity 1), P_3 (capacity 5) and P_4 (capacity 3) are included in the flow, the residual graph is given in Figure 15.12.

Since in this last residual graph the source and the well are no longer connected, the maximal flow is reached, with a full capacity of $6 + 1 + 5 + 4 = 16$.

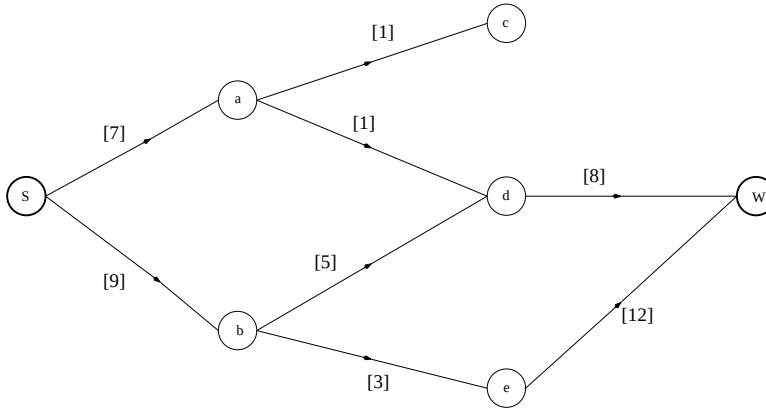


Figure 15.11. *After a first path part of the flow*

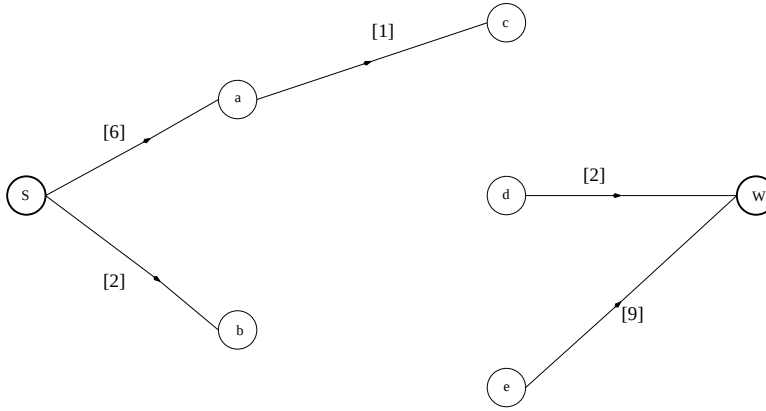


Figure 15.12. *The residual graph after the shortest paths are included in the flow*

15.6. Conclusion

The flow problem has important applications: transportation of energy or water, schedule of network tasks, etc. The algorithm presented in this chapter makes it possible to calculate the optimal flow over a network in a mean time of $O(n^2)$ while the Ford-Fulkerson algorithm has a complexity in $O(n^3)$ where n is the number of vertices of the graph.

15.7. Bibliography

[AHU 93] AHUJA R.K., MAGNANTI T.L, ORLIN J.B., *Network flows: theory, algorithms and applications*, Prentice Hall, New Jersey, 1993.

[BER 87] BERTSEKAS D., GALLAGER R., *Data Networks*, Prentice-Hall, Inc., 1987.

[CHO 02] CHOPLIN S., *Dimensionnements de Résaux Virtuels de Télécommunications*, Sophia-Antipolis University, PhD thesis, November 2002.

[CRA 97] CRAWFORD J.S., WATERS A.G., Low Cost Quality of Service Multicast Routing in High Speed NEtworks, Technical Reports 13-97, University of Kent at Canterbury, December 1997.

[DIJ 59] DIJKSTRA E W., "A note on two problems in connexion with graphs", *Numer. Math.*, vol. 1, p. 269-271, 1959.

[EDM 72] EDMONDS J., KARP R.M., "Theoretical improvements in algorithmic efficiency for network flow problems", *Journal of ACM*, vol. 19, p. 248-264, 1972.

[FLO 62] FLOYD R. W., "Algorithm 97, Shortest Path", *Communications of the ACM*, vol. 5, num. 6:345, 1962.

[FOR 56] FORD L.R., FULKERSON D.R., "Maximum flow through a network", *Canadian Journal of Mathematics*, vol. 8, p. 399-404, 1956.

[GIB 89] GIBBONS A., *Algorithmic Graph Theory*, Cambridge University Press, 1989.

[GIL 68] GILBERT E. N., POLLACK H. O., "Steiner Minimal Trees", *SIAM Journal on Applied Mathematics*, vol. 16, 1968.

[JOH 77] JOHNSON D. B., "Efficient Algorithms for Shortest Paths in Sparse Networks", *Journal of the ACM (JACM)*, vol. 24, num. 1, p. 1-13, 1977.

[KOM 93] KOMPELLA P., *Multicast Routing Algorithms for Multimedia Traffic*, PhD Thesis, University of California, San Diego, USA, 1993.

[KOS 01] KOSKAS M., A Divide and Conquer Algorithm of Image Pattern Finding, Manuscript.

[KOS 02] KOSKAS M., A Divide And Conquer Algorithm To Solve The Shortest Path Problem, Manuscript.

[KOS 04] KOSKAS M., A Hierarchical Database Manager, Annales du Lamsade, vol. 2, p. 277-317, 2004.

[MOY 94] MOY J., "Multicast Extensions to OSPF", *RCF 1584*, march 1994.

[SAL 95] SALAMA H.F., REEVES D.S., VINITOS I., TSANG-LIN S., "Evaluation of Multicast Routing Algorithms for Real-Time Communications on High Speed Networks", *Proceedings of the 6-th IFIP Conference on High Performance Networks (HPN'95)*, 1995.

[SAL 97] SALAMA H.F., REEVES D.S., VINITOS I., "Evaluation of Multicast Routing Algorithms for Real-Time Communication on High Speed Networks", *IEEE Journal on Selected Area in Communications*, vol. 15, num. 3, p. 332-345, April 1997.

[WAT 96] WATERS A. G., CRAWFORF J.S., Low-Cost ATM Multimedia Routing with Constrained Delays, Multimedia Telecommunications and Applications (3rd COST 237 Workshop, Barcelona, Spain), 23-40, Springer, November, 1996.

[WAX 88] WAXMAN B.M., "Routing of Multipoint Connection", *IEEE journal on selected areas in communications*, vol. 6, num. 9, p. 1617-1622, 1988.

[WEI 77] WEIDE B., "A Survey of Analysis Techniques for Discrete Algorithms", *ACM Computing Surveys (CSUR)*, vol. 9, num. 4, p. 291-313, December 1977.

[WID 94] WIDYONO R., The Design and Evaluation of Routing Algorithms for Real-Time Channels, Tr-94-024, University of California at Berkeley and International Computer Science Institute, September 1994.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 16

The Complexity of the Exact Weighted Independent Set Problem

16.1. Introduction

Suppose that we have a well-solved optimization problem, such as minimum spanning tree, maximum cut in planar graphs, minimum weight perfect matching or maximum weight independent set in a bipartite graph. How hard is it to determine whether there exists a solution with a given weight? Papadimitriou and Yannakakis showed in [PAP 82] that these so-called *exact* versions of the above optimization problems are **NP**-complete when the weights are encoded in binary. The question is then, what happens if the weights are "small", i.e., encoded in unary? Contrary to the binary case, the answer to this question depends on the problem.

– The *exact spanning tree* problem and, more generally, the *exact arborescence* problem are solvable in pseudo-polynomial time [BAR 87].

– The *exact cut* problem is solvable in pseudo-polynomial time for planar and toroidal graphs [BAR 87].

– The *exact perfect matching* problem is solvable in pseudo-polynomial time for planar graphs [BAR 87] and, more generally, for graphs that have a Pfaffian orientation[1] (provided one is given). We recall that a *matching* of a graph $G = (V, E)$ is a set E' of pairwise non-adjacent edges of G. If $|E'| = |V|/2$, then E' is said to be a *perfect matching* of G. Karzanov [KAR 87] gives a polynomial-time algorithm for the

Chapter written by Martin MILANIČ and Jérôme MONNOT.

1. Bipartite graphs with a Pfaffian orientation have been characterized in [THO 06], where a polynomial-time recognition algorithm is also presented.

special case of the exact perfect matching problem, when the graph is either complete or complete bipartite, and the weights are restricted to 0 and 1. Papadimitriou and Yannakakis show in [PAP 82] that the problem for general (or bipartite) graphs with weights encoded in unary is polynomially reducible to that with 0-1 weights. Mulmuley, Vazirani and Vazirani [MUL 87] show that the exact perfect matching problem has a randomized pseudo-polynomial-time algorithm.

The exact perfect matching problem is of great practical importance. It has applications in such diverse areas as bus-driver scheduling, statistical mechanics (see [LEC 86]), DNA sequencing [BŁA 06], and robust assignment problems [DEĬ 06]. The problem consists of determining whether a given edge-weighted graph contains a perfect matching of a given weight. Despite polynomial results for special cases, the deterministic complexity of the exact perfect matching problem remains unsettled for general graphs and even for bipartite graphs. Papadimitriou and Yannakakis conjectured that the problem is **NP**-complete [PAP 82].

This open problem motivates us to introduce and study the *exact weighted independent set* problem and a restricted version of it, which are both closely related to the exact perfect matching problem.

An *independent set* (sometimes called *stable set*) in a graph is a set of pairwise non-adjacent vertices. The *weighted independent set* (WIS) problem consists of determining an independent set of maximum weight in a given weighted graph (G, w). If all weights are the same, we speak about the *independent set* (IS) problem. The optimal values of these problems are denoted by $\alpha_w(G)$ and $\alpha(G)$, respectively.

The *exact weighted independent set* (EWIS) problem consists of determining whether a given weighted graph (G, w) with $G = (V, E)$ and $w : V \to \mathbb{Z}$ contains an independent set whose total weight (i.e., the sum of the weights of its members) equals a given integer b. Formally, the solution to the EWIS problem, given (G, w, b), is *yes* if and only if there is an independent set I of G with $w(I) = b$. The restriction where we require the independent set to be of a maximum independent set of the graph will be called the *exact weighted maximum independent set problem* and denoted by EWIS_α. Thus, $\text{EWIS}_\alpha(G, w, b)$ consists of deciding upon the existence of an independent set I of G with $|I| = \alpha(G)$ and $w(I) = b$.

The connection between the exact perfect matching problem and the exact weighted independent set problem is best understood through line graphs. The *line graph* $L(G)$ of a graph $G = (V, E)$ is the graph whose vertex set is E, and whose two vertices are adjacent if and only if they share a common vertex as edges of G. Clearly, there is a one-to-one correspondence between the matchings of a graph and the independent sets of its line graph. The *exact matching* problem, i.e., the problem of determining whether a given edge-weighted graph contains a matching of a given weight, is then precisely the exact weighted independent set problem, which

is restricted to the class of line graphs. Similarly, under the (polynomially verifiable) assumption that the input graph has a perfect matching, the exact perfect matching problem is precisely the exact weighted maximum independent set problem, which is restricted to the class of the line graphs of graphs with a perfect matching.

In this chapter, we focus on the problem of determining the complexities of the EWIS and EWIS_α problems for particular graph classes. On the one hand, we present the first non-trivial strong **NP**-completeness result for these problems. On the other hand, we distinguish several classes of graphs where the problems can be solved in pseudo-polynomial time.

More specifically, we can summarize the main results of this chapter in the spirit of the above list of complexity results on exact problems:

– The *exact weighted independent set* and the *exact weighted maximum independent set* problems are strongly **NP**-complete for cubic bipartite graphs [2].

– The *exact weighted independent set* and the *exact weighted maximum independent set* problems are solvable in pseudo-polynomial time in each of the following graph classes:

- mK_2*-free graphs*,
- *interval graphs* and their generalizations k*-thin graphs*,
- *circle graphs*,
- *chordal graphs*,
- *AT-free graphs*,
- $(claw, net)$*-free graphs*,
- *distance-hereditary graphs*,
- *graphs of bounded treewidth*,
- *graphs of bounded clique-width*,
- certain *subclasses of* P_5*-free and fork-free graphs*.

The results for subclasses of P_5-free and fork-free graphs are derived by means of modular decomposition. The application of modular decomposition to the EWIS problem is described in section 16.4.2 and may be of independent interest.

In view of the relation between the exact perfect matching problem and the exact weighted maximum independent set problem, each of the above polynomial results also gives a polynomial result for the exact perfect matching problem. Whenever the EWIS_α problem is (pseudo-) polynomially solvable for a class of graphs $\mathcal{G}$, the exact perfect matching problem is (pseudo-) polynomially solvable for graphs in the set $\{G : L(G) \in \mathcal{G}\}$. For example:

2. We also strengthen this result considerably, however, we postpone the detailed formulation until section 16.3.2.

– The *exact perfect matching* problem is solvable in pseudo-polynomial time for graphs of bounded treewidth.

Notation and organization. All graphs considered are finite, simple and undirected. Unless otherwise stated, n and m will denote the number of vertices and edges, respectively, of the graph considered. As usual, P_n and C_n denote the chordless path and the chordless cycle on n vertices. For a graph G, we will denote by $V(G)$ and $E(G)$ the vertex-set and the edge-set of G, respectively. Individual edges will be denoted by square brackets: an edge with endpoints u and v will be denoted by $[u, v]$. For a vertex x in a graph G, we denote by $N_G(x)$ the neighborhood of x in G, i.e., the set of vertices adjacent to x, and by $N_G[x]$ the closed neighborhood of x, i.e., the set $N_G(x) \cup \{x\}$. We will write $N(x)$ and $N[x]$ instead of $N_G(x)$ and $N_G[x]$ if no confusion can arise.

We say that a graph H is an *induced subgraph* of G if H can be obtained from G by deletion of some (possibly none) vertices; the subgraph of G *induced by* $U \subseteq V(G)$ is the graph obtained from G by deleting the vertices from $V(G) \backslash U$ and it will be denoted by $G[U]$. For a graph G, we denote by co-G (also $\overline{G}$) the edge-complement of G. By K_n we denote the complete graph on n vertices, and by $K_{s,t}$ the complete bipartite graph with parts of size s and t. By component we will always mean a connected component. For graph-theoretical terms not defined here, see [BER 73]. For a subset of vertices $V' \subseteq V$, we let $w(V') = \sum_{v \in V'} w(v)$. For a positive integer k, we write $[k]$ for the set $\{1, \ldots, k\}$.

The triple (G, w, b) will always represent an instance of the EWIS (or EWIS$_\alpha$) problem, i.e., $G = (V, E)$ is a graph, $w : V \to \mathbb{Z}$ are vertex weights, and $b \in \mathbb{Z}$ is the target weight. If H is an induced subgraph of G, we will also consider triple of the form (H, w, b) as instances of EWIS, with the weights w representing the restriction of w to $V(H)$. We will denote by EWIS(G, w, b) the solution to the instance (G, w, b) of the EWIS problem, i.e., EWIS(G, w, b) is *yes* if there is an independent set I in G with $w(I) = b$, and *no* otherwise. Similarly, EWIS$_\alpha(G, w, b)$ is *yes* if there is a maximum independent set I in G with $w(I) = b$, and *no* otherwise.

The chapter is organized as follows. In section 16.2, we continue the introductory discussion and present some polynomial preprocessing steps that simplify the input and which we will later on assume as performed. We also discuss some relations between the complexities of the problems WIS, EWIS and EWIS$_\alpha$. Section 16.3 is devoted to the strong **NP**-completeness results. In section 16.4, pseudo-polynomial time solutions to the exact weighted independent set problem are presented. We conclude the chapter with a short discussion in section 16.5 that places the class of line graphs of bipartite graphs between two graph classes with known complexities of the EWIS problem.

16.2. Preliminary observations

The exact weighted independent set problem is (weakly) **NP**-complete for any class of graphs containing the edgeless graphs $\{\overline{K_n} : n \geqslant 0\}$. There is a direct equivalence between the exact weighted independent set problem on $\{\overline{K_n} : n \geqslant 0\}$ and the subset sum problem, which is known to be **NP**-complete (see [GAR 79]). The *subset sum* problem is the following: given n integers $a_1, \ldots, a_n$ and a bound b, determine whether there is a subset $J \subseteq [n]$ such that $\sum_{j \in J} a_j = b$.

Therefore, for a given class of graphs $\mathcal{G}$, the question of interest is whether the EWIS problem is strongly **NP**-complete for graphs in $\mathcal{G}$, or is it solvable in pseudo-polynomial time.

First, let us observe that in any class of graphs $\mathcal{G}$, we may restrict our attention to instances with positive weights only.

REMARK 16.1.– The EWIS problem with arbitrary integer weights is polynomially equivalent to the EWIS problem, which is restricted to instances (G, w, b) such that $b \leqslant w(V(G))$ and $1 \leqslant w(v) \leqslant b$, for all $v \in V$. The same equivalence holds true for the EWIS_α problem.

Proof. Solving the EWIS problem for any particular instance amounts to solving n problems EWIS_k, in which the independent sets are restricted to be of size k, for all $k \in [n]$ (unless $b = 0$, in which case the solution is trivial, as the empty set is an independent set of weight 0). The weights in EWIS_k can be assumed to be positive: otherwise, we can add a suitably large constant N to each vertex weight and replace b by $b + kN$ to get an equivalent EWIS_k problem with positive weights only. Finally, applying the same transformation again with $N = w(V) + 1$ reduces the problem EWIS_k to a single EWIS problem with positive weights. Repeating this for all values of $k \in [n]$, the result follows.

Finally, if all vertex weights are positive, we can delete from the graph all vertices whose weight exceeds b, as they will never appear in a solution. Furthermore, the solution is clearly *no* if $b > w(V)$.

The same assumption on vertex weights as for the EWIS problem can also be made for the instances (G, w, b) of its restricted counterpart EWIS_α. Again, if some of the weights are negative, we can modify the weights and the target value as we did above for EWIS_k. Now we only do it for $k = \alpha(G)$. Note that we can calculate $\alpha(G)$ as that only $p \in [n]$ such that the value of $\text{EWIS}_\alpha(G, \mathbf{1}, p)$ is *yes*, where $\mathbf{1}$ denotes the unit vertex weights. ∎

We now discuss some relations between the complexities of the problems WIS, EWIS and EWIS_α, when restricted to particular graph classes.

LEMMA 16.1.– *Let $\mathcal{G}$ be a class of graphs. The following statements are true:*
(i) if the EWIS$_\alpha$ *problem is solvable in pseudo-polynomial time for graphs in $\mathcal{G}$, then the* WIS *problem is solvable in pseudo-polynomial time for graphs in $\mathcal{G}$;*
(ii) if the EWIS *problem is solvable in pseudo-polynomial time for graphs in $\mathcal{G}$, then the* EWIS$_\alpha$ *problem is solvable in pseudo-polynomial time for graphs in $\mathcal{G}$;*
(iii) let $\mathcal{G}' = \{G' : G \in \mathcal{G}\}$ where $G' = (V', E')$ is the graph, obtained from a graph $G = (V, E) \in \mathcal{G}$, by adding pendant vertices, as follows: $V' = V \cup \{v' : v \in V\}$, $E' = E \cup \{[v, v'] : v \in V\}$. If the EWIS$_\alpha$ *problem is solvable in pseudo-polynomial time for graphs in $\mathcal{G}'$, then the* EWIS *problem is solvable in pseudo-polynomial time for graphs in $\mathcal{G}$.*

Proof. (i) Let (G, w, k) be an instance of the decision version of the weighted independent set problem. As we can assume positive weights, G contains an independent set of total weight which is at least k if and only if G contains a maximum independent set of total weight which is at least k. By testing values for b from $w(V)$ down to k and using an algorithm for the EWIS$_\alpha$ problem on the instance (G, w, b), we can decide whether G contains a maximum independent set of total weight which is at least k.

(ii) Let (G, w, b) be an instance of the EWIS$_\alpha$ problem. It is easy to see that the following algorithm solves EWIS$_\alpha$.

Step 1. Calculate $\alpha(G)$, which is equal to the maximum $k \in [n]$ such that the value of EWIS$(G, \mathbf{1}, k)$ is *yes*, where $\mathbf{1}$ denotes the unit vertex weights.

Step 2. Let $N = w(V)+1$. For every vertex $v \in V(G)$, let $w'(v) = w(v)+N$. Let $b' = b + \alpha(G)N$. Then it is easy to verify that EWIS$_\alpha(G, w, b) =$ EWIS(G, w', b').

(iii) Let (G, w, b) with $G = (V, E) \in \mathcal{G}$ be an instance of the exact weighted independent set problem. Let G' be the graph, which is defined as in the lemma. Let $n = |V(G)|$ and let $w'(v) = (n + 1)w(v)$ for all $v \in V$ and $w'(v) = 1$ for $v \in V'$. Then, it is easy to verify that the value of EWIS(G, w, b) is *yes* if and only if the value of EWIS$_\alpha(G', w', b')$ is *yes* for some $b' \in \{(n + 1)b, \ldots, (n + 1)b + n - 1\}$. ∎

The problem EWIS is clearly in **NP** and so is EWIS$_\alpha$ for any class of graphs $\mathcal{G}$ where IS is polynomially solvable. Therefore, Lemma 16.1 implies the following result.

COROLLARY 16.1.– Let $\mathcal{G}$ be a class of graphs. The following statements are true.
(i) If the WIS problem is strongly **NP**-complete for graphs in $\mathcal{G}$, then the EWIS$_\alpha$ problem is strongly **NP**-hard for graphs in $\mathcal{G}$. If, in addition, the IS problem is polynomial for graphs in $\mathcal{G}$, then the EWIS$_\alpha$ problem is strongly **NP**-complete for graphs in $\mathcal{G}$.
(ii) If the EWIS$_\alpha$ problem is strongly **NP**-hard for graphs in $\mathcal{G}$, then the EWIS problem is strongly **NP**-complete for graphs in $\mathcal{G}$.

(iii) Let $\mathcal{G}'$ be as in Lemma 16.1. If the EWIS problem is strongly **NP**-complete for graphs in $\mathcal{G}$, then the EWIS_α problem is strongly **NP**-hard for graphs in $\mathcal{G}'$. If, in addition, the IS problem is polynomial for graphs in $\mathcal{G}'$, then the EWIS_α problem is strongly **NP**-complete for graphs in $\mathcal{G}'$.

Thus, we are mainly interested in determining the complexity (strong **NP**-complete or pseudo-polynomial results) of the exact weighted independent set problem in those classes of graphs where the weighted independent set problem is solvable in pseudo-polynomial time. Moreover, combining parts (ii) and (iii) of the lemma shows that when $\mathcal{G} \in \{\text{forests, bipartite graphs, chordal graphs}\}$, the problems EWIS and EWIS_α are equivalent (in the sense that, when restricted to the graphs in $\mathcal{G}$, they are either both solvable in pseudo-polynomial time, or they are both strongly **NP**-complete). Recall that a graph G is a *forest* if it is acyclic, *bipartite* if any cycle of G has even length, and *chordal* if any cycle of G with size of at least 4 has a chord (i.e., an edge connecting two non-consecutive vertices of the cycle).

We conclude this section by showing that a similar equivalence remains valid for the class of line graphs. More precisely, if L, $L(Bip)$, $L(K_{2n})$ and $L(K_{n,n})$ denote the classes of line graphs, line graphs of bipartite graphs, line graphs of complete graphs with an even number of vertices and line graphs of complete balanced bipartite graphs, respectively, we have the following result.

LEMMA 16.2.– *The* EWIS *problem is strongly* ***NP****-complete for graphs in* L *(resp.,* $L(Bip)$*) if and only if the* EWIS_α *problem is strongly* ***NP****-complete for graphs in* $L(K_{2n})$ *(resp.,* $L(K_{n,n})$*).*

Proof. The backward implication is given by part (ii) of Lemma 16.1. The forward implication follows from a reduction of the exact matching problem to the exact perfect matching problem which we show now. Given an instance $G = (V, E)$ with edge weights w and a target b for the exact matching problem, construct an instance $(K_{n'}, w', b')$ for the exact perfect matching problem as follows. If $n = |V|$ is odd, we add a new vertex and we complete the graph G. For an edge e of G, let $w'(e) = Nw(e)$ where $N = w(E) + 1$, for an edge $e \notin E$ let $w'(e) = 1$. The transformation is clearly polynomial and G has a matching of weight b if and only if $K_{n'}$ has a perfect matching of weight $Nb + k$ for some value of $k \in \{0, \ldots, n-1\}$. Also, it is easy to see that in the case of bipartite graphs $G = (L, R; E)$ with $|L| \leqslant |R|$, we can add $|R \setminus L|$ vertices to L to balance the bipartition. ∎

16.3. Hardness results

The weighted independent set problem is solvable in polynomial time for bipartite graphs by network flow techniques. However, as we show in this section, the exact version of the problem is strongly **NP**-complete even for cubic bipartite graphs.

16.3.1. *Bipartite graphs*

A bipartite graph is a graph $G = (V, E)$ whose vertex set admits a partition $V = L \cup R$ into the left set L and the right set R such that any edge of G connects a vertex of L to a vertex of R. In general, a bipartite graph may admit several such partitions. Since we only consider connected bipartite graphs (which have a unique such partition, up to switching the parts), we will also write $G = (L, R; E)$.

The strong **NP**-completeness of the EWIS problem in bipartite graphs follows from a straightforward reduction from the balanced biclique problem which is known to be **NP**-complete [GAR 79]. The *balanced biclique* problem consists of determining whether, given a bipartite graph $G = (L, R; E)$ and an integer k, there are subsets $L' \subseteq L$ and $R' \subseteq R$ with $|L'| = |R'| = k$ such that the subgraph induced by $L' \cup R'$ is a complete bipartite subgraph (also called *biclique* of size k). In [DAW 01], a variation of this latter problem is introduced where we must have $|L'| = a$ and $|R'| = b$ (called the *biclique* problem). From an instance G and k of balanced biclique, we introduce weight 1 on each vertex of L, weight $B = \max\{|L|, |R|\} + 1$ on each vertex of R and we set $b = k + Bk$. It is clear that there is an independent set in $(L, R; (L \times R) \setminus E)$ with weight b if and only if there is a balanced biclique in $(L, R; E)$ of size k.

We now strengthen this result by proving that the EWIS_α problem is strongly **NP**-complete for cubic bipartite graphs. By contrast, for graphs of maximum degree 2, EWIS and EWIS_α are pseudo-polynomially solvable problems. Every connected graph in this class is either a cycle or a path, and the treewidth of such graphs is at most 2[3]. According to Corollary 16.3 and Theorem 16.14 from section 16.4, the problem is solvable in pseudo-polynomial time in this class.

THEOREM 16.1.– *The* EWIS_α *problem is strongly* ***NP****-complete in cubic bipartite graphs.*

Proof. The problem is clearly in **NP**, as the IS problem is solvable in polynomial time for bipartite graphs. The hardness reduction is made from the decision version of the clique problem in regular graphs which is known to be **NP**-complete (see [GAR 79]).

A *clique* V^* is a subset of vertices of G such that the subgraph induced by V^* is complete. Let (G, k) be an input to the clique problem, where $G = (V, E)$ is a Δ-regular graph on n vertices and let k be an integer. Without loss of generality, assume that $0 < k < \Delta < n - 1$, since otherwise the problem is easy.

We build the instance $I = (G', w)$ of the EWIS_α problem where $G' = (L, R; E')$ is a bipartite graph as follows:

3. For the definition of treewidth, see section 16.4.1.8.

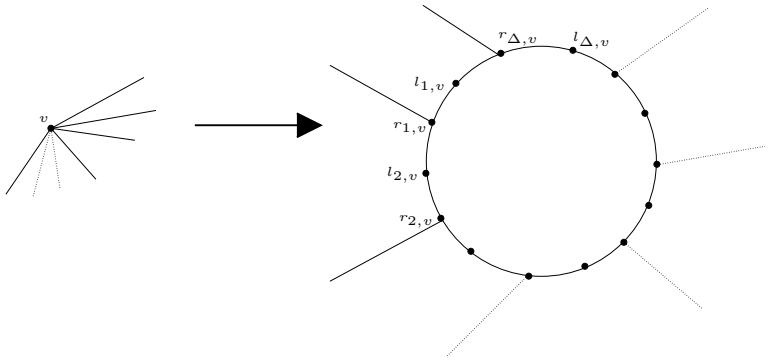


Figure 16.1. *The gadget $H(v)$*

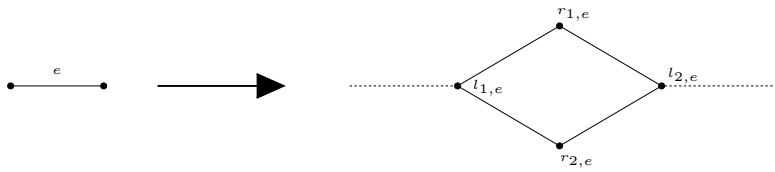


Figure 16.2. *The gadget $H(e)$*

• For each vertex $v \in V$, we construct a gadget $H(v)$ which is a cycle of length 2Δ. Thus, it is a bipartite graph where the left set is $L_v = \{l_{1,v}, \ldots, l_{\Delta,v}\}$ and the right set is $R_v = \{r_{1,v}, \ldots, r_{\Delta,v}\}$. The weights are $w(l_{i,v}) = 1$ and $w(r_{i,v}) = n\Delta(\frac{2+n\Delta}{2})$ for $i \in [\Delta]$.

The gadget $H(v)$ is illustrated in Figure 16.1.

• For each edge $e \in E$, we construct a gadget $H(e)$ which is a cycle of length 4. Thus, it is a bipartite graph where the left set is $L_e = \{l_{1,e}, l_{2,e}\}$ and the right set is $R_e = \{r_{1,e}, r_{2,e}\}$. The weights are $w(l_{i,e}) = \frac{(n\Delta)}{2}$ and $w(r_{i,e}) = \frac{(n\Delta)^2}{2}(\frac{2+n\Delta}{2})$ for $i = 1, 2$.

The gadget $H(e)$ is illustrated in Figure 16.2.

• We interconnect these gadgets by iteratively applying the following procedure.

For each edge $e = [u, v] \in E$, we add two edges $[r_{i,u}, l_{1,e}]$ and $[l_{i,u}, r_{1,e}]$ where $l_{i,u}$ is a neighbor of $r_{i,u}$ in $H(u)$ between gadgets $H(u)$, $H(e)$ and two edges $[r_{j,v}, l_{2,e}]$ and $[l_{j,v}, r_{2,e}]$ where $l_{j,v}$ is a neighbor of $r_{j,v}$ in $H(v)$ between gadgets $H(v)$, $H(e)$ such that the vertices $r_{i,u}$, $l_{i,u}$, $r_{j,v}$ and $l_{j,v}$ have degree 3.

It is clear that G' is bipartite and the weights are polynomially bounded. Moreover, since G is a Δ-regular graph, we conclude that G' is 3-regular.

We claim that there is a clique V^* of G with size of at least k if and only if the value of $\text{EWIS}_\alpha(G', w, b)$ is *yes*, where:

$$b = k\Delta + n\Delta\frac{k(k-1)}{2} + n\Delta(\frac{2+n\Delta}{2})\left((n-k)\Delta + (\frac{n\Delta}{2} - \frac{k(k-1)}{2})n\Delta\right)$$

Let I be a maximum independent set of G' with $w(I) = b$. Since G' is cubic and bipartite, G' has a perfect matching (for instance, take a perfect matching in each gadget $H(v)$ and $H(e)$) and we conclude that $\alpha(G) = |I| = |R| = |L|$. This implies in particular that for any vertex $v \in V$, either L_v or R_v is a subset of I. Moreover, the same property holds for any $e \in E$ (i.e., either L_e or R_e is a subset of I). By construction of the weights, the quantity $k\Delta$ of b must come from vertices $l_{i,v}$, $r_{i,v}$ or $l_{i,e}$. Since $k < n$, this quantity cannot come from $r_{i,v}$. Moreover, since $l_{i,e} \in I$ if and only if $L_e \subseteq I$, the contribution of L_e in I is $n\Delta$. In this case, the contribution of $k\Delta$ must come from $l_{i,v}$. Thus, we obtain:

$$|I \cap L_V| = k\Delta\,, \qquad |I \cap R_V| = (n-k)\Delta \tag{16.1}$$

where $L_V = \cup_{v\in V} L_v$ and $R_V = \cup_{v\in V} R_v$. Thus, using [16.1] we must obtain:

$$\begin{aligned} w\left(I \cap (L_E \cup R_E)\right) &= n\Delta\frac{k(k-1)}{2} \\ &\quad + n\Delta(\frac{2+n\Delta}{2})(\frac{n\Delta}{2} - \frac{k(k-1)}{2})n\Delta \end{aligned} \tag{16.2}$$

where $L_E = \cup_{e\in E} L_e$ and $R_E = \cup_{e\in E} R_e$. Now, we prove that there are exactly $k(\frac{k-1}{2})$ gadgets $H(e)$ with $L_e \subseteq I$. Assume the converse; then, $|I \cap L_E| = k(k-1) - 2p$ and $|I \cap R_E| = n\Delta - k(k-1) + 2p$ for some $p \neq 0$ (p can be negative). Combining these equalities with equality [16.2], we deduce that $p = 0$, which is a contradiction.

Thus, if we set $V^* = \{v \in V : L_v \subseteq I\}$, we deduce from above that $|V^*| = k$ and we will necessarily obtain that V^* is a clique of G.

Conversely, let V' be a clique of G with $|V'| \geqslant k$ and consider a subclique $V^* \subseteq V'$ of size exactly k. We set $S = S_L \cup S_R$ with $S_L = \cup_{v\in V^*} L_v \cup_{e\in E(V^*)} L_e$ and $S_R = \cup_{v\in V\setminus V^*} R_v \cup_{e\in E\setminus E(V^*)} R_e$. We can easily verify that $w(I) = b$ and that I is a maximum independent set of G'. Indeed, let us assume the converse; thus, there exist $r_{i,v} \in I$ (and thus $R_v \subseteq I$), $l_{j,e} \in I$ (with $j = 1, 2$) and $[r_{i,v}, l_{j,e}] \in E'$. By construction of I, we deduce that $e = [u, v] \in E(V^*)$ and then $L_v \subseteq I$, which is a contradiction. The proof is complete. ∎

As corollary of Theorem 16.1, we can derive that the biclique problem remains **NP**-complete when the minimum degree of $G = (L, R; E)$ is $n - 3$ where $|L| = |R| = n$. In this case, we replace any gadget $H(e)$ of Theorem 16.1 by a cycle of length $2n\Delta$ and we delete edges $[l_{i,u}, r_{1,e}]$ and $[l_{j,v}, r_{2,e}]$.

We also note that Theorem 16.1 implies the strong **NP**-completeness of the EWIS_α problem for perfect graphs, which are a well-known class where the weighted independent set problem is solvable in polynomial time [GRÖ 84].

16.3.2. *A more general hardness result*

Let us now strengthen the main result of the previous section. To this end, we first introduce some notations. Let $\mathcal{F}$ be a set of graphs. We denote the class of graphs containing no induced subgraphs from the set $\mathcal{F}$ by $Free(\mathcal{F})$. Any graph in $Free(\mathcal{F})$ will be called $\mathcal{F}$-free. Our hardness results will be expressed in terms of a parameter related to the set of forbidden induced subgraphs $\mathcal{F}$.

Let C_i and H_i denote the cycle of length i and the graph in Figure 16.3, respectively.

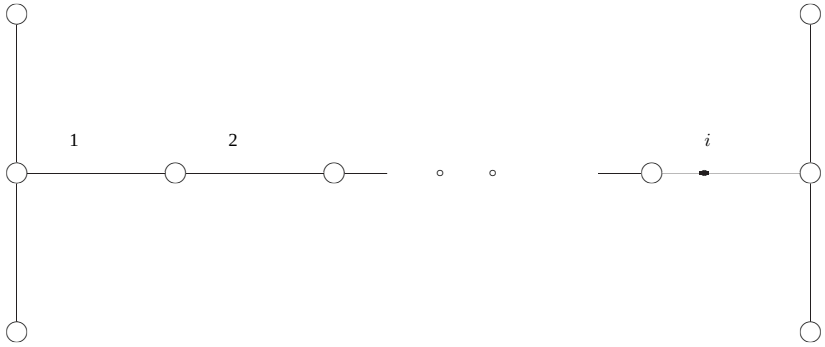


Figure 16.3. *Graph H_i*

We associate with every graph G a parameter $\kappa(G)$, which is the minimum value of $i \geqslant 1$ such that G contains an induced copy of either C_i or H_i. If G is an acyclic graph with no induced graphs of the form H_i, we let $\kappa(G) = \infty$. For a (possibly infinite) non-empty set of graphs $\mathcal{F}$, we define:

$$\kappa(\mathcal{F}) = \sup \{ \kappa(G) \ : \ G \in \mathcal{F} \}$$

Finally, for a set of graphs X, let X_3 denote the set of graphs of degree at most 3 in X.

With these definitions in mind, we can use the strong **NP**-completeness of the EWIS_α problem for bipartite graphs of degree at most 3 (which is an immediate corollary of Theorem 16.1), and the reduction typically used for the IS problem (see e.g. [MUR 92, POL 74]) to derive the following hardness result.

THEOREM 16.2.– *Let $\mathcal{G}$ be the class of $\mathcal{F}$-free bipartite graphs of maximum degree at most 3. If $\kappa(\mathcal{F}_3) < \infty$, then the EWIS_α problem is strongly **NP**-complete in the class $\mathcal{G}_3$.*

Proof. The problem is clearly in **NP**. We show completeness in two steps. First, for $k \geqslant 3$, let $\mathcal{S}_k$ be the class of all bipartite $(C_3, \ldots, C_k, H_1, \ldots, H_k)$-free graphs of vertex degree at most 3, and let us show that for any fixed k, the problem is strongly **NP**-complete for graphs in $\mathcal{S}_k$. Let (G, w, b) be an instance of the EWIS_α problem where G is a bipartite graph of maximum degree at most 3.

We can transform the graph G in polynomial time into a weighted graph G', as follows. Let $k' = \lceil \frac{k}{2} \rceil$. We replace each edge e of G by a path $P(e)$ on $2k' + 2$ vertices. Let $N = w(V) + 1$. We set the weights w' of the endpoints of $P(e)$ equal to the weights of the corresponding endpoints of e, while each internal vertex of $P(e)$ gets weight N. It is easy to verify that G' belongs to $\mathcal{S}_k$.

We claim that the value of $\text{EWIS}_\alpha(G, w, b)$ is *yes* if and only if the value of $\text{EWIS}_\alpha(G', w', b + mk'N)$ is *yes*, where $m = |E(G)|$.

One direction is immediate, as each maximum independent set of G can be extended to a maximum independent set of G' by simply adding k' internal vertices of each newly added path. By doing so, the weight increases by $mk'N$.

Suppose now that the value of $\text{EWIS}_\alpha(G', w', b + mk'N)$ is *yes*. Let I' be a maximum independent set of G' of weight $b + mk'N$. Since I' is independent, it can contain at most k' internal vertices of each newly added path. Therefore, for each $e \in E(G)$, the set I' must contain *exactly* k' internal vertices of $P(e)$, otherwise its weight would be at most $w(V) + (mk' - 1)N$, thus contradicting our choice of N.

Let I denote the set obtained from I' by deleting the internal vertices of newly added paths. Then, I is an independent set of G. Indeed, if $e = [u, v] \in E(G)$ for some $u, v \in I$, then I' would contain at most $k' - 1$ internal vertices of $P(e)$, thus contradicting the above observation. Also, it is easy to see that I is a maximum independent set of G. Finally, as the weight of I is exactly b, we conclude that the value of $\text{EWIS}_\alpha(G, w, b)$ is *yes*.

This shows that the EWIS_α problem is strongly **NP**-complete in the class $\mathcal{S}_k$. To prove strong **NP**-completeness of the problem in the class $\mathcal{G}_3$, we now show that the class $\mathcal{G}_3$ contains all graphs in $\mathcal{S}_k$, for $k := \max\{3, \kappa(\mathcal{F})\}$. Let G be a graph

from $\mathcal{S}_k$. Assume that G does not belong to $\mathcal{G}_3$. Then G contains a graph $A \in \mathcal{F}_3$ as an induced subgraph. From the choice of G we know that A belongs to $\mathcal{S}_k$, but then $k < \kappa(A) \leqslant \kappa(\mathcal{F}_3) \leqslant k$, which is a contradiction. Therefore, $G \in \mathcal{G}_3$ and the theorem is proved. ∎

16.4. Polynomial results

In this section, we present pseudo-polynomial solutions to the exact weighted independent set problem, when the input graphs are restricted to particular classes.[4] The algorithms resemble those for the WIS problem in respective graph classes, and are based either on a dynamic programming approach (section 16.4.1), or on the modular decomposition (section 16.4.2).

First, we observe that when developing polynomial-time solutions to the EWIS problem, we may restrict our attention to connected graphs.

LEMMA 16.3.– *Let (G, w, b) be an instance of the* EWIS *problem and let $C_1, \ldots, C_r$ be the connected components of G. Suppose that for each $i \in [r]$, the set of solutions $(\text{EWIS}(C_i, w, k) : k \in [b])$ for C_i is given. Then, we can calculate the set of solutions $(\text{EWIS}(G, w, k) : k \in [b])$ for G in time $O(rb^2)$.*

In order to show Lemma 16.3, we consider the following generalization of the subset sum problem.

GENERALIZED SUBSET SUM (GSS)
Instance: Non-empty sets of positive integers $A_1, \ldots, A_n$ and a positive integer b.
Question: Is there a non-empty subset J of $[n]$ and a mapping $a : J \to \cup_{j \in J} A_j$ such that $a(j) \in A_j$ for all $j \in J$, and $\sum_{j \in J} a(j) = b$?

By generalizing the dynamic programming solution to the subset sum problem, it is easy to show the following.

LEMMA 16.4.– *GSS can be solved in time $O(nb^2)$ by dynamic programming.*

In fact, in the stated time, not only we can verify if there is a $J \subseteq [n]$ and a mapping a as above such that $\sum_{j \in J} a(j) = b$ for the given b, but we can answer this question for *all* values $b' \in [b]$.

Proof. Let B denote the set of *all* values $b' \in [b]$ such that there is a non-empty subset S of $[n]$ and a mapping $a : S \to \cup_{i \in S} A_i$ such that $a(i) \in A_i$ for all $i \in S$, and $\sum_{i \in S} a(i) = b'$.

4. For a comprehensive survey on graph classes, see [BRA 99].

Let us show by induction on n that we can generate the set B in time $O(nb^2)$. The statement is trivial for $n = 1$.

Suppose now that $n > 1$. Let $I = (A_1, \ldots, A_n; b)$ be an instance of the GSS problem. Let B' be the inductively constructed set of all possible values of $b' \in [b]$ such that the solution to the GSS problem on the instance $(A_1, \ldots, A_{n-1}; b')$ is *yes*. By induction, the set B' was constructed in time $O((n-1)b^2)$.

Let $\beta \in [b]$. Then, β will belong to B, i.e., the solution to the GSS problem, given $(A_1, \ldots, A_n; \beta)$, will be *yes*, if and only if either $\beta \in B'$, or we can write β as $\beta = b' + a_n$ for some $b' \in B'$ and $a_n \in A_n$. In other words, $B = B' \cup B''$, where B'' denotes the set of all such sums: $B'' = \{b' + a_n : b' \in B', a_n \in A_n, b' + a_n \leqslant b\}$.

The set B'' can be constructed in time $O(b^2)$. Adding this time complexity to the time $O((n-1)b^2)$ needed to construct B' proves the above statement and hence the lemma. ∎

Lemma 16.3 now follows immediately.

Proof. [Lemma 16.3] It is enough to observe that for every $k \in [b]$, the value of EWIS(G, w, k) is *yes* if and only if the solution to the GSS problem on the instance $(A_1, \ldots, A_r; k)$ is *yes*, where A_i denotes the set of all values $k' \in [b]$ such that EWIS(C_i, w, k') is *yes*. ∎

16.4.1. *Dynamic programming solutions*

We can summarize the results of this section in the following theorem.

THEOREM 16.3.– *The exact weighted independent set and the exact weighted maximum independent set problems admit pseudo-polynomial-time solutions in each of the following graph classes: mK_2-free graphs, interval graphs and their generalizations k-thin graphs, circle graphs, chordal graphs, AT-free graphs, $(claw, net)$-free graphs, distance-hereditary graphs, graphs of treewidth at most k, and graphs of clique-width at most k.*

The rest of this section is devoted to proving this result. According to part (ii) of Lemma 16.1, it suffices to develop pseudo-polynomial solutions for the EWIS problem. Most of the algorithms resemble those for the WIS problem and exploit the special structure of graphs in the classes.

16.4.1.1. *mK_2-free graphs*

Our first example deals with graphs with no large induced matchings.

Recall that K_2 denotes the graph consisting of two adjacent vertices. The disjoint union of m copies of K_2 is denoted by mK_2. Thus, graphs whose largest induced matching consists of less than m edges are precisely the mK_2-free graphs.

THEOREM 16.4.– *For every positive integer m, the* EWIS *problem admits a pseudo-polynomial algorithm for mK_2-free graphs.*

Proof. An mK_2-free graph contains only polynomially many maximal independent sets (see [ALE 91, BAL 89, PRI 95]). Tsukiyama *et al.* describe in [TSU 77] an algorithm that generates all the maximal independent sets of a graph with polynomial delay. This implies that we can enumerate all maximal independent sets $I_1, \ldots, I_N$ of a given mK_2-free graph G in polynomial time.

For each vertex x of G and for each $k \in [b]$, the value of EWIS$(G[\{x\}], w, k)$ is *yes* if and only if $k = w(x)$. Therefore, we can apply Corollary 16.3 to each maximal independent set I of G in order to calculate the set of solutions to EWIS$(G[I], w, k)$ for all $k \in [b]$ in pseudo-polynomial time.

As each independent set of G is contained in some maximal independent set I_i of G, the value of EWIS(G, w, k) is *yes* if and only if there is an $i \in [N]$ such that the value of EWIS(I_i, w, k) is *yes*. This shows that the EWIS problem can be solved in pseudo-polynomial time for mK_2-free graphs. ∎

16.4.1.2. *Interval graphs*

Interval graphs are one of the most natural and well-understood classes of intersection graphs. They are intersection graphs of intervals on the real line, and many optimization problems can be solved by dynamic programming on these graphs.

Formally, given a collection $\mathcal{I}$ of intervals on the real line, its *intersection graph* $G(\mathcal{I})$ is defined by $V(G(\mathcal{I})) = \mathcal{I}$ and there is an edge connecting two intervals if and only if their intersection is nonempty. The collection $\mathcal{I}$ is said to be an *interval model* of $G(\mathcal{I})$. Finally, a graph G is said to be an *interval graph* if it admits an interval model, i.e., if there is a collection $\mathcal{I}$ of intervals on the real line such that $G = G(\mathcal{I})$.

A representation of interval graphs that is particularly suitable for the EWIS problem is the following. It has been shown by Ramalingam and Pandu Rangan [RAM 88] that a graph $G = (V, E)$ is interval if and only if it admits a vertex ordering $(v_1, \ldots, v_n)$ such that for all triple (r, s, t) with $1 \leqslant r < s < t \leqslant n$, the following implication is true:

$$\text{if } [v_r, v_t] \in E \text{ then } [v_s, v_t] \in E$$

Moreover, such an ordering of an interval graph can be found in time $O(n + m)$. Based on this ordering, we can prove the following statement.

THEOREM 16.5.– *The* EWIS *problem admits an* $O(bn + m)$ *algorithm for interval graphs.*

Proof. Let $(v_1, \ldots, v_n)$ be a vertex ordering such that $[v_s, v_t] \in E$, whenever $[v_r, v_t] \in E$, for all triples (r, s, t) with $1 \leqslant r < s < t \leqslant n$.

For every $i \in [n]$, let G_i denote the subgraph of G induced by $\{v_1, \ldots, v_i\}$ (also, let G_0 be the empty graph). Then, for every $i \in [n]$, either there is a $j = j(i)$ such that $N_{G_i}(v_i) = \{j, j+1, \ldots, i-1\}$, or $N_{G_i}(v_i) = \{\,\}$ (in which case let us define $j(i) = i$). Now, if I is an independent set of G_i, then either $v_i \in I$ (in which case $I \backslash \{v_i\}$ is an independent set of $G_{j(i)-1}$), or $v_i \notin I$ (in which case I is an independent set of G_{i-1}). This observation is the key to the following simple $O(bn + m)$ dynamic programming solution to the EWIS problem on interval graphs.

Step 1. Find a vertex ordering $(v_1, \ldots, v_n)$ as above.
Step 2. Set EWIS(G_0, w, k) to *no* for all $k \in [b]$.
Step 3. For $i = 1, \ldots, n$, do the following:
3.1. Find $j \in [i]$ such that $N_{G_i}(v_i) = \{j, j+1, \ldots, i-1\}$.
3.2. For $k \in [b]$, do the following:
If $k = w(v_i)$, set EWIS(G_i, w, k) to *yes*.
If $k < w(v_i)$, set EWIS(G_i, w, k) to EWIS(G_{i-1}, w, k).
If $k > w(v_i)$, set EWIS(G_i, w, k) to *yes* if at least one of the solutions to EWIS$(G_{j(i)-1}, w, k - w(v_i))$ and EWIS(G_{i-1}, w, k) is *yes*, and to *no* otherwise.

Step 4. Output the value of EWIS(G_n, w, b). ∎

16.4.1.3. *k-thin graphs*

The property used in the above characterization of interval graphs has been generalized by Mannino *et al.* in [MAN 07], where they define the class of k-thin graphs. A graph $G = (V, E)$ is said to be *k-thin* if there is an ordering $(v_1, \ldots, v_n)$ of V and a partition of V into k classes such that, for each triple (r, s, t) with $1 \leqslant r < s < t \leqslant n$, if v_r, v_s belong to the same class and $[v_r, v_t] \in E$, then $[v_s, v_t] \in E$.

Let us mention at this point that finding a feasible frequency assignment of a given cost can be modeled as the EWIS problem on a k-thin graph, where the parameter k depends on the input to the frequency assignment problem. For further details, see [MAN 07].

Based on the same idea as for interval graphs, a dynamic programming solution for k-thin graphs can be obtained, if we are given an ordering and a partition of the vertex set.

THEOREM 16.6.– *Suppose that for a k-thin graph $G = (V, E)$, $k \geqslant 2$, an ordering $(v_1, \ldots, v_n)$ of V and a partition of V into k classes are given such that, for each*

triple (r, s, t) *with* $1 \leqslant r < s < t \leqslant n$*, if* v_r*,* v_s *belong to the same class and* $[v_r, v_t] \in E$*, then* $[v_s, v_t] \in E$*. Then, the* EWIS *problem admits an* $O(bn^k)$ *algorithm for* G*.*

Proof. The proof is a straightforward extension of the proof of Theorem 16.5. Let $V_1, \ldots, V_k$ be the classes of the partition. Instead of the graphs G_i, which are induced by the first i vertices, we now consider all graphs $G(i_1, \ldots, i_k)$, which are induced by the "first" i_r vertices of each class (according to the ordering on V restricted to the class), for all $r \in \{1, \ldots, k\}$ and for all $O(n^k)$ choices of such k-tuples $(i_1, \ldots, i_k) \in [|V_1|] \times \ldots \times [|V_k|]$. ∎

16.4.1.4. *Circle graphs*

Besides intervals on the real line, chords on a circle provide another popular intersection model. The intersection graphs of chords on a circle are called *circle graphs*. In this section, we will present an $O(b^2n^2)$ dynamic-programming algorithm for the EWIS problem in circle graphs.

Our algorithm for the EWIS problem on circle graphs is based on the dynamic programming solution for the IS problem, which is developed by Supowit in [SUP 87].

THEOREM 16.7.– *The* EWIS *problem admits an* $O(b^2n^2)$ *algorithm for circle graphs.*

Proof. Consider a finite set of N chords on a circle. We may assume without loss of generality that no two chords share an endpoint. Number the endpoints of the chords from 1 to $2N$ in the order as they appear as we move clockwise around the circle (from an arbitrary but fixed starting point).

The idea is simple. For $1 \leqslant i < j \leqslant 2N$, let $G(i, j)$ denote the subgraph of G induced by chords whose both endpoints belong to the set $\{i, i+1, \ldots, j\}$. Obviously $G = G(1, 2N)$.

Let $1 \leqslant i < j \leqslant 2N$. If $j = i + 1$, then the value of EWIS$(G(i, j), w, k)$ is *yes* if and only if either $k = 0$, or $(i, i + 1)$ is a chord and $k = w((i, i + 1))$.

Otherwise, let r be the other endpoint of the chord whose one endpoint is j. If $r < i$ or $r > j$, then no independent set of the graph $G(i, j)$ contains the chord (r, j), so the value of EWIS$(G(i, j), w, k)$ equals the value of EWIS$(G(i, j - 1), w, k)$. Suppose now that $i \leqslant r \leqslant j-1$ and let I be an independent set of $G(i, j)$. The set I may or may not contain the chord (r, j). If I does not contain (r, j), then I is an independent set of of $G(i, j - 1)$ as well. If I contains (r, j), then no other chord in I can intersect the chord (r, j). In particular, this implies that I is of the form $I = \{(r, j)\} \cup I_1 \cup I_2$ where I_1 is an independent set of $G(i, r - 1)$ and I_2 is an independent set of $G(r + 1, j - 1)$.

Therefore, the value of EWIS$(G(i,j),w,k)$ is *yes* if and only if either EWIS$(G(i,j-1),w,k)$ is *yes*, or EWIS(G',w,k) is *yes*, where G' is the graph whose connected components are $G[\{(r,j)\}]$, $G(i,r-1)$ and $G(r+1,j-1)$. Assuming that the solutions for $G(i,r-1)$ and $G(r+1,j-1)$ have already been obtained recursively, we can apply Corollary 16.3 in this case.

The above discussion implies an obvious $O(b^2n^2)$ algorithm that correctly solves the problem. ∎

16.4.1.5. *Chordal graphs*

Chordal (or triangulated) graphs are graphs in which every cycle of length at least four has a chord. They strictly generalize interval graphs and provide another class where the WIS problem is polynomially solvable. Unfortunately for our purpose, the usual approaches for the WIS problem in chordal graphs ([FRA 76, TAR 85]) heavily rely on the maximization nature of the problem and generally do not preserve the overall structure of independent sets. As such, they do not seem to be directly extendable to the exact version of the problem. Instead, we develop a pseudo-polynomial time solution to the EWIS problem in chordal graphs by using one of the many characterizations of chordal graphs: their *clique-tree representation*.

THEOREM 16.8.– *The* EWIS *problem admits an* $O(b^2n(n+m))$ *algorithm for chordal graphs.*

Proof. Given a chordal graph G, we first calculate a *clique tree* of G. This can be done in time $O(n+m)$ [HSU 99]. A clique tree of a chordal graph G is a tree T whose nodes are the maximal cliques of G, such that for every vertex v of G, the subgraph T_v of T induced by the maximal cliques containing v is a tree. Furthermore, we fix an arbitrary node K_r in the clique tree in order to obtain a *rooted clique tree*. For a maximal clique K, we denote by $G(K)$ the subgraph of G induced by the vertices of K and all vertices contained in some descendant of K in T.

The algorithm is based on a set of identities developed by Okamoto, Uno and Uehara in [OKA 05], where a clique tree representation was used to develop linear-time algorithms to count independent sets in a chordal graph. Let $\mathcal{IS}(G)$ be the family of independent sets in G. For a vertex v, let $\mathcal{IS}(G,v)$ be the family of independent sets in G that contain v. For a vertex set U, let $\overline{\mathcal{IS}}(G,U)$ be the family of independent sets in G that contain no vertex of U. Consider a maximal clique K of G, and let $K_1,\ldots,K_l$ be the children of K in T (if K is a leaf of the clique tree, we set $l=0$). Then, as shown in [OKA 05], for every distinct $i,j\in[l]$, the sets $V(G(K_i))\backslash K$ and $V(G(K_j))\backslash K$ are disjoint. Moreover, if $\sqcup$ denotes the disjoint union, the following

relations hold:

$$\begin{aligned}
\mathcal{IS}(G(K)) &= \overline{\mathcal{IS}}(G(K),K) \sqcup \bigsqcup_{v\in K} \mathcal{IS}(G(K),v) \\
\mathcal{IS}(G(K),v) &= \Big\{ I\cup\{v\} \;\Big|\; I=\bigcup_{i=1}^{l} I_i, \\
&\qquad I_i \in \begin{cases} \mathcal{IS}(G(K_i),v), & \text{if } v\in K_i; \\ \overline{\mathcal{IS}}(G(K_i),K\cap K_i), & \text{otherwise.} \end{cases} \\
\overline{\mathcal{IS}}(G(K),K) &= \Big\{ I \;\Big|\; I=\bigsqcup_{i=1}^{l} I_i,\ I_i\in\overline{\mathcal{IS}}(G(K_i),K\cap K_i)\Big\} \\
\\
\overline{\mathcal{IS}}(G(K_i),K\cap K_i) &= \overline{\mathcal{IS}}(G(K_i),K_i)) \\
&\quad \sqcup \bigsqcup_{u\in K_i\setminus K} \mathcal{IS}(G(K_i),u) \quad \text{for each } i\in[l]
\end{aligned}$$

We extend our usual Boolean predicate $\mathrm{EWIS}(H,w,k)$ to the following two: for a vertex v of a weighted graph (H,w) and an integer k, let $\mathrm{EWIS}(H,w,k,v)$ denote the Boolean predicate whose value is *yes* if and only if in H there is an independent set I of total weight k that contains v. Also, for a set of vertices U let $\overline{\mathrm{EWIS}}(H,w,k,U)$ take the value *yes* if and only if in H there is an independent set of total weight k that contains no vertex from U. Based on the above equations, we can develop the following recursive relations for EWIS:

$$\begin{aligned}
\mathrm{EWIS}(G(K),w,k) &= \overline{\mathrm{EWIS}}(G(K),w,k,K) \\
&\quad \vee \bigvee_{v\in K: w(v)\leqslant k} \mathrm{EWIS}(G(K),w,k,v)
\end{aligned} \qquad [16.3]$$

where $\vee$ denotes the usual *Boolean OR function* (with the obvious identification $yes \leftrightarrow 1$, $no \leftrightarrow 0$). That is, its value is *yes* if at least one of its arguments is *yes*, and *no* otherwise.

$$\mathrm{EWIS}(G(K),w,k,v) = \mathbf{GSS}(A_1,\ldots,A_l,k-w(v)) \qquad [16.4]$$

where $\mathbf{GSS}(A_1,\ldots,A_l,k)$ denotes the solution to the generalized subset sum problem on the input instance $(A_1,\ldots,A_l,k)$, where the sets A_i for $i\in[l]$ are given by:

$$A_i = \begin{cases} \{k'-w(v) \;:\; w(v)\leqslant k'\leqslant k,\ \mathrm{EWIS}(G(K_i),w,k',v)=yes\}, & \text{if } v\in K_i \\ \{k' \;:\; 1\leqslant k'\leqslant k,\ \overline{\mathrm{EWIS}}(G(K_i),w,k',K\cap K_i)=yes\}, & \text{otherwise} \end{cases}$$

Note that if $I_i\in\mathcal{IS}(G(K_i),v)$ and $I_j\in\mathcal{IS}(G(K_j),v)$ for some distinct indices $i,j\in[l]$, then we have $I_i\cap I_j=\{v\}$. Moreover, since this is the only possible nonempty intersection of two independent sets from $\bigcup_{i=1}^{l} I_i$ in the equation for

$\mathcal{IS}(G(K), v)$, it follows that the sum of the weights of the sets $I_i \backslash \{v\}$ (over all $i \in [l]$) equals the weight of $\left(\bigcup_{i=1}^{l} I_i\right) \backslash \{v\}$, thus justifying equation [16.4].

Similarly, we have:

$$\overline{\text{EWIS}}(G(K), w, k, K) = \mathbf{GSS}(A_1, \ldots, A_l, k) \qquad [16.5]$$

where, for each $i \in [l]$, the set A_i is given by:

$$A_i = \{k' : 1 \leqslant k' \leqslant k,\ \overline{\text{EWIS}}(G(K_i), w, k', K \cap K_i) = yes\}$$

and, finally, for each $i \in [l]$, we have:

$$\overline{\text{EWIS}}(G(K_i), w, k, K \cap K_i) = \overline{\text{EWIS}}(G(K_i, w, k, K_i)) \vee \bigvee_{u \in K_i \backslash K} \text{EWIS}(G(K_i), w, k, u) \qquad [16.6]$$

Given the above equations, it is now easy to develop a pseudo-polynomial dynamic programming algorithm. Having constructed a rooted tree T of G, we traverse it in a bottom-up manner. For a leaf K, we have:

$$\overline{\text{EWIS}}(G(K), w, k, K) = \begin{cases} yes, & \text{if } k = 0 \\ no, & \text{otherwise} \end{cases}$$

and

$$\text{EWIS}(G(K), w, k, v) = \begin{cases} yes, & \text{if } w(v) = k \\ no, & \text{otherwise} \end{cases}$$

For every other node K, we calculate the values of $\overline{\text{EWIS}}(G(K), w, k, K)$ and $\text{EWIS}(G(K), w, k, v)$ by referring to the recursive relations [16.6], [16.5] and [16.4] in this order. Finally, the value of $\text{EWIS}(G, w, k)$ equals the value of $\text{EWIS}(G(K_r), w, k)$, which can be calculated using equation [16.3].

The correctness of the procedure follows immediately from the above discussion. To justify the time complexity, observe that in a node K of the tree with children $K_1, \ldots, K_l$, the number of operations performed is $O(\sum_{i=1}^{l} |K_i| + lb^2 + |K|lb^2)$. Summing up over all the nodes of the clique tree, and using the fact that a chordal graph has at most n maximal cliques, which satisfy $\sum_{K \in V(T)} |K| = O(n + m)$ [OKA 05], the claimed complexity bound follows. ∎

16.4.1.6. *AT-free graphs*

Another generalization of interval graphs is given by the so-called *AT-free graphs*. Besides interval graphs, the family of AT-free graphs contains other well-known subclasses of perfect graphs, for instance, *permutation graphs* and their superclass, i.e., the class of *co-comparability graphs*.

A triple $\{x, y, z\}$ of pairwise non-adjacent vertices in a graph G is an *asteroidal triple* if for every two of these vertices there is a path between them avoiding the closed neighborhood of the third. Formally, x and y are in the same component of $G - N[z]$, x and z are in the same component of $G - N[y]$, and y and z are in the same component of $G - N[y]$[5]. A graph is called *AT-free* if it has no asteroidal triples.

Our dynamic programming algorithm that solves the EWIS problem for AT-free graphs is based on the dynamic programming solution to the WIS problem in AT-free graphs, developed by Broersma, Kloks, Kratsch and Müller in [BRO 99].

Let us start with a definition.

DEFINITION 16.1.– *Let x and y be two distinct non-adjacent vertices of an AT-free graph G. The interval $I(x, y)$ is the set of all vertices z of $V(G)\backslash\{x, y\}$ such that x and z are in one component of $G - N[y]$, and z and y are in one component of $G - N[x]$.*

Now, we recall some structural results from [BRO 99].

THEOREM 16.9.– *[BRO 99] Let $I = I(x, y)$ be a nonempty interval of an AT-free graph G, and let $s \in I$. Then there are components $C_1^s, \ldots, C_t^s$ of $G - N[s]$ such that the components of $I\backslash N[s]$ are precisely $I(x, s)$, $I(s, y)$ and $C_1^s, \ldots, C_t^s$.*

THEOREM 16.10.– *[BRO 99] Let G be an AT-free graph, let C be a component of $G - N[x]$, let $y \in C$ and let D be a component of the graph $C - N[y]$. Then $N[D] \cap (N[x]\backslash N[y]) = \{\}$ if and only if D is a component of $G - N[y]$.*

THEOREM 16.11.– *[BRO 99] Let G be an AT-free graph, let C be a component of $G - N[x]$, let $y \in C$, and let C' be the component of $G - N[y]$ that contains x. Let $B_1, \ldots, B_l$ denote the components of the graph $C - N[y]$ that are contained in C'. Then $I(x, y) = \cup_{i=1}^{l} B_i$.*

The following general lemma is obvious.

5. Recall that the closed neighborhood of x is defined as $N[x] = N(x) \cup \{x\}$.

LEMMA 16.5.– *Let (G, w) be a weighted graph. Then, the value of* EWIS(G, w, k) *is yes if and only if there is a vertex $x \in V(G)$ such that the value of* EWIS$(G - N(x), w, k)$ *is yes.*

Combining Lemma 16.5 with Theorems 16.10 and 16.11, we obtain the following result.

LEMMA 16.6.– *Let (G, w) be a weighted AT-free graph, $G = (V, E)$. Let $x \in V$ and let C be a component of $G - N[x]$. For a vertex y of C, let C_y denote the subgraph of G induced by $C - N(y)$. Then, the value of* EWIS(C, w, k) *is yes if and only if there is a vertex $y \in C$ such that the value of* EWIS(C_y, w, k) *is yes. Moreover, the connected components of such a C_y are precisely $\{y\}$, $I(x, y)$, and the components of $G - N[y]$ contained in C.*

Combining Lemma 16.5 with Theorem 16.9, we obtain the following result.

LEMMA 16.7.– *Let (G, w) be a weighted AT-free graph, $G = (V, E)$. Let $I = I(x, y)$ be an interval of G. If $I = \{\}$, then the value of* EWIS$(G[I], w, k)$ *is yes if and only if $k = 0$. Otherwise, let us denote by I_s the subgraph of G induced by $I - N(s)$, for all $s \in I$. Then, the value of* EWIS(I, w, k) *is yes if and only if there is a vertex $s \in I$ such that the value of* EWIS(I_s, w, k) *is yes. Moreover, the connected components of such an I_s are precisely $\{s\}$, $I(x, s)$, $I(s, y)$, and the components of $G - N[s]$ contained in I.*

THEOREM 16.12.– *The EWIS problem admits a pseudo-polynomial algorithm for AT-free graphs.*

Proof. It follows from the above discussion that the following pseudo-polynomial algorithm correctly solves the problem.

Step 1. For every $x \in V$, calculate the components of $G - N[x]$.
Step 2. For every pair of nonadjacent vertices $x, y \in V(G)$, calculate the interval $I(x, y)$.
Step 3. Sort all the components and intervals according to non-increasing number of vertices.
Step 4. In the order of Step 3, calculate the solutions to EWIS(C, w, k), for each component C (for all $k \in [w(C)]$) and the solutions to EWIS(I, w, k) for each interval I (for all $k \in [w(I)]$). To calculate the solutions to EWIS(C, w, k) for a component C, first calculate the solutions to EWIS$(C - N(y), w, k)$, for all $y \in C$, by applying Lemma 16.6 and Corollary 16.3. Similarly, to calculate the solutions to EWIS(I, w, k) for an interval I, first calculate the solutions to EWIS$(I - N(s), w, k)$, for all $s \in I$, by applying Lemma 16.7 and Corollary 16.3.
Step 5. Compute EWIS(G, w, b) using Lemma 16.5 and Corollary 16.3. ∎

A *claw* is the graph $K_{1,3}$. A *net* is the graph obtained from a triangle by attaching one pendant edge to each vertex. The following result is an immediate consequence of Theorem 16.12.

COROLLARY 16.2.– The EWIS problem admits a pseudo-polynomial algorithm for $(claw, net)$-free graphs.

Proof. In [BRA 03], it is shown that for every vertex v of a $(claw, net)$-free graph G, the non-neighborhood of v in G is AT-free. Thus, the problem amounts to solving $O(nb)$ subproblems in AT-free graphs, which can be done in pseudo-polynomial time by Theorem 16.12. ∎

16.4.1.7. *Distance-hereditary graphs*

Distance-hereditary graphs are graphs such that the distance between any two connected vertices is the same in every induced subgraph in which they remain connected.[6] Bandelt and Mulder provided in [BAN 86] a *pruning sequence* characterization of distance-hereditary graphs: whenever a graph contains a vertex of degree one, or a vertex with a twin (another vertex sharing the same neighbors), remove such a vertex. A graph is distance-hereditary if and only if it the application of such vertex removals results in a single-vertex graph.

More formally, a *pruning sequence* of a distance-hereditary graph G is a sequence of the form $\sigma = (x_1R_1y_1, x_2R_2y_2, \ldots, x_{n-1}R_{n-1}y_{n-1})$ where $(x_1, \ldots, x_n)$ is a total ordering of $V(G)$ such that for all $i \in [n-1]$, the following holds:

– $R_i \in \{P, T, F\}$.

– If we denote by G_i the subgraph of G induced by $\{x_i, \ldots, x_n\}$, then:

- If $R_i = P$ then x_i is a *pendant vertex*, i.e., a vertex of degree one in the graph G_i, with $N_{G_i}(x_i) = \{y_i\}$.

- If $R_i = T$ then x_i and y_i are *true twins* in G_i, i.e., $N_{G_i}[x_i] = N_{G_i}[y_i]$.

- If $R_i = F$ then x_i and y_i are *false twins* in G_i, i.e., $N_{G_i}(x_i) = N_{G_i}(y_i)$.

A pruning sequence of a distance-hereditary graph can be calculated in linear time [DAM 01]. Our algorithm for the EWIS problem on distance-hereditary graphs is based on the dynamic programming solution for the WIS problem, which was developed by Cogis and Thierry in [COG 05].

We note that that every distance-hereditary graph is a circle graph. However, the algorithm developed here for distance-hereditary graphs is faster than that for general circle graphs given by Theorem 16.7.

6. The distance between two vertices u and v in a connected graph G is the length (i.e., the number of edges) of a shortest path connecting them.

THEOREM 16.13.– *The EWIS problem admits an $O(b^2n+m)$ algorithm for distance-hereditary graphs.*

Proof. We first define an auxiliary problem:

P1(G, b, p, q)
Instance: a graph G, a positive integer b, and two functions:

$$p, q : V \times \{0, 1, \ldots, b\} \to \{0, 1\}$$

Question: is there an independent set I of G, and a mapping $w : V \to \{0, 1, \ldots, b\}$ such that the following holds:

- $\sum_{x \in V} w(x) = b$,
- $p(x, w(x)) = 1$ whenever $x \in I$,
- $q(x, w(x)) = 1$ whenever $x \notin I$?

Let us show that the EWIS problem is reducible to the P1 problem in $O(nb)$ time. Let (G, w, b) be an instance of the EWIS problem. Define $p, q, : V(G) \times \{0, 1, \ldots, b\} \to \{0, 1\}$ as follows. For each $x \in V(G)$ and each $k \in \{0, 1, \ldots, b\}$, let:

$$p(x, k) = \begin{cases} 1, & \text{if } k = w(x) \\ 0, & \text{otherwise} \end{cases}$$

and:

$$q(x, k) = \begin{cases} 1, & \text{if } k = 0 \\ 0, & \text{otherwise} \end{cases}$$

Then, it is easy to see that the value of EWIS(G, w, b) is *yes* if and only if P1(G, b, p, q) is *yes*.

In what follows, we present an $O(b^2n+m)$ to solve the problem P1 on an instance (G, b, p, q), if G is a distance-hereditary graph. This will in turn imply the statement of the theorem. For two functions $f, g : \{0, 1, \ldots, N\} \to \{0, 1\}$, we denote their *convolution* $f * g$ as the function $f * g : \{0, 1, \ldots, N\} \to \{0, 1\}$, which is given by the following rule: for every $k \in \{0, 1, \ldots, N\}$, we have:

$$(f*g)(k) = \begin{cases} 1, & \text{if there is a } k' \in \{0, 1, \ldots, k\} \text{ such that } p(k') = q(k - k') = 1 \\ 0, & \text{otherwise} \end{cases}$$

Procedure P1-DH.
Input: A distance-hereditary graph G, a positive integer b, and two functions $p, q : V \times \{0, 1, \ldots, b\} \to \{0, 1\}$.
Output: The answer to the question in P1(G, b, p, q).

Step 1. Calculate the pruning sequence $\sigma = (x_1R_1y_1, x_2R_2y_2, \ldots, x_{n-1}R_{n-1}y_{n-1})$ for G. Associate each vertex $x \in V(G)$ with a pair of functions $p^x, q^x : \{0, 1, \ldots, b\} \to \{0, 1\}$, given by $p^x(\cdot) = p(x, \cdot)$ and $q^x(\cdot) = q(x, \cdot)$.

Step 2. Check if the pruning sequence is empty. If yes, there is only one vertex x left. If $\max\{p^x(b), q^x(b)\} = 1$, then output *yes*. Otherwise, output no.
Otherwise, let xRy be the head of the pruning sequence. Update the pruning sequence by removing xRy from it. Update p^y and q^y as follows:

– If $R = P$ then let:

$$\begin{array}{rcl} p^y(k) & \leftarrow & (p^y * q^x)(k) \\ q^y(k) & \leftarrow & \max\{(p^x * q^y)(k), (q^x * q^y)(k)\} \end{array}$$

for each $k \in \{0, 1, \ldots, b\}$.

– If $R = T$ then let:

$$\begin{array}{rcl} p^y(k) & \leftarrow & \max\{(p^y * q^x)(k), (p^x * q^y)(k)\} \\ q^y(k) & \leftarrow & (q^x * q^y)(k) \end{array}$$

for each $k \in \{0, 1, \ldots, b\}$.

– If $R = F$ then let:

$$\begin{array}{rcl} p^y(k) & \leftarrow & \max\{(p^x * q^y)(k), (p^x * p^y)(k), (q^x * p^y)(k)\} \\ q^y(k) & \leftarrow & (q^x * q^y)(k) \end{array}$$

for each $k \in \{0, 1, \ldots, b\}$.

Go to *Step 2*.

The correctness of the algorithm can be easily proved by induction on n. We leave this routine proof to the reader. Clearly, the algorithm can be implemented so that it runs in time $O(b^2n + m)$. ∎

16.4.1.8. *Graphs of treewidth at most k*

Graphs of treewidth at most k, also known as *partial k-trees*, generalize trees and are very important from an algorithmic viewpoint: many graph problems that are **NP**-hard for general graphs are solvable in linear time when restricted to graphs of treewidth at most k [ARN 89]. It is easy to see that on trees, the EWIS problem admits a simple dynamic programming solution. With some care, the same approach can be generalized to graphs of bounded treewidth.

Let us first recall the definition of treewidth, and some related basic facts.

A *tree-decomposition* of a graph $G = (V, E)$ is a tree $T = (\mathcal{I}, F)$ where each vertex $i \in \mathcal{I}$ has a label $X_i \subseteq V$ such that:

(i) $\cup_{i\in\mathcal{I}} X_i = V$,

(ii) For every edge $[u, v] \in E$, there exists an $i \in \mathcal{I}$ such that $u, v \in X_i$,

(iii) For every $v \in V$, the vertices of T whose label contains v induce a connected subtree of T.

The *width* of such a decomposition is $\max_{i\in\mathcal{I}} |X_i|$. The *treewidth* of a graph G is the minimum k such that G has a tree-decomposition of width k.

Any graph of treewidth k has a tree-decomposition $T = (\mathcal{I}, F)$ such that:

– all the sets X_i in the decomposition have size $k + 1$,

– if $[i, j] \in F$ then $|X_i \cap X_j| = k$.

Such a decomposition can be obtained in linear time from any tree-decomposition of G of width k. Also, given a graph of treewidth k, a tree-decomposition of G of width k can be obtained in linear time [BOD 96].

THEOREM 16.14.– *For every fixed k, the EWIS problem admits an $O(nb^2)$ algorithm for graphs of treewidth at most k.*

Proof. Let $G = (V, E)$ be a weighted graph of treewidth k. First, we construct a special decomposition $T = (\mathcal{I}, F)$ of width k as mentioned above. We further refine this composition by subdividing each edge $[i, j]$ of T and labeling the new node with the set $X_i \cap X_j$. Now, every edge connects a set of size k with one of its supersets of size $k + 1$.

We root the decomposition tree at an arbitrary node r. The new, rooted decomposition tree has the following properties:

– any node corresponding to a set of size k has exactly one child,

– for a node corresponding to a set X_i of size k, its child corresponds to a superset of X_i of size $k + 1$,

– every child of a node corresponding to a set X_i of size $k + 1$ corresponds to a subset of X_i of size k.

For a node i of the decomposition tree, let Y_i denote the set of all vertices of G which appear either in X_i, or in any of the sets corresponding to the descendants of i.

For any node i, any subset Z of X_i, and any integer $p \in \{0, 1, \ldots, b\}$, define the $\{0, 1\}$-valued function:

$$ewis(i, Z, w) = \begin{cases} 1, & \text{if there is an independent set } I \text{ in } G[Y_i] \text{ of weight } p \\ & \text{with } I \cap X_i = Z \\ 0, & \text{otherwise} \end{cases}$$

Clearly, the value of EWIS(G, w, b) is *yes* if and only if there is a subset Z of the set X_r corresponding to the root r such that $ewis(r, Z, b) = 1$.

If X_i is a leaf of the decomposition tree, then it is easy to calculate the values $ewis(i, Z, p)$. Indeed, in this case $Y_i = X_i$, so we can set:

$$ewis(i, Z, p) = \begin{cases} 1 & \text{if } Z \text{ is an independent set in } G \text{ of weight } p \\ 0 & \text{otherwise} \end{cases}$$

For the internal vertices, we consider two cases.

Case 1. The size of X_i is k. This implies that i has only one child, for example j. The set X_j is a superset of X_i of size $k+1$, so $X_i = X_j \backslash \{v\}$ for some vertex v. Also, $Y_i = Y_j$, since X_i does not add any new vertices. We can calculate $ewis(i, Z, p)$ by the following formula:

$$ewis(i, Z, p) = ewis(j, Z, p) \vee ewis(j, Z \cup \{v\}, p)$$

where $\vee$ denotes the Boolean OR function.

Case 2. The size of X_i is $k+1$. Let $\{j_1, \ldots, j_t\}$ be the children of i in the decomposition tree. We would like to calculate $ewis(i, Z, p)$, where Z is a subset of X_i, and $w \in \{0, 1, \ldots, b\}$. If Z is not independent, then we set $ewis(i, Z, p)$ to 0.

From now on, assume that Z is independent. Recall that each of the sets X_{j_s}, for $s \in \{1, \ldots, t\}$, is a subset of X_i.

Let I be an independent set in $G[Y_i]$ with $I \cap X_i = Z$. Then $I \cap X_{j_s} = Z \cap X_{j_s}$. For $s \in [t]$, let us denote by I_s that part of the set I which belongs to Y_{j_s} but not to Z, i.e., $I_s = I \cap (Y_{j_s} \backslash Z)$. In particular, this implies that $I_s \cap X_i = \{\}$, and consequently $I_s \cap X_{j_s} = \{\}$.

Note that the set I equals to the disjoint union of the set Z and the sets $I_1, \ldots, I_t$. Therefore, if the weight of I is p, then:

$$p = w(I) = w(Z) + \sum_{s=1}^{t} w(I_s)$$

Thus:

$$\sum_{s=1}^{t} w(I_s) = p - w(Z)$$

which implies:

$$\sum_{s=1}^{t} w(I \cap Y_{j_s}) = \sum_{s=1}^{t} (w(I_s) + w(Z \cap X_{j_s})) = p - w(Z) + \sum_{s=1}^{t} w(Z \cap X_{j_s})$$

In particular, $ewis(i, Z, p)$ will take the value 1 if and only if there are non-negative integers $p_1, \ldots, p_t$ such that:

– (i) $w(Z \cap X_{j_s}) \leqslant p_s \leqslant p$ for all $s \in [t]$,
– (ii) $\sum_{s=1}^{t} p_s = p - w(Z) + \sum_{s=1}^{t} w(Z \cap X_{j_s})$, and
– (iii) $ewis(j_s, Z \cap X_{j_s}, p_s) = 1$ for all $s \in [t]$.

One direction is immediate: if $ewis(i, Z, p)$ takes value 1, then there is an I as above, and we can take $p_s = w(I \cap Y_{j_s})$, for $s \in [t]$.

On the other hand, the existence of such integers $p_1, \ldots, p_t$ implies that there are sets $I'_1, \ldots, I'_t$ such that, for all $s \in [t]$, I'_s is an independent set in $G[Y_{j_s}]$ of weight p_s with $I'_s \cap X_{j_s} = Z \cap X_{j_s}$.

We claim that the set $I = \cup_{s=1}^{t} I'_s \cup Z$ is an independent set in $G[Y_i]$ of weight p with $I \cap X_i = Z$. To see this, let us write $I_s = I'_s \cap (Y_{j_s} \setminus Z)$. Then, each I'_s equals the disjoint union of the sets I_s and $I'_s \cap X_{j_s} = Z \cap X_{j_s}$. Moreover, the set I equals the disjoint union of the set Z and the sets $I_1, \ldots, I_t$. Therefore:

$$\begin{aligned}
p &= \sum_{s=1}^{t} p_s + w(Z) - \sum_{s=1}^{t} w(Z \cap X_{j_s}) && [16.7]\\
&= w(Z) + \sum_{s=1}^{t} (w(I'_s) - w(Z \cap X_{j_s})) && [16.8]\\
&= w(Z) + \sum_{s=1}^{t} w(I_s) && [16.9]\\
&= w(I) && [16.10]
\end{aligned}$$

To see that I intersects X_i in Z, we just need to observe that:

$$I \cap X_i = (\cup_{s=1}^{t} I_s \cup Z) \cap X_i = \cup_{s=1}^{t} (I_s \cap X_i) \cup (Z \cap X_i) = Z \cap X_i = Z$$

as $I_s \cap X_i = \{\,\}$ for all $s \in [t]$.

Finally, we need to show that I is independent. By contradiction, suppose that there are vertices $u, v \in I$ such that $[u, v] \in E(G)$. As Z is independent by assumption, at most one of u and v is contained in Z. We may therefore assume without loss of generality that $u \in I_1$. As $I_1 \subseteq I'_1$ and I'_1 is independent, this implies that $v \notin I'_1$.

According to the tree decomposition properties, there is a set X_{i^*} such that $[u, v] \subseteq X_{i^*}$. Also, the set $S_u = \{j : u \in X_j\}$ forms a subtree of our decomposition tree.

Since u is not contained in Z and $I \cap X_i = Z$, the vertex u is not contained in X_i either. However, u is an element of I_1 and is therefore contained in Y_{j_1}.

These observations imply that the set S_u is contained in the subtree rooted at j_1. In particular, the node i^* is a descendant of j_1 in our decomposition tree. Next, it follows from $v \in I \backslash I'_1$ that v is also contained in some X_j such that j is a (not necessarily proper) descendant of i which is *not* contained in the subtree rooted at j_1. As $v \in X_{i^*}$ and as the set $S_v = \{j : v \in X_j\}$ also forms a connected subgraph, we conclude that $v \in X_{j_1}$. However, together with $I \cap X_{j_1} = Z \cap X_{j_1} = I'_1 \cap X_{j_1} \subseteq I'_1$, this leads to a contradicting $v \in I'_1$.

The existence of such p_i can be determined in time $O(tb^2)$ by dynamic programming. Indeed, finding such w_i is equivalent to the following restricted version of the GENERALIZED SUBSET SUM problem defined as follows

Instance: Nonempty sets of positive integers $A_1, \ldots, A_n$ and a positive integer b.
Question: Are there $a(i) \in A_i$ for all $i \in [n]$ such that $\sum_{i=1}^{n} a(i) = b$?

The correspondence is given by setting:

– $n = t$,

– $A_s = \{p_s : w(Z \cap X_{j_s}) \leqslant p_s \leqslant p, ewis(j_s, Z \cap X_{j_s}, p_s) = 1\}$ for all $s \in [t]$, and

– $b = p - w(Z) + \sum_{s=1}^{t} w(Z \cap X_{j_s})$.

A slight modification of the proof of Lemma 16.4 shows that this problem can be solved in time $O(tb^2)$.

Summing up over all internal nodes and taking into account that $|E(G)| = O(n)$ (when viewing k as a constant), the total complexity of solving the EWIS problem for graphs of treewidth k is $O(nb^2)$. ∎

Note that the same algorithm runs in pseudo-polynomial time whenever the treewidth of the input graph is of the order $O(\log n)$.

16.4.1.9. *Graphs of clique-width at most k*

The *clique-width* of a graph G is defined as the minimum number of labels needed to construct G, using the following four graph operations:

(i) Create a new vertex v with label i (denoted by $i(v)$).

(ii) Take the disjoint union of two labeled graphs G and H (denoted by $G \oplus H$).

(iii) Join by an edge each vertex with label i to each vertex with label j ($i \neq j$, denoted by $\eta_{i,j}$).

(iv) Rename label i to j (denoted by $\rho_{i \to j}$).

An expression built from the above four operations is called a *clique-width expression*. A clique-width expression using k labels is called a *k-expression*. Each k-expression t uniquely defines a labeled graph $lab(t)$, where the labels are integers $\{1, \ldots, k\}$ associated with the vertices and each vertex has exactly one label. We say that a k-expression t defines a graph G if G is equal to the graph obtained from the labeled graph $lab(t)$ after removing its labels. The clique-width of a graph G is equal to the minimum k such that there exists a k-expression defining G.

As shown by Corneil and Rotics in [COR 05], the clique-width of a graph of treewidth k is bounded above by $3 \cdot 2^{k-1}$. This implies that a class of graphs with uniformly bounded treewidth is also of bounded clique-width. The converse is generally not true, as the complete graphs show: for every $n \geqslant 2$, the clique-width of K_n is 2, while its treewidth is $n - 1$. In this sense, showing that a problem can be efficiently solved for graphs of bounded clique-width is more general than showing the same statement for graphs of bounded treewidth.

Many graph problems that are **NP**-hard for general graphs are solvable in linear time when restricted to graphs of clique-width at most k, if a k-expression is given as part of the input (see e.g. [COU 97])[7]. The EWIS problem is no exception.

THEOREM 16.15.– *For every fixed k, the EWIS problem admits an $O(b^2 l)$ algorithm for graphs of clique-width at most k, where l is the number of operations in a given k-expression for G.*

Proof. Suppose that the labels are integers $\{1, \ldots, k\} = [k]$. For every subset of labels $S \subseteq [k]$, let $\text{EWIS}(G, w, S, m)$ denote the answer to the following question: "is there an independent set of G with total weight m that contains exactly the labels from S?"

Given a k-expression t defining the input graph G, we can determine the value of $\text{EWIS}(G, w, b)$ by first calculating all the values for $\text{EWIS}(G, w, S, m)$, for every

7. If only a graph G of clique-width at most k is given, then an $O(2^{6k})$ expression defining G can be calculated in $O(n^3)$ time, as shown by Oum in [OUM 05].

subset of labels $S \subseteq [k]$, and every $m \in [b]$. It is easy to see that this can be performed in time $O(b^2 l)$ by the following dynamic programming algorithm.

If $|V| = 1$, then let $v \in V$. For all $S \subseteq [k]$, and for all $m \in [b]$, let:

$$\text{EWIS}(G, w, S, m) = \begin{cases} yes, & \text{if } S = \{label(v)\} \text{ and } m = w(v) \\ no, & \text{otherwise} \end{cases}$$

If $G = G_1 \oplus G_2$, then let for all $S \subseteq [k]$, and for all $m \in [b]$:

$$\text{EWIS}(G, w, S, m) = \begin{cases} yes, & \text{if EWIS}(G_1, w, S, m) = \textit{yes} \\ yes, & \text{if EWIS}(G_2, w, S, m) = \textit{yes} \\ yes, & \text{if there is an } m' \in [m-1] \text{ such that} \\ & \text{EWIS}(G_1, w, S, m') = \text{EWIS}(G_2, w, S, m - m') \\ & = \textit{yes} \\ no, & \text{otherwise} \end{cases}$$

This can be calculated in time $O(b^2)$, similarly as in Corollary 16.3.

If $G = \eta_{i,j}(G_1)$ then let for all $S \subseteq [k]$, and for all $m \in [b]$:

$$\text{EWIS}(G, w, S, m) = \begin{cases} \text{EWIS}(G_1, w, S, m), & \text{if } \{i, j\} \not\subseteq S \\ no, & \text{otherwise} \end{cases}$$

If $G = \rho_{i \to j}(G_1)$ then let for all $S \subseteq [k]$, and for all $m \in [b]$:

$$\text{EWIS}(G, w, S, m) = \begin{cases} \text{EWIS}(G_1, w, S, m), & \text{if } S \cap \{i, j\} = \{\,\} \\ \text{EWIS}(G_1, w, S \cup \{i\}, m), & \text{if } S \cap \{i, j\} = \{j\} \\ no, & \text{otherwise} \end{cases}$$

Having calculated all the values $\text{EWIS}(G, w, S, m)$, the solution to $\text{EWIS}(G, w, b)$ is clearly given by:

$$\text{EWIS}(G, w, b) = \begin{cases} yes, & \text{if there is an } S \subseteq [k] \text{ such that EWIS}(G, w, S, b) = \textit{yes} \\ no, & \text{otherwise} \end{cases}$$

∎

The same algorithm runs in pseudo-polynomial time whenever the clique-width of the input graph is of the order $O(\log n)$.

Due to the unknown complexity of the exact perfect matching problem, the problem of determining the complexity of EWIS is of particular interest for line graphs of

bipartite graphs, and their subclasses and superclasses. Line graphs of bipartite graphs form a hereditary class of graphs. Their characterization in terms of forbidden induced subgraphs has been obtained in [STA 98], as follows. A graph G is the line graph of a bipartite graph if and only if G is $\mathcal{F}$-free, where $\mathcal{F} = \{claw, diamond, C_5, C_7, \ldots\}$. A *diamond* is the graph obtained by deleting a single edge from a complete graph on 4 vertices.

Keeping in mind this characterization of line graphs of bipartite graphs, it is interesting to consider the following immediate consequence of Theorem 16.15.

COROLLARY 16.3.– The EWIS problem admits a pseudo-polynomial solution in each of the following graph classes:

– $(claw, co\text{-}claw)$-free graphs,

– $(gem, fork, co\text{-}P)$-free graphs (Figure 16.4) and their subclass $(claw, diamond, co\text{-}P)$-free graphs,

– $(P_5, diamond)$-free graphs.

Proof. Each of the above subclasses is of bounded clique-width (see [BRA 02, BRA 03a, BRA 04]). ∎

Also, we can derive from Theorem 16.15 a particular complexity result for the exact perfect matching problem.

COROLLARY 16.4.– For every fixed k, the exact perfect matching problem admits a pseudo-polynomial algorithm for graphs of treewidth at most k.

Proof. As shown by Gurski and Wanke [GUR 07], a set $\mathcal{G}$ of graphs has bounded treewidth if and only if the set $L(\mathcal{G}) = \{L(G) : G \in \mathcal{G}\}$ has bounded clique-width. Since the exact perfect matching problem in $\mathcal{G}$ is polynomially equivalent to the EWIS_α problem in the set $L(\mathcal{G})$, the statement follows from Theorem 16.15 and part (ii) of Lemma 16.1. ∎

16.4.2. *Modular decomposition*

The idea of *modular decomposition* was first described in the 1960s by Gallai [GAL 67], and also appeared in other works under various other names such as *prime tree decomposition* [EHR 90], *X-join decomposition* [HAB 79], or *substitution decomposition* [MÖH 85]. This technique makes it possible to reduce many graph problems from arbitrary graphs to so-called *prime* graphs. In this section, we show how to apply modular decomposition to the EWIS problem.

Let $G = (V, E)$ be a graph, U a subset of V and x a vertex of G outside U. We say that x *distinguishes* U if x has both a neighbor and a non-neighbor in U. A subset

$U \subset V(G)$ is called a *module* in G if it is indistinguishable for the vertices outside U. A module U is *non-trivial* if $1 < |U| < |V|$, otherwise it is *trivial*. A graph whose every module is trivial is called *prime*.

An important property of maximal modules is that if G and co-G are both connected, then the maximal modules of G are pairwise disjoint. Moreover, from the above definition it follows that if U and U' are maximal modules, then either there are all possible edges between U and U', or there are no edges between them. This property is crucial for the modular decomposition, which provides a reduction of many graph problems from a graph G to the graph G^0 obtained from G by contracting each maximal module to a single vertex.

We formally describe this reduction for the EWIS problem in the recursive procedure MODULAR_EWIS(G, W, b) below. It turns out that in order to apply this decomposition to the EWIS problem, we need to relax the problem so that each vertex of the input graph is equipped with a nonempty set of possible weights (instead of just a single weight). For simplicity, we still name this problem EWIS. When all sets are singletons, the problem coincides with the original EWIS problem.

EXACT WEIGHTED INDEPENDENT SET (EWIS)
Instance: an ordered triple (G, W, b), where $G = (V, E)$ is a graph, b is a positive integer and $W = (W_v : v \in V)$ with $W_v \subseteq [b]$ for all $v \in V$ is the collection of possible weights for each vertex of G.
Question: is there an independent set I of G and a mapping $w : I \to [b]$ such that $w(v) \in W_v$ for all $v \in I$ and $\sum_{v \in I} w(v) = b$?

In graph classes that are closed under duplicating vertices, this extended version is pseudo-polynomially equivalent to the original one: given an input (G, W, b) to the extended version, we can construct a weighted graph (G', w') from (G, W) by replacing each vertex v of G with a clique K_v on $|W_v|$ vertices, assigning different weights from W_v to different vertices of K_v, and joining a vertex from K_u with a vertex from K_v by an edge if and only if $[u, v]$ was an edge of G. Then, it is clear that EWIS$(G, W, b) = yes$ if and only if EWIS$(G', w', b) = yes$. However, working with the extended version enables us to apply modular decomposition to *arbitrary* graph classes.

Algorithm MODULAR_EWIS(G, W, b)

Input: An ordered triple (G, W, b), where $G = (V, E)$ is a graph, b is a positive integer and $W = (W_v : v \in V)$ with $W_v \subseteq [b]$ for all $v \in V$ is the collection of possible weights for each vertex of G.

Output: (EWIS$(G, W, k) : k \in [b]$)

1. If $|V| = 1$, for example, $V = \{v\}$, set, for each $k \in [b]$:

$$\text{EWIS}(G, W, k) = \begin{cases} yes, & \text{if } k \in W_v \\ no, & \text{otherwise} \end{cases}$$

and stop.

2. If G is disconnected, partition it into connected components $\mathcal{M}_1, \ldots, \mathcal{M}_r$ and go to step 5.

3. If co-G is disconnected, partition G into co-components $\mathcal{M}_1, \ldots, \mathcal{M}_r$ and go to step 5.

4. If G and co-G are connected, partition G into maximal modules $\mathcal{M}_1, \ldots, \mathcal{M}_r$.

5. Construct a graph G^0 from G by contracting each $\mathcal{M}_j$ (for $j \in [r]$) to a single vertex and assign to that vertex the set of weights:

$$W_{\mathcal{M}_j} = \{k \in [b] : \text{EWIS}(G[\mathcal{M}_j], W, k) = yes\}$$

6. For each $k \in [b]$, let:

$$\text{EWIS}(G, W, k) = \text{EWIS}(G^0, (W_{\mathcal{M}_j} : j \in [r]), k)$$

and stop.

We note that for each input graph, at most one of the steps 2-4 is performed (at most one among $\{G$, co-$G\}$ is disconnected; moreover, if G and co-G are both connected, then the maximal modules of G are pairwise disjoint). Observe that the graph G^0 constructed in step 5 of the algorithm is either an edgeless graph, a complete graph, or a prime graph. Therefore, the modular decomposition approach reduces the problem from a graph to its prime induced subgraphs.

The correctness of the procedure is straightforward: every independent set I of G consists of pairwise disjoint independent sets in the subgraphs of G induced by $\mathcal{M}_1, \ldots, \mathcal{M}_r$; moreover, those $\mathcal{M}_i$ that contain a vertex from I form an independent set in G^0. And conversely, for every independent set I^0 in G^0 and every choice of independent sets $\{I_j : j \in I^0\}$ with I_j independent in $G[\mathcal{M}_j]$, the set $\cup_{j \in [r]} I_j$ is independent in G.

The following theorem answers the question on the complexity of such a reduction.

THEOREM 16.16.– *Let $\mathcal{G}$ be a class of graphs and $\mathcal{G}^*$ the class of all prime induced subgraphs of the graphs in $\mathcal{G}$. If there is a $p \geqslant 1$ and a $q \geqslant 2$ such that the EWIS problem can be solved for graphs in $\mathcal{G}^*$ in time $O(b^q n^p)$, then the EWIS problem can be solved for graphs in $\mathcal{G}$ in time $O(b^q n^p + m)$.*

Proof. Let G be a graph in $\mathcal{G}$ with n vertices and m edges. The recursive decomposition of G produced by the algorithm can be implemented in time $O(n+m)$ [McC 99]. This

decomposition associates with G a tree $T(G)$ whose leaves correspond to the vertices of G, while the internal nodes of $T(G)$ represent induced subgraphs of G with at least two vertices.

Consider an internal node U of $T(G)$ and let G_U denote the induced subgraph of G corresponding to U. Then the children of G_U correspond to the subgraphs $G[\mathcal{M}_1], \ldots, G[\mathcal{M}_r]$, where $\{\mathcal{M}_1, \ldots, \mathcal{M}_r\}$ is the partition of G_U defined in steps 2–4 of the algorithm.

If G_U is disconnected, then G_U^0 is an empty graph, and the problem can be solved for G_U^0 in time $O(b^2|V(G_U^0)|)$, since it is a generalized subset sum problem (cf. Lemma 16.4). If G_U is disconnected, then G_U^0 is a complete graph, and the problem can be solved trivially for G_U^0 in time $O(b|V(G_U^0)|)$.

If both G and co-G are connected, then G_U^0 is a prime induced subgraph of G, and the problem can be solved for G_U^0 in time $O(b^q|V(G_U^0)|^p)$ by our assumption. Summing up over all internal nodes of $T(G)$, we conclude that the total time complexity of the problem on G is bounded by $O(b^q \sum_U |V(G_U^0)|^p)$. It is not difficult to see that the total number of vertices in all graphs G_U^0 corresponding to internal nodes $U \in V(T(G))$ equals the number of edges of $T(G)$, i.e., $|V(T(G))| - 1$. Since the number of leaves of $T(G)$ is n and the number of internal nodes is at most $n-1$, we conclude that:

$$b^q \sum_U |V(G_U^0)|^p \leqslant b^q \left(\sum_U |V(G_U^0)| \right)^p \leqslant b^q (2n-2)^p = O(b^q n^p)$$

By adding the term $O(n+m)$ needed to obtain the decomposition tree, we obtain the desired time complexity. The theorem is proved. ∎

Just like for the weighted independent set problem, modular decomposition is the key to pseudo-polynomial-time solutions to the EWIS problem in several subclasses of P_5-free and *fork*-free graphs. The results are summarized in the following theorem; all graphs mentioned in the theorem or its proof are depicted in Figure 16.4.

THEOREM 16.17.– *The EWIS problem is solvable in pseudo-polynomial time for each of the following classes:*

– (P_5, *double-gem, co-domino*)*-free graphs (and their subclass,* (P_5, *co-P*)*-free graphs),*

– (*bull, fork*)*-free graphs,*

– (*co-P, fork*)*-free graphs,*

– (P_5, *fork*)*-free graphs.*

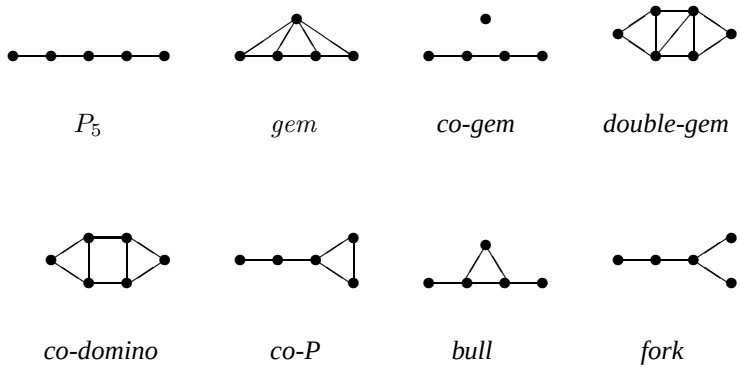


Figure 16.4. *Some 5- and 6-vertex graphs*

Proof. This theorem essentially follows from Theorem 16.16 and the results in [BRA 04a] and [HOÀ 89] (see also [BRA 04b] for some applications of modular decomposition to the WIS problem). We briefly summarize the main ideas.

Every prime (P_5, *double-gem*, *co-domino*)-free graph is $2K_2$-free (the complementary version of this statement is proved in [HOÀ 89]). Since we can easily extend Theorem 16.4 to the extended version of the EWIS problem, this implies the result for (P_5, *double-gem*, *co-domino*)-free graphs.

The (extended) EWIS problem can be solved in pseudo-polynomial time for *co-gem*-free graphs. Indeed, for every vertex v of a *co-gem*-free graph G, the non-neighborhood of v in G is P_4-free. So the problem amounts to solving $O(nb)$ subproblems in P_4-free graphs, which can be done by modular decomposition. It is well known (see, for example, [COR 81]) that every P_4-free graph is either disconnected, or its complement is disconnected. Thus, the only prime P_4-free graph is the graph on a single vertex.

In [BRA 04a], it is shown that prime graphs that contain a *co-gem* and are either $(bull, fork)$-free, $(co\text{-}P, fork)$-free or $(P_5, fork)$-free have a very simple structure. The (extended) EWIS problem can be solved in pseudo-polynomial time for such graphs. Together with the above observation about *co-gem*-free graphs and Theorem 16.16, this concludes the proof. ∎

16.5. Conclusion

As we saw in the introduction, the motivation for studying the exact weighted independent set problem comes from the fact that the complexity of the exact matching

problem is still unknown, even for bipartite graphs. As the exact matching problem for bipartite graphs is the same as the exact weighted independent set problem for line graphs of bipartite graphs, the problem of determining the complexity of the EWIS problem is of particular interest for line graphs of bipartite graphs, and their subclasses and superclasses. We will now show that the class $L(Bip)$ of line graphs of bipartite graphs is sandwiched between two graph classes for which the complexity of the EWIS problem is known, and whose sets of forbidden induced subgraphs differ only in two graphs.

Let us recall that the line graphs of bipartite graphs are precisely the $(claw, diamond, C_5, C_7, \ldots)$-free graphs. Replacing the diamond in the above characterization by its subgraph C_3 results in a smaller class of $(K_{1,3}, C_3, C_5, C_7, \ldots)$-free graphs. It is easy to see that this is precisely the class of bipartite graphs of maximum degree 2. Every connected graph in this class is either an even cycle or a path, and the treewidth of such graphs is at most 2. According to Corollary 16.3 and Theorem 16.14, the problem is solvable in pseudo-polynomial time in this class.

On the other hand, if we replace $K_{1,3}$ with $K_{1,4}$ in the above characterization of $L(Bip)$, we obtain a class of graphs that strictly contains line graphs of bipartite graphs. This class of $(K_{1,4}, diamond, C_5, C_7, \ldots)$-free graphs contains the class of $(K_{1,4}, C_3, C_5, C_7, \ldots)$-free graphs, which is precisely the class of bipartite graphs of maximum degree at most 3. The results of section 16.3.1 imply that the problem is strongly **NP**-complete for this class, and hence also for the larger class of $(K_{1,4}, diamond, C_5, C_7, \ldots)$-free graphs.

To summarize, the class $L(Bip)$ of line graphs of bipartite graphs is sandwiched between two graph classes for which the complexity of the EWIS problem is known, as the following diagram shows:

	$Free(\{K_{1,3}, C_3, C_5, C_7, \ldots\})$	pseudo-polynomial
$\subset$	$L(Bip)$	???
$\subset$	$Free(\{K_{1,4}, diamond, C_5, C_7, \ldots\})$	strongly **NP**-complete

16.6. Bibliography

[ALE 91] ALEKSEEV V.E., "On the number of maximal independent sets in graphs from hereditary classes", *Combinatorial-algebraic methods in discrete optimization*, University of Nizhny Novgorod, p. 5-8, 1991 (in Russian).

[ARN 89] ARNBORG S., PROSKUROWSKI A., "Linear time algorithms for NP-hard problems restricted to partial k-trees", *Discrete Appl. Math.*, vol. 23, p. 11-24, 1989.

[BAL 89] BALAS E., YU C.S., "On graphs with polynomially solvable maximum-weight clique problem", *Networks*, vol. 19, p. 247-253, 1989.

[BAN 86] BANDELT H.J., MULDER H.M., "Distance-hereditary graphs", *J. Combin. Theory Ser. B*, vol. 41, p. 182-208, 1986.

[BAR 87] BARAHONA F., PULLEYBLANK W.R., "Exact arborescences, matchings, and cycles", *Discrete Appl. Math.*, vol. 16, p. 91-99, 1987.

[BER 73] BERGE C., *Graphs and hypergraphs*, North Holland, Amsterdam, 1973.

[BŁA 06] BŁAŻEWICZ J., FORMANOWICZ P., KASPRZAK M., SCHUURMAN P., WOEGINGER G., "A polynomial time equivalence between DNA sequencing and the exact perfect matching problem", *Discrete Optim.*, (2006), doi:10.1016/j.disopt.2006.07.004.

[BOD 96] BODLAENDER H.L., KLOKS T., "Efficient and constructive algorithms for the pathwidth and treewidth of graphs", *J. Algorithms*, vol. 21, p. 358-402, 1996.

[BRA 99] BRANDSTÄDT A., LE V.B., SPINRAD J., *Graph classes: a survey.*, SIAM Monographs on Discrete Mathematics and Applications. SIAM, Philadelphia, PA, 1999.

[BRA 02] BRANDSTÄDT A., MAHFUD S., "Maximum weight stable set on graphs without claw and co-claw (and similar graph classes) can be solved in linear time", *Inform. Process. Lett.*, vol. 84, p. 251-259, 2002.

[BRA 03] BRANDSTÄDT A., DRAGAN F.F., "On linear and circular structure of (claw, net)-free graphs", *Discrete Appl. Math.*, vol. 129, p. 285-303, 2003.

[BRA 03a] BRANDSTÄDT A., LE H.-O., VANHERPE J.-M., "Structure and stability number of chair-, co-P- and gem-free graphs revisited", *Inform. Process. Lett.*, vol. 86, p. 161-167, 2003.

[BRA 04] BRANDSTÄDT A., "(P_5, diamond)-free graphs revisited: structure and linear time optimization", *Discrete Appl. Math.*, vol. 138, p. 13-27, 2004.

[BRA 04a] BRANDSTÄDT A., LE V.B., DE RIDDER H.N., "Efficient robust algorithms for the maximum weight stable set problem in chair-free graph classes", *Inform. Process. Lett.*, vol. 89, p. 165-173, 2004.

[BRA 04b] BRANDSTÄDT A., HOÀNG C.T., VANHERPE J.-M., "On minimal prime extensions of a four-vertex graph in a prime graph", *Discrete Math.*, vol. 288, p. 9-17, 2004.

[BRO 99] BROERSMA H., KLOKS T., KRATSCH D., MÜLLER H., "Independent sets in asteroidal triple-free graphs", *SIAM J. Discrete Math.*, vol. 12, p. 276-287, 1999.

[COG 05] COGIS O., THIERRY E., "Computing maximum stable sets for distance-hereditary graphs", *Discrete Optim.*, vol. 2, p. 185-188, 2005.

[COR 81] CORNEIL D.G., LERCHS H., STEWART-BURLINGHAM L., "Complement reducible graphs", *Discrete Appl. Math.*, vol. 3, p. 163-174, 1981.

[COR 05] CORNEIL D.G., ROTICS U., "On the relationship between clique-width and treewidth", *SIAM J. Comput.*, vol. 34, p. 825-847, 2005.

[COU 97] COURCELLE B., "The expression of graph properties and graph transformations in monadic second-order logic", *Handbook of graph grammars and computing by graph transformation*, vol. 1, p. 313-400, World Sci. Publishing, River Edge, NJ, 1997.

[DAM 01] DAMIAND G., HABIB M., PAUL C., "A simple paradigm for graph recognition: application to cographs and distance hereditary graphs", *Combinatorics and computer science (Palaiseau, 1997). Theoret. Comput. Sci.*, vol. 263, p. 99-111, 2001.

[DAW 01] DAWANDE M., KESKINOCAK P., SWAMINATHAN J.M. , TAYUR S., "On Bipartite and Multipartite Clique Problems", *J. Algorithms*, vol. 41, p. 388-403, 2001.

[DEĬ 06] DEĬNEKO V.G., WOEGINGER G.J., "On the robust assignment problem under a fixed number of cost scenarios", *Oper. Res. Lett.*, vol. 34, p. 175-179, 2006.

[EHR 90] EHRENFEUCHT A., ROZENBERG G., "Primitivity is hereditary for 2-structures", *Theoret. Comput. Sci.*, vol. 70, p. 343-358, 1990.

[FRA 76] FRANK A., "Some polynomial algorithms for certain graphs and hypergraphs", *Proc. of the 5th Brit. Comb. Conf., Aberdeen 1975, Congr. Numer. XV*, p. 211-226, 1976.

[GAL 67] GALLAI T., "Transitiv orientierbare graphen", *Acta Math. Acad. Sci. Hungar.*, vol. 18, p. 25-66, 1967.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and intractability. A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco, 1979.

[GRÖ 84] GRÖTSCHEL M., LOVÁSZ L., SCHRIJVER A., "Polynomial algorithms for perfect graphs", *Topics on perfect graphs*, p. 325-356, North-Holland Math. Stud., 88, North-Holland, Amsterdam, 1984.

[GUR 07] GURSKI F., WANKE E., "Line graphs of bounded clique-width ", *Discrete Math.*, to appear, 2007.

[HAB 79] HABIB M., MAURER M.C., "On the X-join decomposition for undirected graphs", *Discrete Appl. Math.*, vol. 1, p. 201-207, 1979.

[HOÀ 89] HOÀNG C.T., REED B., "Some classes of perfectly orderable graphs", *J. Graph Theory*, vol. 13, p. 445-463, 1989.

[HSU 99] HSU W.-L., MA T.-H., "Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs", *SIAM J. Comput.*, vol. 28, p. 1004-1020, 1999.

[KAR 87] KARZANOV A.V., "Maximum matching of given weight in complete and complete bipartite graphs", *Cybernetics*, vol. 23, p. 8-13, 1987. Translation from *Kibernetika*, vol. 1, p. 7-11, 1987.

[LEC 86] LECLERC M., "Polynomial time algorithms for exact matching problems", Master's Thesis, University of Waterloo, Waterloo, 1986.

[MAN 07] MANNINO C., ORIOLO G., RICCI F., CHANDRAN S., "The stable set problem and the thinness of a graph", *Oper. Res. Lett.*, vol. 35, p. 1-9, 2007.

[McC 99] MCCONNELL R.M., SPINRAD J.P., "Modular decomposition and transitive orientation", *Discrete Math.*, vol. 201, p. 199-241, 1999.

[MÖH 85] MÖHRING R.H., "Algorithmic aspects of comparability graphs and interval graphs", in I. Rival (Ed.), *Graphs and Orders*, D. Reidel, Boston, 1985. p. 41-101.

[MUL 87] MULMULEY K., VAZIRANI U., VAZIRANI V.V., "Matching is as easy as matrix inversion", *Combinatorica*, vol. 7, p. 105-113, 1987.

[MUR 92] MURPHY O.J., "Computing independent sets in graphs with large girth", *Discrete Appl. Math.*, vol. 35, p. 167-170, 1992.

[OKA 05] OKAMOTO Y., UNO T., UEHARA R., "Linear-time counting algorithms for independent sets in chordal graphs,", *Graph-theoretic concepts in computer science, Lecture Notes in Comput. Sci., 3787*, Springer, Berlin, 2005, p. 433-444.

[OUM 05] OUM S.I., "Approximating rank-width and clique-width quickly", *Graph-theoretic concepts in computer science, Lecture Notes in Comput. Sci., 3787*, Springer, Berlin, 2005, p. 49-58.

[PAP 82] PAPADIMITRIOU C.H., YANNAKAKIS M., "The complexity of restricted spanning tree problems", *J. ACM*, vol. 29, p. 285-309, 1982.

[POL 74] POLJAK S., "A note on stable sets and coloring of graphs", *Comment. Math. Univ. Carolinae*, vol. 15, p. 307–309, 1974.

[PRI 95] PRISNER E., "Graphs with few cliques", *Inform. Process. Lett.*, vol. 1, 2, (Kalamazoo, MI, 1992), p. 945-956, 1995, Wiley-Interscience Publ., Wiley, New York.

[RAM 88] RAMALINGAM J., PANDU RANGAN C., "A unified approach to domination problems on interval graphs", *Inform. Process. Lett.*, vol. 27, p. 271-274, 1988.

[STA 98] STATON W., WINGARD G.C., "On line graphs of bipartite graphs", *Util. Math.*, vol. 53, p. 183-187, 1998.

[SUP 87] SUPOWIT K.J., "Finding a maximum planar subset of a set of nets in a channel", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-6, p. 93-94, 1987.

[TAR 85] TARJAN R.E., "Decomposition by clique separators", *Discrete Math.*, vol. 55, p. 221-232, 1985.

[THO 06] THOMAS R., "A survey of Pfaffian orientations of graphs", *Proc. of the Internat. Congress of Mathematicians*, Vol. III (Madrid, 2006), p. 963–984, EMS Publishing House, 2006.

[TSU 77] TSUKIYAMA S., IDE M., ARIYOSHI H., SHIRAKAWA I., "A new algorithm for generating all the maximal independent sets", *SIAM J. Comput.*, vol. 6, p. 505-517, 1977.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 17

The Labeled Perfect Matching in Bipartite Graphs: Complexity and (in)Approximability

17.1. Introduction

This chapter presents some recent works accomplished by the author ([MON 05, MON 06]) about the complexity and the approximation properties of the LABELED perfect matching problems.

Let Π be a **NPO** problem accepting simple graphs $G = (V, E)$ as instances, edge-subsets $E' \subseteq E$ satisfying a given polynomial-time decidable property $Pred$ as solutions, and the solutions cardinality as objective function. The labeled problem associated with Π, denoted by LABELED Π, consists, given an instance $I = (G, L)$ where $G = (V, E)$ is a simple graph and L is a mapping from E to $\{c_1, \ldots, c_q\}$, of determining a subset E' satisfying $Pred$ that optimizes the size of the set $L(E') = \{L(e) : e \in E'\}$. Note that two versions of LABELED Π may be considered according to the optimization goal: LABELED Min Π that consists of minimizing $|L(E')|$ and LABELED Max Π that consists of maximizing $|L(E')|$. Roughly speaking, the mapping L corresponds to assigning a color (or a label) to each edge and the goal of LABELED Min Π (resp., Max Π) is to find an edge subset using the fewest (resp., the most) number of colors. If a given **NPO** problem Π is **NP**-hard, then the associated labeled problem LABELED Π is clearly **NP**-hard (consider a distinct color per edge). For instance, the LABELED longest path problem or the LABELED maximum induced matching problem are both **NP**-hard. Moreover, if the decision problem (derived from Π) where we aim at deciding if a graph G contains an edge subset satisfying $Pred$,

Chapter written by Jérôme MONNOT.

or not, is **NP**-complete, then LABELED $Min\ \Pi$ cannot be approximated within performance ratio better than $2-\varepsilon$ for all $\varepsilon > 0$ unless **P**=**NP**, even if the graph is complete. Indeed, if we color the edges from $G=(V,E)$ with a color per edge and then we complete the graph, adding a new color per edge, then it is **NP**-complete to decide between $\mathrm{opt}(I)=1$ and $\mathrm{opt}(I)\geqslant 2$, where $\mathrm{opt}(I)$ is the value of an optimal solution. Notably, it is the case of the LABELED traveling salesman problem or the LABELED minimum partition problem into paths of length k for any $k\geqslant 2$.

Thus, labeled problems have been mainly studied, from a complexity and an approximability point of view, when Π is polynomial, [BRO 97, BRO 05, BRU 03, CHA 97, KRU 98, WAN 02, XIO 05]. For example, the first labeled problem introduced in other works is the LABELED minimum spanning tree problem, which has several applications in communication network design. This problem is **NP**-hard and many complexity and approximability results have been proposed in [BRO 97, BRU 03, CHA 97, KRU 98, WAN 02, XIO 05]. On the other hand, the LABELED maximum spanning tree problem has been shown polynomial in [BRO 97]. Very recently, the LABELED path and the LABELED cycle problems have been studied in [BRO 05, HAS 06]; in particular, in [BRO 05] authors prove that the LABELED minimum path problem is **NP**-hard and give some exact algorithms, whereas in [HAS 06] several approximation algorithms with performance guarantee are presented. Note that the **NP**-completeness also appears in [CAR 00] since the LABELED path problem is a special case of the red-blue set cover problem. Some other results can be found in [HAS 07] when the objective labeled function is of type bottleneck.

In this chapter, we go thoroughly into the investigation of the complexity and the approximability of labeled problems, with the analysis of the matching problem in bipartite graphs. The maximum matching problem is one of the most known combinatorial optimization problems and arises in several applications such as image analysis, artificial intelligence or scheduling. It turns out that a problem very closed to it has been studied in other works and is called in [ITA 78] the restricted perfect matching problem. This latter aims at determining, given a graph $G=(V,E)$, a partition $E_1,\ldots,E_k$ of E and k positive integers $r_1,\ldots,r_k$, whether there exists a perfect matching M on G satisfying for all $j=1,\ldots,k$ the restrictions $|M\cap E_j|\leqslant r_j$. This problem has some relationship with the timetable problem, since a solution may be seen as a matching between classes and teachers that satisfies additional restrictions (for instance, no more that r laboratories at the same time). The restricted perfect matching problem is proved to be **NP**-complete in [ITA 78], even if (i) $|E_j|\leqslant 2$, (ii) $r_j=1$, and (iii) G is a bipartite graph. On the other hand, it is shown in [YI 02] that the restricted perfect matching problem is polynomial when G is a complete bipartite graph and $k=2$; some others results of this problem can be found in [COS 07]. A perfect matching M only satisfying condition (ii) (i.e., $|M\cap E_i|\leqslant 1$) is called good in [CAM 97]. Thus, we deduce that the LABELED maximum perfect matching problem is **NP**-hard in bipartite graph since $\mathrm{opt}(I)=n$ iff G contains a good matching.

In section 17.2, we analyze both the complexity and the approximability of the LABELED minimum perfect matching problem and the LABELED maximum perfect matching problem in 2-regular bipartite graphs. In particular, we deduce that both these problems are in **APX** when the graph has a maximum degree 2. Then, in section 17.3, we propose some inapproximation results when the bipartite graphs have a maximum degree at least 3 or are 3-regular. Actually, we prove first that LABELED minimum perfect matching is not in **APX** whenever the bipartite graphs have a maximum degree of 3. Hence, there is a gap of approximability between graphs of maximum degree 2 and 3. Using a weaker complexity hypothesis, we can even obtain that LABELED minimum perfect matching is not $2^{O(log^{1-\varepsilon}n)}$-approximable in bipartite graphs of maximum degree 3 on n vertices, unless **NP** $\subseteq$ **DTIME** $\left(2^{O(log^{1/\varepsilon}n)}\right)$. Dealing with the unbounded degree case, this yields to the fact that LABELED minimum perfect matching is not in **polyLog-APX**, unless **P** $=$ **NP**. Finally, section 17.4 focuses on the case of complete bipartite graphs. In particular, it is shown that a greedy algorithm picking at each iteration a monocolored matching of maximum size provides a $\frac{r+H_r}{2}$-approximation in bipartite complete graphs where r is the maximum number of times that a color appears in the graph and H_r is the r-th harmonic number.

Now, we introduce some terminology and notations that will be used in the chapter. A *matching* M on a graph $G = (V, E)$ is a subset of edges that are pairwise non-adjacent; M is said to be a *perfect matching* if it covers the vertex set of G. A graph $G = (V, E)$ is *bipartite* if the vertex set can be partitioned into two sets, the left set L and the right set R such that every edge of G has an endpoint in L and the other in R. In the labeled perfect matching problem (LABELED PM in short), we are given a simple graph $G = (V, E)$ on $|V| = 2n$ vertices which contains a perfect matching together with a color (or label) function $L : E \rightarrow \{c_1, \ldots, c_q\}$ on the edge set of G. For $i = 1, \ldots, q$, we denote by $L^{-1}(\{c_i\}) \subseteq E$ the set of edges of color c_i. The goal of LABELED $Min\ PM$ (resp., $Max\ PM$) is to find a perfect matching on G using a minimum (resp., a maximum) number of colors. An equivalent formulation of LABELED $Min\ PM$ could be the following: if $G[\mathcal{C}]$ denotes the subgraph induced by the edges of colors $\mathcal{C} \subseteq \{c_1, \ldots, c_q\}$, then LABELED $Min\ PM$ aims at finding a subset $\mathcal{C}$ of minimum size such that $G[\mathcal{C}]$ contains a perfect matching. The restriction of LABELED PM to the case where each color occurs at most r times in $I = (G, L)$ (i.e., $|L^{-1}(\{c_i\})| \leqslant r$ for $i = 1, \ldots, q$) will be denoted by LABELED PM_r. The LABELED $Min\ PM$ problem has some relationship with the timetable problem, since a solution may be seen as a matching between classes and teachers that satisfies additional restrictions (for instance, a color corresponds to a school where we assume that a professor may teach in several schools). An inspector would like to assess all teachers during one lecture of each one of them and it would be desirable that he does not visit the same class twice. Hence, the lectures to be attended would form a maximum matching. For convenience, the inspector would like these lectures to take place in the smallest possible number of schools. Then clearly the inspector has to construct a maximum matching meeting a minimum number of colors in the graph associated with the

lectures. In [RIC 92], a generalization, called perfect matching under categorization, has been studied. In this framework, each edge e has also a non-negative weight $w(e)$, and the colors are called categories (thus, q indicates the number of categories). The goal is to find a perfect matching M of E minimizing $\sum_{i=1}^{q} \max_{e \in \mathcal{L}_i \cap M} w(e)$. In [RIC 92], it is shown that, on the one hand, the problem is polynomial when the number of categories (i.e., colors) is fixed and, on the other hand, the problem is **NP**-hard when the weights take values 0 or 1 and the graph is a collection of disjoint 4-cycles. Note that the case $w(e) = 1, \forall e \in E$ corresponds to LABELED $Min\ PM$.

We denote by $\mathrm{opt}(I)$ and $\mathrm{apx}(I)$ the value of an optimal and an approximate solution, respectively. We say that an algorithm $\mathcal{A}$ is an ε-approximation of LABELED $Min\ PM$ with $\varepsilon \geqslant 1$ (resp., $Max\ PM$ with $\varepsilon \leqslant 1$) if $\mathrm{apx}(I) \leqslant \varepsilon \times \mathrm{opt}(I)$ (resp., $\mathrm{apx}(I) \geqslant \varepsilon \times \mathrm{opt}(I)$) for any instance $I = (G, L)$.

17.2. The 2-regular bipartite case

In this section, we deal with a particular class of graphs that consist of a collection of pairwise disjoint cycles of even length; note that such graphs are 2-regular bipartite graphs.

THEOREM 17.1.– LABELED $Min\ PM_r$ *is* ***APX****-complete in 2-regular bipartite graphs for any* $r \geqslant 2$.

Proof. Observe that any solution of LABELED $Min\ PM_r$ is an r-approximation. The rest of the proof will be done via an approximation preserving reduction from the minimum balanced satisfiability problem with clauses of size at most r, MIN BALANCED r-SAT for short. An instance $I = (\mathcal{C}, X)$ of MIN BALANCED r-SAT consists of a collection $\mathcal{C} = (C_1, \ldots, C_m)$ of clauses over the set $X = \{x_1, \ldots, x_n\}$ of Boolean variables, such that each clause C_j has at most r literals and each variable appears positively as many times as negatively; B_i denotes this number for any $i = 1, \ldots, n$. The goal is to find a truth assignment f satisfying a minimum number of clauses. MIN BALANCED 2-SAT where $2 \leqslant B_i \leqslant 3$ has been shown **APX**-complete by the way of an L-reduction from MAX BALANCED 2-SAT where $B_i = 3$ [BER 98, KAR 05].

We only prove the case $r = 2$. Let $I = (\mathcal{C}, X)$ be an instance of MIN BALANCED 2-SAT on m clauses $\mathcal{C} = \{C_1, \ldots, C_m\}$ and n variables $X = \{x_1, \ldots, x_n\}$ such that each variable x_i has either 2 positive occurrences and 2 negative occurrences, or 3 positive occurrences and 3 negative occurrences. We build the instance $I' = (H, L)$ of LABELED $Min\ PM_2$ where H is a collection of pairwise disjoint cycles $\{H(x_1), \ldots, H(x_n)\}$ and L is a function which colors edges of H with colors $c_1, \ldots, c_j, \ldots, c_m$, by applying the following process:

- For each variable x_i, create the $2B_i$-long cycle

$$H(x_i) = \{e_{i,1}, \ldots, e_{i,k}, \ldots, e_{i,2B_i}\}$$

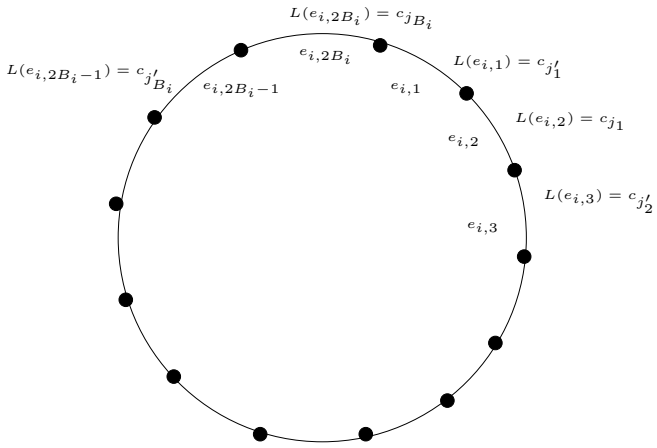


Figure 17.1. *The gadget $H(x_i)$ and the color of its edges*

• Color the edges of $H(x_i)$ as follows: if x_i appears positively in clauses $C_{j_1}, \ldots, C_{j_{B_i}}$ and negatively in clauses $C_{j'_1}, \ldots, C_{j'_{B_i}}$, then set $L(e_{i,2k}) = c_{j_k}$ and $L(e_{i,2k-1}) = c_{j'_k}$ for $k = 1, \ldots, B_i$.

Figure 17.1 provides an illustration of the gadget $H(x_i)$. Clearly, H is made of n disjoint cycles and is painted with m colors. Moreover, each color appears at most twice.

Let f^* be an optimal truth assignment on I satisfying m^* clauses and consider the perfect matching $M = \cup_{i=1}^{n} M_i$ where $M_i = \{e_{i,2k} | k = 1, \ldots, B_i\}$ if $f(x_i) = true$, $M_i = \{e_{i,2k-1} | k = 1, \ldots, B_i\}$ otherwise; M uses exactly m^* colors and thus:

$$\mathrm{opt}(I) \leqslant m^* \qquad [17.1]$$

Conversely, let M' be a perfect matching on H using $\mathrm{apx}(I) = m'$ colors; if we set $f'(x_i) = true$ if $e_{i,2} \in M'$, $f'(x_i) = false$ otherwise, we can easily observe that the truth assignment f' satisfies m' clauses:

$$\mathrm{apx}(I) = \mathrm{val}(f') \qquad [17.2]$$

Hence, using inequalities [17.1] and [17.2] the result follows. ∎

Trivially, the problem becomes obvious when each color is used exactly once. We now show that we have a 2-approximation in 2-regular bipartite graphs, thus showing that the restriction of LABELED $Min\ PM$ to 2-regular bipartite graphs is as easy to approximate as MINSAT.

THEOREM 17.2.– *There exists an approximation preserving reduction from* LABELED $Min\ PM$ *in 2-regular bipartite graphs to* MINSAT *of expansion* $c(\varepsilon) = \varepsilon$.

Proof. The result comes from the reciprocal of the previous transformation. Let $I = (G, L)$ be an instance of LABELED $Min\ PM$ where $G = (V, E)$ is a collection $\{H_1, \ldots, H_n\}$ of disjoint cycles of even length and $L(E) = \{c_1, \ldots, c_m\}$ defines the label set. We describe every cycle H_i as the union of two matchings M_i and $\overline{M_i}$. We construct an instance $I' = (\mathcal{C}, X)$ of the satisfiability problem, MINSAT where $\mathcal{C} = \{C_1, \ldots, C_m\}$ is a set of m clauses and $X = \{x_1, \ldots, x_n\}$ is a set of n variables, as follows. The clause set $\mathcal{C}$ is in one-to-one correspondence with the color set $L(E)$ and the variable set X is in one-to-one correspondence with the connected components of G; a literal x_i (resp., $\overline{x_i}$) appears in C_j iff $c_j \in L(M_i)$ (resp., $c_j \in L(\overline{M_i})$). We easily deduce that any truth assignment f on I' that satisfies k clauses can be converted into a perfect matching M_f on I that uses k colors. ∎

Using the 2-approximation of MINSAT [MAR 96] and Theorem 17.2, we deduce:

COROLLARY 17.1.– LABELED $Min\ PM$ in 2-regular bipartite graphs is 2-approximable.

When dealing with LABELED $Max\ PM_r$, the result of [ITA 78] shows that calculating a *good matching* is **NP**-hard even if the graph is bipartite and each color appears at most twice; a good matching M is a perfect matching using $|M|$ colors. Thus, we deduce from this result that LABELED $Max\ PM_r$ is **NP**-hard for any $r \geqslant 2$. We strengthen this result using a reduction from MAX BALANCED 2-SAT.

THEOREM 17.3.– $Max\ PM_r$ *is* ***APX****-complete in 2-regular bipartite graphs for any* $r \geqslant 2$.

In the same way, using the approximate result for MAXSAT [ASA 02], we obtain:

COROLLARY 17.2.– LABELED $Max\ PM$ in 2-regular bipartite graphs is 0.7846-approximable.

17.3. Some inapproximation results

In order to simplify the proofs exposed in the rest of the section, the results concern a variation of LABELED $Min\ PM$, where the value of each perfect matching M is given by $\mathrm{val}_1(M) = \mathrm{val}(M) - 1$. This problem is denoted by LABELED $Min\ PPM$ and we have for any instance I, $\mathrm{apx}_1(I) = \mathrm{apx}(I) - 1$ and

$\mathrm{opt}_1(I) = \mathrm{opt}(I) - 1$. It is important to note that a $\rho(n)$-approximation of LABELED $Min\ PM$ becomes a $2\rho(n)$-approximation of LABELED $Min\ PPM$ and conversely a $\rho(n)$-approximation of LABELED $Min\ PPM$ remains a $\rho(n)$-approximation of LABELED $Min\ PM$. Actually, since LABELED $Min\ PM$ is simple [PAZ 81] (i.e., the restriction to $\mathrm{opt}(I) \leqslant k$ is polynomial), we can see that LABELED $Min\ PM$ and LABELED $Min\ PPM$ are asymptotically equivalent to approximate. Hence, the proposed results for LABELED $Min\ PPM$ also hold LABELED $Min\ PM$.

We now propose a self-improving operation for some classes of instances $\mathcal{P}_k$ described as follows. $I = (H, \mathcal{L}) \in \mathcal{P}_k$ where $H = (V, E)$ if and only if the following properties are satisfied:

(i) H is planar of maximum degree k and connected.

(ii) $\exists u, v \in V$ such that $[u, u_1]$ and $[v, v_1]$ for some $u_1, v_1 \in V$ are the only edges incident to u and v. Moreover, these two edges have color c_0, i.e., $\mathcal{L}([u, u_1]) = \mathcal{L}([v, v_1]) = c_0$.

(iii) H is bipartite and admits a perfect matching.

(iv) $H[\{c_0\}]$, the subgraph induced by edges of color c_0 does not have any perfect matching and the subgraph $H[\mathcal{L}(E) \setminus \{c_0\}]$ induced by edges of colors different from c_0 is acyclic.

(v) If $H' = H \setminus \{u, v\}$ denotes the subgraph induced by $V \setminus \{u, v\}$, then $H'[\{c_0\}]$ has a perfect matching denoted by M_{c_0}.

We have $\mathcal{P}_1 = \{\ \}$ and $\mathcal{P}_2$ is the set of odd paths from u to v alternating matchings M and M_{c_0} where M_{c_0} is only colored by color c_0. Finally, we define the class $\mathcal{P}$ by $\mathcal{P} = \cup_k \mathcal{P}_k$.

Restricted label squaring operation. Given an instance $I = (H, \mathcal{L}) \in \mathcal{P}_k$ of LABELED $Min\ PM$, its *label squaring instance* is $I^2 = (H^2, \mathcal{L}^2)$ with $H^2 = (V^2, E^2)$, where:

1) The graph H^2 is created by removing each edge $e = [x, y]$ of H with color different from c_0 and placing instead of it a copy $H(e)$ of H, such that x and y are now identified with u and v of H, respectively.

2) For each copy $H(e)$ of H and for an edge e' in $H(e)$ with color different from c_0, the new color of e' is $\mathcal{L}^2(e') = (\mathcal{L}(e), \mathcal{L}(e'))$. The remaining edges of copy $H(e)$ keep their color c_0, i.e., if $\mathcal{L}(e') = c_0$, then $\mathcal{L}^2(e') = c_0$.

Let us prove that classes $\mathcal{P}_k$ are closed under the restricted label squaring operation.

LEMMA 17.1.– *If $I \in \mathcal{P}_k$, then $I^2 \in \mathcal{P}_k$.*

Proof. Let $I \in \mathcal{P}_k$. The proofs of (i) and (ii) are obvious.

For (iii), since H and $H \setminus \{u, v\}$ admit a perfect matching, we deduce that $u \in L$ and $v \in R$ where (L, R) is the bipartition of H. Thus, we can extend the bipartition to H^2 by taking for each $H(e)$ a copy of the bipartition. Finally, it is easy to verify that H^2 admits a perfect matching if H does.

For (iv) assume the reverse, i.e., that $H^2[\{c_0\}]$ admits a perfect matching M and $H[\{c_0\}]$ does not. By hypothesis, in each copy $H([x, y])$, the vertices x and y are not saturated by M and then the edges of M which do not traverse copies $H(e)$ form a perfect matching of $H[\{c_0\}]$, which is a contradiction. Moreover, using property (ii), it is easy to verify that the subgraph $H^2[\mathcal{L}^2(E^2) \setminus \{c_0\}]$ is acyclic whenever $H[\mathcal{L}(E) \setminus \{c_0\}]$ is acyclic.

For (v) let M_{c_0} be a perfect matching of $H' = H \setminus \{u, v\}$ only using color c_0. We complete M_{c_0} by taking for each copy $H(e)$ a copy of M_{c_0}. In this way, we obtain a perfect matching of $H^2 \setminus \{u, v\}$ that uses only color c_0. ∎

We now propose an approximation preserving reduction using the label squaring operation on $\mathcal{P}_k$.

THEOREM 17.4.– *Let $I = (H, \mathcal{L}) \in \mathcal{P}_k$. If there exists a (polynomial) ρ-approximation of I^2 for* LABELED $Min\ PPM$*, then there exists a $\sqrt{\rho}$-approximation of I for* LABELED $Min\ PPM$.

Proof. Let M^* be an optimal perfect matching of $I \in \mathcal{P}_k$ using $\mathrm{opt}(I)$ colors and let $e_1, \cdots, e_p$ be the edges of H using colors which are different from c_0. For each copy $H(e_i)$ we take a copy of M^* using colors $(\mathcal{L}(e_i), \mathcal{L}(e_j))$ for $j = 1, \cdots, p$ and color c_0. For the remaining copies, we take a copy of M_{c_0} (a perfect matching on $H \setminus \{u, v\}[\{c_0\}]$) and we complete this matching into a perfect matching of H^2 using the remaining edges of M^*. This matching uses $(\mathrm{opt}(I) - 1)^2 + 1$ colors and thus:

$$\underset{1}{\mathrm{opt}}(I^2) \leqslant \overset{2}{\underset{1}{\mathrm{opt}}}(I) \qquad [17.3]$$

Now, consider an approximate perfect matching M^2 of H^2 with value $\mathrm{apx}(I^2)$ and let $H(e_1), \cdots, H(e_p)$ be the copies of H such that the restriction of M^2 to $H(e_i)$ is a perfect matching. Hence, we may always assume that $M^2 \setminus (\cup_{i=1}^{p} H(e_i))$ only uses color c_0. Therefore, if we denote $\mathcal{L}' = \{\mathcal{L}(e_i) : i = 1, \cdots, p\}$, then for any $c_j \in \mathcal{L}'$ there exists a perfect matching $M_{c_j,k} \subseteq M^2$ in copy $H(e_k)$ such that edge e_k has color c_j. Let M_{c_j} be a matching of H minimizing $|\mathcal{L}(M_{c_j,k})|$ for any $c_j \in \mathcal{L}'$ and let M_0 be a perfect matching of H containing edges $\{e_1, \ldots, e_p\}$ and some other edges of color c_0.

The approximate perfect matching M of I will be given by one of the matchings M_{c_j} or M_0 with value $\mathrm{apx}(I) = \min\{|\mathcal{L}(M_0)|, |\mathcal{L}(M_{c_j})| : c_j \in \mathcal{L}'\}$. Thus, we

deduce that $\operatorname{apx}_1(I) = \operatorname{apx}(I) - 1 = \min\{|\mathcal{L}(M_0)| - 1, |\mathcal{L}(M_{c_j})| - 1 : c_j \in \mathcal{L}'\}$ and hence:

$$\begin{aligned} \underset{1}{\overset{2}{\operatorname{apx}}}(I) &\leqslant (|\mathcal{L}(M_0)| - 1) \min\{|\mathcal{L}(M_{c_j})| - 1 : c_j \in \mathcal{L}'\}; \\ &\leqslant \sum_{c_j \in \mathcal{L}'} \big(|\mathcal{L}(M_{c_j})| - 1\big) \leqslant \underset{1}{\operatorname{apx}}(I^2) \end{aligned} \quad [17.4]$$

By applying inequality [17.4] with an optimal perfect matching M^2 of H^2, we obtain $\operatorname{opt}_1^2(I) \leqslant \operatorname{opt}_1(I^2)$. Using inequality [17.3], we deduce $\operatorname{opt}_1^2(I) = \operatorname{opt}_1(I^2)$ and the expected result follows. ∎

THEOREM 17.5.– LABELED $Min\ PPM$ *is not* $c \log n$ *approximable for some constant* $c > 0$ *for instances in* $\mathcal{P}$ *having* $2n$ *vertices, unless* ***P*=*NP***.

Proof. Given a family $\mathcal{S} = \{S_1, \ldots, S_{n_0}\}$ of subsets of a ground set $X = \{x_1, \ldots, x_{m_0}\}$ (we assume that $\cup_{i=1}^{n_0} S_i = X$), a set cover of X is a subfamily $\mathcal{S}' = \{S_{f(1)}, \ldots, S_{f(p)}\} \subseteq \mathcal{S}$ such that $\cup_{i=1}^{p} S_{f(i)} = X$. The *set cover problem* denoted by MINSC is the problem of determining a minimum-size set cover $\mathcal{S}^* = \{S_{f^*(1)}, \ldots, S_{f^*(q)}\}$ of X. Given an instance $I_0 = (\mathcal{S}, X)$ of MINSC, its *characteristic graph* $G_{I_0} = (L_0, R_0; E_{I_0})$ is a bipartite graph with a left set $L_0 = \{l_1, \ldots, l_{n_0}\}$ that represents the members of the family $\mathcal{S}$ and a right set $R_0 = \{r_1, \ldots, r_{m_0}\}$ that represents the elements of the ground set X; the edge-set E_{I_0} of the characteristic graph is defined by $E_{I_0} = \{[l_i, r_j] : x_j \in S_i\}$.

From I_0, we construct the instance $I = (H, \mathcal{L})$ of LABELED $Min\ PPM$ containing $(n_0 + 1)$ colors $\{c_0, c_1, \cdots, c_{n_0}\}$, described as follows:

• For each element $x_j \in X_0$, we build a gadget $H(x_j)$ that consists of a bipartite graph of $2(d_{G_{I_0}}(r_j) + 3)$ vertices and $3d_{G_{I_0}}(r_j) + 4$ edges, where $d_{G_{I_0}}(r_j)$ denotes the degree of vertex $r_j \in R$ in G_{I_0}. The graph $H(x_j)$ is illustrated in Figure 17.2.

• Assume that vertices $\{l_{f(1)}, \ldots, l_{f(p)}\}$ are the neighbors of r_j in G_{I_0}, then color $H(x_j)$ as follows: for any $k = 1, \ldots, p$, $L(v_{3,j}, l_{j,f(k)}) = c_{f(k)}$ and the other edges receive color c_0.

– We complete $H = \cup_{x_j \in X} H(x_j)$ by adding edges $[v_{2,j}, v_{1,j+1}]$ with color c_0 for $j = 1, \cdots, m_0 - 1$.

– Finally, we set $u = v_{1,1}$ and $v = v_{2,m_0}$.

Clearly, $I \in \mathcal{P}$ and has $2n = 2\sum_{r_j \in R}(d_{G_{I_0}}(r_j) + 3) = 2|E_{I_0}| + 6m_0$ vertices.

Let $\mathcal{S}^*$ be an optimal set cover on I_0. From $\mathcal{S}^*$, we can easily construct a perfect matching M^* of $I = (H, \mathcal{L})$ that uses exactly $(|\mathcal{S}^*| + 1)$ colors. Conversely, let M be

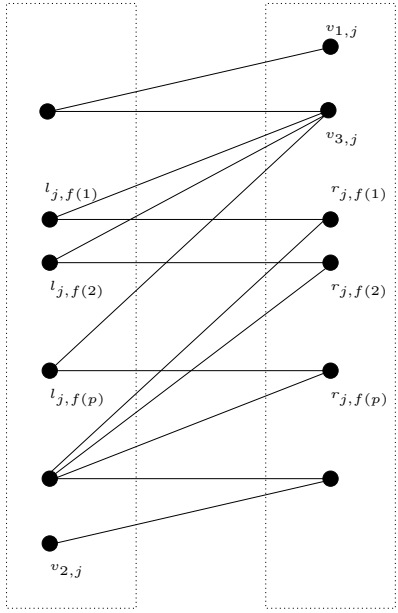


Figure 17.2. *The gadget $H(x_j)$*

a perfect matching on I; by construction, the subset $\mathcal{S}' = \{S_k : c_k \in \mathcal{L}(M)\}$ of $\mathcal{S}$ is a set cover of X using $(|\mathcal{L}(M)| - 1)$ sets.

Now, it is well known that the set cover problem is **NP**-hard to approximate within factor $c \log n_0$ for some constant $c > 0$. This result also applies to instances $(X, \mathcal{S})$ when $|X|$ and $|\mathcal{S}|$ are polynomially related (i.e., $|X|^q \leqslant |\mathcal{S}| \leqslant |X|^p$ for some constants p, q).

Hence, given such an instance $I_0 = (X, \mathcal{S})$, from any algorithm A solving LABELED $Min\ PPM$ within a performance ratio $\rho_{\mathrm{A}}(I) \leqslant \frac{c}{q+1} \times \log(n)$ for a bipartite graph on $2n$ vertices, we can deduce an algorithm for MINSC that guarantees the performance ratio $c\frac{1}{q+1}\log(n) \leqslant c\frac{1}{q+1}\log(n_0^{q+1}) = c\log(n_0)$, which is a contradiction. ∎

Starting from the **APX**-completeness result for the vertex cover problem in cubic graphs, [ALI 00], we are able to obtain the following result.

COROLLARY 17.3.– LABELED $Min\ PPM$ for instances in $\mathcal{P}_3$ is not in **PTAS**.

Proof. Starting from the restriction of set cover where each element x_i is covered by exactly two sets (this case is usually called the *vertex cover problem* and denoted by

MINVC), we apply the same proof as in Theorem 17.5. The instance I becomes an element of $\mathcal{P}_3$ and by using, for instance, the hardness result of [ALI 00], the expected result follows. ∎

By applying the well known self-improving method, we obtain the two following results:

THEOREM 17.6.– LABELED $Min\ PPM$ *for instances in* $\mathcal{P}_3$ *is not in* ***APX****, unless* ***P*** $=$ ***NP****.*

Proof. Assume the reverse and let A be a polynomial algorithm solving LABELED $Min\ PPM$ within a constant performance ratio ρ. Let $\varepsilon > 0$ (with $\varepsilon < \rho - 1$) and choose the smallest integer q such that:

$$q \geqslant \log\log\rho - \log\log(1+\varepsilon) \qquad [17.5]$$

Consider now an instance $I = (H, \mathcal{L}) \in \mathcal{P}_3$ and use the restricted label squaring operation on I. We produce the instance $I^2 = (H^2, \mathcal{L}^2)$ and by repeating q times this operation on I^2, we obtain, thanks to Lemma 17.1, the instance $I^{2^q} = (H^{2^q}, \mathcal{L}^{2^q}) \in \mathcal{P}_3$, in time $P(|I|)$ for some polynomial P since on the one hand, I^2 is obtained from I in time $O(|I|^2)$ (we have $|V(H^2)| = O(|V(H)|^2)$ and $|\mathcal{L}^2(E(H^2))| = O(|\mathcal{L}(E(H))|^2)$) and on the other hand, we repeat this operation a constant number of times. Using Theorem 17.4, from the ρ-approximation on I^{2^q} given by A , we obtain a $\rho^{2^{-q}}$-approximation on I. Thanks to inequality [17.5], we deduce $\rho^{2^{-q}} \leqslant 1 + \varepsilon$. Hence, we obtain a polynomial time approximation scheme for instances in $\mathcal{P}_3$, which is a contradiction to Corollary 17.3. ∎

THEOREM 17.7.– *For any* $\varepsilon > 0$ LABELED $Min\ PPM$ *is not* $2^{O(log^{1-\varepsilon}n)}$*-approximable for instances in* $\mathcal{P}_3$ *on* n *vertices, unless* ***NP*** $\subseteq$ ***DTIME*** $\left(2^{O(log^{1/\varepsilon}n)}\right)$.

Proof. Let $\varepsilon > 0$ and $I = (H, \mathcal{L}) \in \mathcal{P}_3$ where H has n vertices. Choose the smallest integer p such that $n^{2^p} \geqslant 2^{\log^{1/\varepsilon} n}$. Thus, $2^{2^p \times \log n} \geqslant 2^{\log^{1/\varepsilon} n}$ and then:

$$2^{p \times \varepsilon} \geqslant \log^{1-\varepsilon} n \qquad [17.6]$$

Using the restricted label squaring operation on I, we produce the instance $I^2 = (H^2, \mathcal{L}^2)$. By repeating p times this operation on I^2, we obtain the instance $I^{2^p} = (H^{2^p}, \mathcal{L}^{2^p}) \in \mathcal{P}_3$. Since H has n vertices, we derive from property (iv) of Lemma

17.1 that the number n' of vertices of H^{2^p} and the number $|\mathcal{L}^{2^p}(E(H^{2^p}))|$ of colors of H^{2^p} satisfy:

$$n' \leqslant n^{2^p} \text{ and } |\mathcal{L}^{2^p}(E(H^{2^p}))| \leqslant |\mathcal{L}(E(H))|^{2^p} \qquad [17.7]$$

Now, assume that we have a $f(n')$-approximation on I^{2^p} where $f(n') \leqslant 2^{c \times \log^{1-\varepsilon} n'}$ for some $c > 0$. Using Theorem 17.4, we obtain a $f(n')^{2^{-p}}$-approximation on I. Using inequalities [17.6] and [17.7], we deduce:

$$\begin{aligned}
\underset{1}{\text{apx}}(I) &\leqslant f(n')^{2^{-p}} \underset{1}{\text{opt}}(I) \\
&\leqslant 2^{c \times \frac{\log^{1-\varepsilon} n'}{2^p}} \underset{1}{\text{opt}}(I) \\
&\leqslant 2^{c \times \frac{\log^{1-\varepsilon} n}{2^{\varepsilon \times p}}} \underset{1}{\text{opt}}(I) \\
&\leqslant 2^c \underset{1}{\text{opt}}(I)
\end{aligned}$$

Thus, using inequality [17.7], we obtain a constant approximation in time $poly(n') = 2^{O(\log^{1/\varepsilon} n)}$, and thus, a contradiction with Theorem 17.6. ∎

It is natural to wonder whether the problem is easier in 3-regular bipartite graphs (also called cubic bipartite graphs) or not. Here, we prove that the answer is negative.

THEOREM 17.8.– LABELED $Min\ PPM$ *is not in* ***APX*** *in connected planar cubic bipartite graphs, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. The proof consists of two steps. First, using a quite similar reduction to that of Corollary 17.3, we prove that Theorem 17.7 also holds for the subfamily $\mathcal{P}_3'$ of $\mathcal{P}_3$ where each vertex has a degree 3, except u and v. Then, we transform any instance of $\mathcal{P}_3'$ into a connected planar cubic bipartite graph.

Let $G = (V, E)$ with $V = \{v_1, \cdots, v_n\}$ and $E = \{e_1, \cdots, e_n\}$ be an instance of vertex cover. We transform any edge $e_j = [x, y]$ into gadget $H(e_j)$ described in Figure 17.3. All edges of $H(e_j)$, except $[v_{3,j}, l_{j,x}]$ and $[v_{3,j}, l_{j,y}]$, have color c_0. We have $\mathcal{L}([v_{3,j}, l_{j,x}]) = c_x$ and $\mathcal{L}([v_{3,j}, l_{j,y}]) = c_y$. Finally, $H(e_j)$ is linked to $H(e_{j+1})$ using the graph depicted in Figure 17.4 where each edge is colored with c_0.

Clearly, LABELED $Min\ PPM$ is **APX**-hard in class $\mathcal{P}_3'$. Since the restricted label squaring operation also preserves the membership in $\mathcal{P}_3'$, we deduce that LABELED

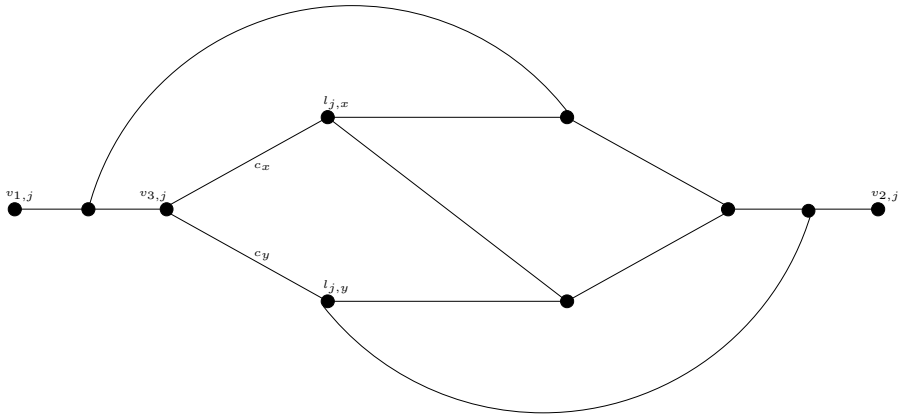


Figure 17.3. *The gadget $H(e_j)$ for $e_j = [x, y]$*

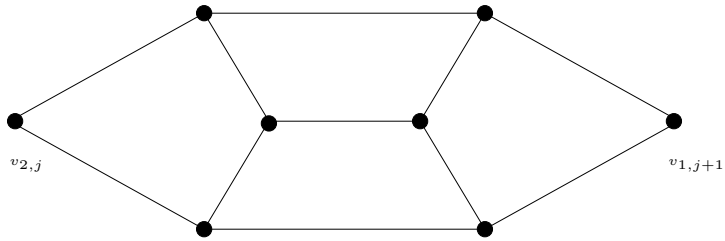


Figure 17.4. *The gadget linking $H(e_j)$ to $H(e_{j+1})$*

$Min\ PPM$ is not in **APX** when the instances are restricted to $\mathcal{P}'_3$. Finally, given $I \in \mathcal{P}'_3$ with $I = (G, \mathcal{L})$, we consider the instance I' where G is duplicated 3 times into G_1, G_2, G_3. If u_i, v_i denote the extreme vertices of G_i, we shrink vertices u_1, u_2, u_3 into u and v_1, v_2, v_3 into v. Clearly, this new graph G' is connected, bipartite, planar and cubic. Finally, since we can restrict ourselves to perfect matchings M' of G' that use only color c_0 for exactly two copies of G, the result follows. ∎

Dealing with the unbounded degree case (i.e., instances of $\mathcal{P}$), we can deduce the following stronger result:

THEOREM 17.9.– LABELED $Min\ PPM$ *for instances in $\mathcal{P}$ is not in* ***polyLog-APX****, unless* $\boldsymbol{P = NP}$.

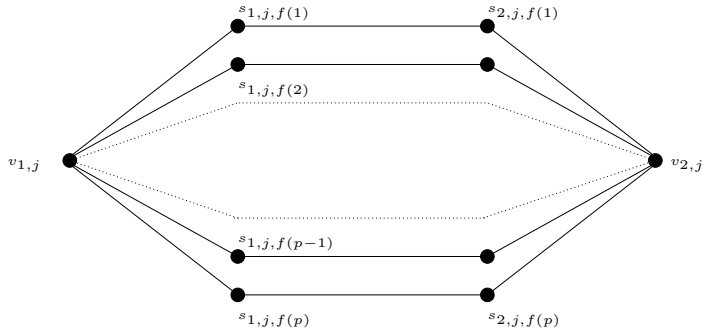


Figure 17.5. *The gadget $H(x_j)$*

Proof. Assume the reverse, i.e., LABELED $Min\ PPM$ is $f(n)$-approximable with $f(n) \leqslant c\log^k n$ for some constants $c > 0$ and $k \geqslant 1$. Let $I = (H, \mathcal{L}) \in \mathcal{P}$ where H has $2n$ vertices. Let $p = \lceil logk \rceil + 1$. Using as previously 2^p times the restricted label squaring operation on I, we produce in polynomial-time the instance $I^{2^p} = (H^{2^p}, \mathcal{L}^{2^p}) \in \mathcal{P}$. The same arguments as in Theorem 17.7 allow us to obtain a contradiction with Theorem 17.5. ∎

17.4. The complete bipartite case

When considering complete bipartite graphs, we obtain several results:

THEOREM 17.10.– LABELED $Min\ PM_r$ *is* ***APX****-complete in complete bipartite graphs $K_{n,n}$ for any $r \geqslant 6$.*

Proof. We give an approximation preserving L-reduction (see Papadimitriou and Yannakakis [PAP 91]) from a restriction of the set cover problem, which is called MINSC$_3$. In this restriction each set is of size at most 3 and each element x_j appears in at most 3 and at least 2 different sets. MINSC$_3$ has been proved **APX**-complete in [PAP 91].

Let G_{I_0} be the characteristic graph corresponding to an instance $I_0 = (\mathcal{S}, X)$ of MINSC$_3$ (see the proof of Theorem 17.5 for a formal definition of MINSC and G_{I_0}). Note that G_{I_0} has a maximum degree 3. From I_0, we construct the instance $I = (K_{n,n}, L)$ of LABELED $Min\ PM_6$ using a slight modification of the construction given in Theorem 17.5). First, we start from a bipartite graph having m_0 connected components $H(x_j)$ and $n_0 + m_0$ colors $\{c_1, \ldots, c_{n_0+m_0}\}$, which is described as follows:

• For each element $x_j \in X$, we build a gadget $H(x_j)$ that consists of a bipartite graph of $2(d_{G_{I_0}}(r_j)+1)$ vertices and $3d_{G_{I_0}}(r_j)$ edges, where $d_{G_{I_0}}(r_j)$ denotes the degree of vertex $r_j \in R$ in G_{I_0}. The graph $H(x_j)$ is illustrated in Figure 17.5.

• Assume that vertices $\{l_{f(1)}, \ldots, l_{f(p)}\}$ are the neighbors of r_j in G_{I_0}, then color $H(x_j)$ as follows: for any $k = 1, \ldots, p$, $L(v_{1,j}, s_{1,j,f(k)}) = L(v_{2,j}, s_{2,j,f(k)}) = c_{f(k)}$ and $L(s_{1,j,f(k)}, s_{2,j,f(k)}) = c_{n_0+j}$.

– We complete $H = \cup_{x_j \in X} H(x_j)$ into $K_{n,n}$, by adding a new color per edge.

Clearly, $K_{n,n}$ is complete bipartite and has $2n = 2\sum_{r_j \in R}(d_{G_{I_0}}(r_j)+1) = 2|E_{I_0}| + 2m_0$ vertices. Moreover, each color is used at most 6 times.

Let $\mathcal{S}^*$ be an optimal set cover on I_0. From $\mathcal{S}^*$, we can easily construct a perfect matching M^* on I using exactly $|\mathcal{S}^*| + m_0$ colors (since we assume that each element appears in at least 2 sets) and thus:

$$\underset{\text{LABELED } Min\ PM_6}{\text{opt}}(I) \leqslant \underset{\text{MINSC}_3}{\text{opt}}(I_0) + m_0 \qquad [17.8]$$

Conversely, we show that any perfect matching M may be transformed into a perfect matching M" using the edges of H and satisfying: $|L(M")| \leqslant |L(M)|$. Let M be a perfect matching on I and consider M_1 the subset of edges from M that link two different gadgets $H(x_j)$. We denote by G the multi-graph of vertex set $\cup_j V_{H(x_j)}$ (each vertex v_j of G corresponds to the gadget $H(x_j)$) and of edge set M_1. So, G is obtained from $K_{n,n}$ where the vertices in each gadget $H(x_j)$ is shrunk into a vertex of G. Note that each connected component of G is Eulerian. Each cycle C on G may be completed into a $2|C|$-long cycle C' on K_{2n} in such a way that the two endpoints of each edge from $C' \setminus C$ do belong to the same gadget $H(x_j)$. Here, for purely formal reasons of the proof, we assume that each gadget $H(x_j)$ is a complete graph by adding a new color per missing edge. Thus, there are edges linking any two vertices $s_{1,j,f(k)}$ and $s_{1,j,f(k')}$ (or $s_{2,j,f(k)}$ and $s_{2,j,f(k')}$). If we swap the edges from each cycle C by the edges from $C' \setminus C$, we obtain a new perfect matching M' whose every edge has its two endpoints in a same gadget $H(x_j)$ and satisfies $|L(M')| = |L(M)|$. For the moment, note that as indicated previously, the perfect matching M' is not necessarily a matching of $K_{n,n}$ since some edges linking 2 vertices of the same part of gadget $H(x_j)$ may exist. Now consider for any j the set M'_j of edges from $M' \cap H(x_j)$; we set $M"_j = \{[v_{1,j}, s_{1,j,f(k)}], [v_{2,j}, s_{2,j,f(k)}]\} \cup \{[s_{1,j,f(i)}, s_{2,j,f(i)}] | i = 1, \ldots, p\}$ for some k such that $[v_{1,j}, s_{1,j,f(k)}] \in M'_j$ or $[v_{2,j}, s_{2,j,f(k)}] \in M'_j$ (if such a k does not exist, set $k = 1$). In any case, $M" = (M' \setminus M'_j) \cup M"_j$ is a perfect matching of $K_{n,n}$ that uses no more colors than M' does. By applying this procedure for any $j = 1, \ldots, m_0$, we obtain the expected matching $M"$ with value $\text{apx}(I)$. From such a matching, we may obtain a set cover $\mathcal{S}" = \{S_k | c_k \in L(M")\}$ on I_0 satisfying:

$$|\mathcal{S}"| = \text{apx}(I) - m_0 \qquad [17.9]$$

Using [17.8] and [17.9], we deduce $\text{opt}_{\text{LABELED } Min\ PM_6}(I) = \text{opt}_{\text{MINSC}_3}(I_0) + m_0$ and $|\mathcal{S}"| - \text{opt}_{\text{MINSC}_3}(I_0) \leqslant |L(M))| - \text{opt}_{\text{LABELED } Min\ PM_6}(I)$. Finally, since $\text{opt}_{\text{MINSC}_3}(I_0) \geqslant \frac{m_0}{3}$ the result follows. ∎

By applying the same kind of proof to the vertex cover problem in cubic graphs [ALI 00], we obtain that LABELED $Min\ PM_r$ in $K_{n,n}$ is **APX**-complete for any $r \geqslant 3$. In order to establish this fact and starting from a cubic graph $G = (V, E)$, we associate with each edge $e = [x, y] \in E$ a 4-long cycle $\{a_{1,e}, a_{2,e}, a_{3,e}, a_{4,e}\}$ together with a coloration L given by: $L(a_{1,e}) = c_x$, $L(a_{2,e}) = c_y$ and $L(a_{3,e}) = L(a_{4,e}) = c_e$. We complete this graph into a complete bipartite graph, adding a new color per edge. Each color c_x ($\forall x \in V$) appears 3 times; c_e ($\forall e \in E$) twice and any other color once. Hence, the application of the proof that was made in Theorem 17.10 leads to the announced result. Unfortunately, we cannot apply the proof of Theorem 17.2 since in this latter, on the one hand, we have some cycles of size 6 and, on the other hand, a color may occur in different gadgets. One open question concerns the complexity of LABELED $Min\ PM_2$ in bipartite complete graphs. Moreover, from Theorem 17.10, we can also obtain a stronger inapproximability result concerning the general problem LABELED $Min\ PM$: we cannot calculate in polynomial-time an approximate solution of LABELED $Min\ PM$ that uses less than $(1/2 - \varepsilon)\ln(\text{opt}_{\text{LABELED } Max\ PM}(I))$ colors in complete bipartite graphs where $\text{opt}_{\text{LABELED } Max\ PM}(I)$ is the value of an optimal solution of LABELED $Max\ PM$, i.e., the maximum number of colors used by a perfect matching.

COROLLARY 17.4.– For any $\varepsilon > 0$, LABELED $Min\ PM$ is not $(\frac{1}{2} - \varepsilon) \times \ln(n)$ approximable in complete bipartite graphs $K_{n,n}$, unless **NP**⊂**DTIME**$(n^{loglogn})$.

Proof. First, we apply the construction made in Theorem 17.10, except that $I_0 = (\mathcal{S}, X)$ is an instance of MINSC such that the number of elements m_0 is strictly larger than the number of sets n_0. From I_0, we construct n_0 instances $I'_1, \ldots, I'_{n_0}$ of LABELED $Min\ PM$ where $I'_i = (H, L_i)$. The colors $L_i(E)$ are the same as $L(E)$, except that we replace colors $c_{n_0+1}, \ldots, c_{n_0+m_0}$ by c_i. Finally, as previously, we complete each instance I'_i into a complete bipartite graph $K_{n,n}$ by adding a new color by edge.

Let $\mathcal{S}^*$ be an optimal set cover on I_0 and, assuming that $S_i \in \mathcal{S}^*$, we consider the instance I_i of LABELED $Min\ PM$. From $\mathcal{S}^*$, we can easily construct a perfect matching M_i^* of I_i that uses exactly $|\mathcal{S}^*|$ colors. Conversely, let M_i be a perfect matching on I_i; by construction, the subset $\mathcal{S}' = \{S_k : c_k \in L(M_i)\}$ of $\mathcal{S}$ is a set cover of X using $|L(M_i)|$ sets. Finally, let A be an approximate algorithm for LABELED $Min\ PM$ and we calculate n_0 perfect matchings M_i by applying A on instances I_i. Thus, if we pick the matching that uses the minimum number x of colors, then we can polynomially construct a set cover on I_0 of cardinality x.

Since $n_0 \leqslant m_0 - 1$, the size n of a perfect matching of $K_{n,n}$ satisfies: $n = |E_{I_0}| + m_0 \leqslant n_0 \times m_0 + m_0 \leqslant m_0(m_0-1) + m_0 = m_0^2$. Hence, from any algorithm A solving LABELED $Min\ PM$ within a performance ratio $\rho_{\rm A}(I) \leqslant \frac{1}{2} \times \ln(n)$, we can deduce an algorithm for MINSC that guarantees the performance ratio $\frac{1}{2}\ln(n) \leqslant \frac{1}{2}\ln(m_0^2) = \ln(m_0)$. Since the negative result of [FEI 98] holds when $n_0 \leqslant m_0 - 1$, i.e., MINSC is not $(1-\varepsilon) \times \ln(m_0)$ approximable for any $\varepsilon > 0$, unless **NP**⊂**DTIME**$(n^{loglogn})$, we obtain a contradiction. ∎

On the other hand, dealing with LABELED $Max\ PM_r$ in $K_{n,n}$, the result of [CAM 97] shows that the case $r = 2$ is polynomial, whereas it becomes **NP**-hard when $r = \Omega(n^2)$. Indeed, it is proved in [CAM 97] that, on the one hand, we can calculate a good matching in $K_{n,n}$ within polynomial-time when each color appears at most twice and, on the other hand, there always exists a good matching in such a graph if $n \geqslant 3$. An interesting question is to determine the complexity and the approximability of LABELED $Max\ PM_r$ when r is a constant greater than 2.

We now study the approximation of LABELED $Min\ PM_r$.

Let us consider the greedy algorithm for LABELED $Min\ PM_r$ in complete bipartite graphs that iteratively picks the color that induces the maximum-size matching in the current graph and deletes the corresponding vertices. Formally, if $L(G')$ denotes the colors that are still available in the graph G' at a given iteration and if $G'[c]$ (resp., $G'[V']$) denotes the subgraph of G' that is induced by the edges of color c (resp., by the vertices V'), then the greedy algorithm consists of the following process:

`Greedy`

1 Set $\mathcal{C}' = \{\}$, $V' = V$ and $G' = G$;

2 While $V' \neq \{\}$ do

2.1 For any $c \in L(G')$, calculate a maximum matching M_c in $G'[c]$;

2.2 Select a color c^* maximizing $|M_c|$;

2.3 $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{c^*\}$, $V' \leftarrow V' \setminus V(M_{c^*})$ and $G' = G[V']$;

3 output $\mathcal{C}'$;

THEOREM 17.11.– *`Greedy` is an $\frac{H_r+r}{2}$-approximation of* LABELED $Min\ PM_r$ *in complete bipartite graphs where H_r is the r-th harmonic number $H_r = \sum_{i=1}^{r} \frac{1}{i}$ and this ratio is tight.*

Proof. Let $I = (G, L)$ be an instance of LABELED $Min\ PM_r$. We denote by $\mathcal{C}'_i$ for $i = 1, \ldots, r$ the set of colors of the approximate solution which appears exactly i

times in $\mathcal{C}'$ and by p_i its cardinality (thus, $\forall c \in \mathcal{C}'_i$ we have $|M_c| = i$ in $G'[c]$); finally, let M_i denote the matching with colors $\mathcal{C}'_i$. If $\mathrm{apx}(I) = |\mathcal{C}'|$, then we have:

$$\mathrm{apx}(I) = \sum_{i=1}^{r} p_i \qquad [17.10]$$

Let $\mathcal{C}^*$ be an optimal solution corresponding to the perfect matching M^* of size $\mathrm{opt}(I) = |\mathcal{C}^*|$; we denote by E_i the set of edges of M^* that belong to $G[\cup_{k=1}^{i} V(M_k)]$, the subgraph induced by $\cup_{k=1}^{i} V(M_k)$ and we set $q_i = |E_i \setminus E_{i-1}|$ (where we assume that $E_0 = \{\}$). For any $i = 1, \ldots, r-1$, we get:

$$\mathrm{opt}(I) \geqslant \frac{1}{i} \sum_{k=1}^{i} q_k \qquad [17.11]$$

Indeed, $\sum_{k=1}^{i} q_k = |E_i|$ and, by construction, each color appears at most i times in $G[\cup_{k=1}^{i} V(M_k)]$.

We also have the following inequality for any $i = 1, \ldots, r-1$:

$$\mathrm{opt}(I) \geqslant \frac{1}{r} \left(2 \sum_{k=1}^{i} k \times p_k - \sum_{k=1}^{i} q_k \right) \qquad [17.12]$$

Since M^* is a perfect matching, the quantity $2 \sum_{k=1}^{i} k \times p_k - \sum_{k=1}^{i} q_k$ counts the edges of M^* of which at least one endpoint belongs to $G[\cup_{k=1}^{i} V(M_k)]$. Because each color appears on at most r edges, the result follows.

Finally, since $\sum_{k=1}^{r} k \times p_k$ is the size of a perfect matching of G, the following inequality holds:

$$\mathrm{opt}(I) \geqslant \frac{1}{r} \sum_{k=1}^{r} k \times p_k \qquad [17.13]$$

Using equality [17.10] and adding inequality [17.11] with coefficient $\alpha_i = \frac{1}{2(i+1)}$ for $i = 1, \ldots, r-1$, inequality [17.12] with coefficient $\beta_i = \frac{r}{2i(i+1)}$ for $i = 1, \ldots, r-1$ and inequality [17.13], we obtain:

$$\mathrm{apx}(I) \leqslant \left(\frac{H_r + r}{2} \right) \mathrm{opt}(I) \qquad [17.14]$$

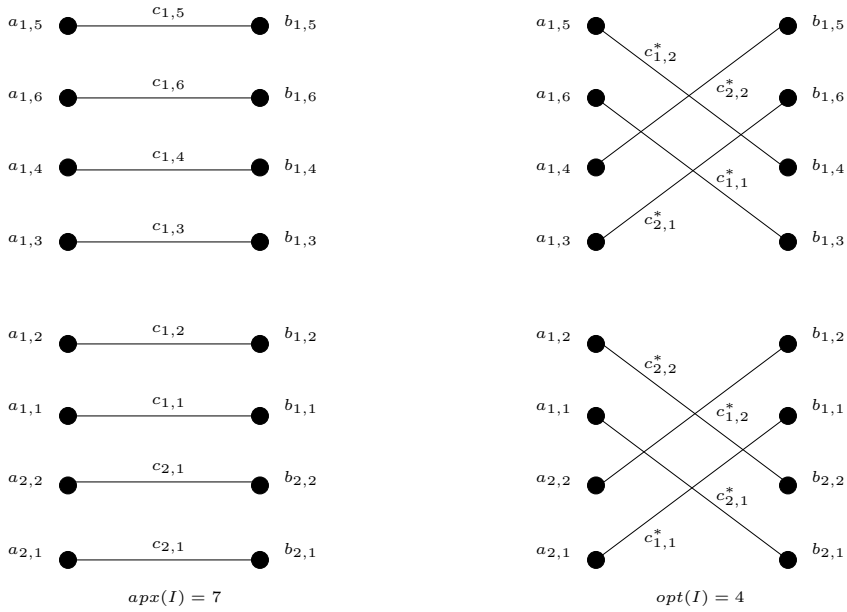


Figure 17.6. *The instance I when $r = 2$*

Indeed, $\sum_{i=1}^{r-1} \alpha_i = \frac{1}{2}H_r - \frac{1}{2}$ and $\sum_{i=1}^{r-1} \beta_i = \frac{r}{2} - \frac{1}{2}$. Thus, $\sum_{i=1}^{r-1} (\alpha_i + \beta_i) + 1 = \frac{H_r + r}{2}$.

The quantity p_j appears in inequality [17.13] and inequality [17.12] for $i = j, \ldots, r-1$. Its total contribution is: $\frac{1}{r} j \times p_j + \frac{2}{r} \left(\sum_{i=j}^{r-1} \beta_i \right) j \times p_j = p_j$. The quantity q_j appears in inequality [17.11] for $i = j, \ldots, r-1$ and inequality [17.12] for $i = j, \ldots, r-1$. We have: $\left(\sum_{i=j}^{r-1} \frac{\alpha_i}{i} \right) - \frac{1}{r} \left(\sum_{i=j}^{r-1} \beta_i \right) q_j = 0$. Thus, using equality [17.10], inequality [17.14] holds.

In order to show the tightness of this bound, consider the instance $I = (K_{n,n}, L)$ where the left set A and the right set B of vertices of the complete bipartite graph are given by $A = \{a_{i,j} : i = 1, \ldots, r,\ j = 1, \ldots, n_i\}$ and $B = \{b_{i,j} : i = 1, \ldots, r,\ j = 1, \ldots, n_i\}$, with $n_1 = (r+1)!$ and $n_i = r!$ for $i = 2, \ldots, r$. Moreover, the edge coloration satisfies:

- For any $i = 1, \ldots, r$ and for any $j = 1, \ldots, n_i$, $L(a_{i,j}, b_{i,j}) = c_{i, \lceil \frac{j}{i} \rceil}$.
- For any $i = 2, \ldots, r$ and for any $j = 1, \ldots, r!$, $L(a_{i,j}, b_{1, i-1+(r-1)(j-1)}) = c^*_{1,j}$ and $L(b_{i,j}, a_{1, i-1+(r-1)(j-1)}) = c^*_{2,j}$.

- For any $j = 1, \ldots, r!$,

$$L(b_{1,j+(r-1)\times r!}, a_{1,(r+1)!-j+1}) = c^*_{1,j}$$

and

$$L(a_{1,j+(r-1)\times r!}, b_{1,(r+1)!-j+1}) = c^*_{2,j}$$

- We associate a new color with each missing edge.

I is clearly an instance of LABELED $Min\ PM_r$. The set of colors $\mathcal{C}' = \{c_{i,\lceil \frac{j}{i} \rceil} : i = 1, \ldots, r,\ j = 1, \ldots, n_i\}$ is the approximate solution outputted by `Greedy` and it uses $\mathrm{apx}(I) = (H_r + r) \times r!$ colors, whereas $\mathcal{C}^* = \{c^*_{i,j} : i = 1, 2,\ j = 1, \ldots, r!\}$ is the set of colors that are used by an optimal solution; this latter satisfies $\mathrm{opt}(I) = 2 \times r!$. The Figure 17.6 describes the instance I for $r = 2$. ∎

17.5. Bibliography

[ALI 00] ALIMONTI P., KANN V., "Some APX-completeness results for cubic graphs", *Theoretical Computer Science*, vol. 237, p. 123-134, 2000.

[ASA 02] ASANO T., WILLIAMSON D. P., "Improved Approximation Algorithms for MAX SAT" , *Journal of Algorithms*, vol. 42, p. 173-202, 2002.

[BER 98] BERMAN P., KARPINSKI M., "On Some Tighter Inapproximability Results", *Electronic Colloquium on Computational Complexity*,vol. 5, 1998.

[BER 03] BERMAN P., KARPINSKI M., SCOTT A. D., "Approximation Hardness of Short Symmetric Instances of MAX-3SAT", *Electronic Colloquium on Computational Complexity*,vol. 10 , 2003.

[BRO 97] BROERSMA H., LI X., "Spanning trees with many or few colors in edgecolored graphs", *Discussiones Mathematicae Graph Theory*,vol. 17, p. 259-269, 1997.

[BRO 05] BROERSMA H., LI X., WOEGINGER G. J., ZHANG S., "Paths and cycles in colored graphs", *Australasian Journal of Combinatorics*,vol. 31, 2005.

[BRU 03] BRÜGGEMANN T., MONNOT J., WOEGINGER G. J., "Local search for the minimum label spanning tree problem with bounded color classes", *Operations Research Letters*,vol. 31, p. 195-201, 2003.

[CAM 97] CAMERON K., "Coloured matchings in bipartite graphs", *Discrete Mathematics*,vol. 169 p. 205-209, 1997.

[CAR 00] CARR R. D., DODDI S., KONJEVOD G., MARATHE M. V., "On the red-blue set cover problem", *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, USA. ACM/SIAM, p. 345-353, 2000.

[CHA 97] CHANG R-S., LEU S-J., "The minimum labeling spanning trees", *Information Processing Letters*,vol. 63, p. 277-282, 1997.

[COS 07] COSTA M. C., DE WERRA D., PICOULEAU C., RIES B., "Bicolored matchings in some classes of graphs", *Graphs and Combinatorics*, (to appear), 2007.

[FEI 98] FEIGE U., "A threshold of for approximating set cove", *Journal of the ACM*,vol. 45, p. 634-652, 1998.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and Intractability. A Guide to the Theory of NP-completeness*, CA, Freeman, 1979.

[HAS 06] HASSIN R., MONNOT J., SEGEV D., "Approximation Algorithms and Hardness Results for Labeled Connectivity Problems", *Rastislav Kralovic, Pawel Urzyczyn (Eds.): Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stara Lesna, Slovakia, Proceedings. Lecture Notes in Computer Science*,vol. 4162, p. 480-491, 2006.

[HAS 07] HASSIN R., MONNOT J., SEGEV D., "The Complexity of Bottleneck Labeled Graph Problems", *submitted*, 2007.

[ITA 78] ITAI A., RODEH M., TANIMOTO S., "Some matching problems in bipartite graphs", *Journal of the ACM*,vol. 25, p. 517-525, 1978.

[KAR 05] KARPINSKI M., "Personal communication", 2005.

[KRU 98] KRUMKE S. O., WIRTH H-C., "On the minimum label spanning tree problem", *Information Processing Letters*,vol. 66, p. 81-85, 1998.

[MAR 96] MARATHE M. V., RAVI S. S., "On Approximation Algorithms for the Minimum Satisfiability Problem", *Information Processing Letters*,vol. 58, p. 23-29 1996.

[MON 05] MONNOT J., "The labeled perfect matching in bipartite graphs", *Information Processing Letters*,vol. 96, p. 81-88 2005.

[MON 06] MONNOT J., "A note on the hardness results for the labeled perfect matching problems in bipartite graphs", *submitted*, 2006.

[PAP 91] PAPADIMITRIOU C. H., YANNAKAKIS M., "Optimization, approximation, and complexity classes", *Journal of Computer and System Sciences*,vol. 43, p. 425-440, 1991.

[PAZ 81] PAZ A., MORAN S., "Non deterministic polynomial optimization problems and their approximation", *Theoretical Computer Science*,vol. 95, p. 251-277, 1981.

[RAZ 97] RAZ R., SAFRA S., "A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP", *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, El Paso, Texas, USA, p. 475-484, 1997.

[RIC 92] RICHEY M. B., PUNNEN A. P., "Minimum Perfect Bipartite Matchings and Spanning Trees under Categorization", *Discrete Applied Mathematics*,vol. 39, p. 147-153, 1992.

[WAN 02] WAN Y., CHEN G., XU Y., "A note on the minimum label spanning tree", *Information Processing Letters*,vol. 84, p. 99-101, 2002.

[XIO 05] XIONG Y., GOLDEN B., WASIL E., "Worst-case behavior of the MVCA heuristic for the minimum labeling spanning tree problem", *Operations Research Letters*,vol. 33, p. 77-80, 2005.

[YI 02] YI T., MURTY K. G., SPERA C., "Matchings in colored bipartite networks", *Discrete Applied Mathematics*,vol. 121, p. 261-277, 2002.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 18

Bounded-size Path Packing Problems

18.1. Introduction

This chapter presents some recent works given by the authors ([MON 07a, MON 07b]) about the complexity and the approximation of several problems on calculating collections of (vertex)-disjoint paths of bounded size.

18.1.1. *Bounded-size path packing problems*

A $\mathbf{P}_k$ partition of the vertex set of a simple graph $G = (V, E)$ is a partition of V into q subsets $V_1, \cdots, V_q$, each of size $|V_i| = k$, such that the subgraph $G[V_i]$ induced by any V_i contains a Hamiltonian path. In other words, the partition $(V_1, \ldots, V_q)$ describes a collection of $|V|/k$ vertex-disjoint simple paths of length $k-1$ (or, equivalently, simple paths on k vertices) on G. The decision problem called $\mathbf{P}_k$ partitioning problem ($\mathbf{P}_k$PARTITION in short) consists, given a simple graph $G = (V, E)$ on $k \times n$ vertices, of deciding whether G admits or not such a partition. The analog problem where the subgraph $G[V_i]$ induced by V_i is isomorphic to $\mathbf{P}_k$ (the chordless path on k vertices) will be denoted by INDUCED $\mathbf{P}_k$PARTITION. These two problems are **NP**-complete for any $k \geqslant 3$, and polynomial otherwise, [GAR 79, KIR 78]. In fact, they both are a particular case of a more general problem called *partition into isomorphic subgraphs*, [GAR 79]. In [KIR 78], Kirkpatrick and Hell give a necessary and sufficient condition for the **NP**-completeness of the partition into isomorphic subgraphs problem in general graphs. $\mathbf{P}_k$PARTITION has been widely studied in other works, mainly because of its closeness to two famous optimization problems, namely:

Chapter written by Jérôme MONNOT and Sophie TOULOUSE.

the minimum k-path partition problem (denoted by MINk-PATHPARTITION) and the maximum $\mathbf{P}_k$ packing problem (denoted by MAX$\mathbf{P}_k$PACKING).

On the one hand, MINk-PATHPARTITION can be viewed as an optimization version of $\mathbf{P}_k$PARTITION where the constrainst on the exact length of the paths is relaxed. MINk-PATHPARTITION consists of partitioning the vertex set of a graph $G = (V, E)$ into the smallest number of paths so that each path has *at most* k vertices (for instance, MIN2-PATHPARTITION is equivalent to the maximum matching problem). The optimal value is usually denoted by $\rho_{k-1}(G)$ for any $k \geqslant 2$ and by $\rho(G)$ when no constraint occurs on the length of the paths (in particular, $\rho(G) = 1$ if and only if G has a Hamiltonian path). MINk-PATHPARTITION has been extensively studied in other works, [STE 03, STE 00, YAN 97], and has applications in broadcasting problems (see, for example, [YAN 97]).

On the other hand, if we relax the exact covering constraint, then we obtain the optimization problems MAX$\mathbf{P}_k$PACKING and MAXINDUCED$\mathbf{P}_k$PACKING which consist, given a simple graph $G = (V, E)$, of finding a maximum number of vertex-disjoint (induced) $\mathbf{P}_k$. When considering the weighted case (denoted by MAXW$\mathbf{P}_k$PACKING and MAXWINDUCED$\mathbf{P}_k$PACKING, respectively), the input graph $G = (V, E)$ is given together with a weight function w on its edges, and the goal is to find a collection $\mathcal{P} = \{P_1, \ldots, P_q\}$ of vertex-disjoint (induced) $\mathbf{P}_k$ that maximizes $w(\mathcal{P}) = \sum_{i=1}^{q} \sum_{e \in P_i} w(e)$.

The special case of MAXW$\mathbf{P}_k$PACKING where the graph is complete on $k \times n$ vertices is called the weighted $\mathbf{P}_k$ partition problem ($\mathbf{P}_k$P in short). In this case, each solution contains exactly n vertex-disjoint paths of length $k - 1$. If the goal is to maximize (MAX$\mathbf{P}_k$P), then we seek a $\mathbf{P}_k$ partition of maximum weight, and if the goal is to minimize (MIN$\mathbf{P}_k$P), then we seek a $\mathbf{P}_k$ partition of minimum weight. When considering the minimization version, it is more often assumed that the instance is metric, i.e., that the weight function satisfies the triangle inequality: $w(x, y) \leqslant w(x, z) + w(z, y), \forall x, y, z$; MINMETRIC$\mathbf{P}_k$P will refer to this restriction. Note that this latter version of the problem is closely related to the vehicle routing problem when restricting the route of each vehicle to at most k intermediate stops, [ARK 06, FRE 78]. Finally, we also will consider the special case of metric instances where the weight function is either 1 or 2; the corresponding problems will be denoted by MAX$\mathbf{P}_k$P$_{1,2}$ and MIN$\mathbf{P}_k$P$_{1,2}$ ($\mathbf{P}_k$P$_{1,2}$ when the goal is not specified). Such a restriction makes sense, since it provides an alternative relaxation of the initial decision problem $\mathbf{P}_k$Partition; moreover, MIN$\mathbf{P}_k$P$_{1,2}$ and MINk-PATHPARTITION are strongly connected.

All these problems are very close to each other. In particular, $\mathbf{P}_k$PARTITION **NP**-completeness implies the **NP**-hardness of both MINk-PATHPARTITION and $\mathbf{P}_k$P

(even when restricting to $\mathbf{P}_k\mathrm{P}_{1,2}$); conversely, $\mathbf{P}_k$PARTITION is polynomial-time decidable on instance families where MINk-PATHPARTITION or MAX$\mathbf{P}_k$PACKING are polynomial-time calculable.

18.1.2. *Complexity and approximability status*

The minimum k-path partition problem is obviously **NP**-complete in general graphs [GAR 79], and remains intractable in comparability graphs, [STE 03], in cographs, [STE 00], and in bipartite chordal graphs, [STE 03] (when k is part of the input). Note that most of the proofs of **NP**-completeness actually establish the **NP**-completeness of $\mathbf{P}_k$PARTITION. Nevertheless, the problem turns out to be polynomial-time solvable in trees, [YAN 97], in cographs when k is fixed, [STE 00] and in bipartite permutation graphs, [STE 03]. Note that we can also find in other works several results dealing with partitioning a graph into disjoint paths of length at least 2, [WAN 94, KAN 03].

This chapter proposes new complexity and inapproximability results for (INDUCED) $\mathbf{P}_k$PARTITION, MINk-PATHPARTITION and MAX(W)(INDUCED)$\mathbf{P}_k$PACKING, mostly in the case of bipartite graphs, discussing the graph maximum degree. Namely, we study the case of bipartite graphs of maximum degree 3: first, these problems are **NP**-complete for any $k \geqslant 3$ (and this even if the graph is planar, for $k = 3$); second, there is no **PTAS** for MAX(INDUCED)$\mathbf{P}_k$PACKING or, more precisely, there is a constant $\varepsilon_k > 0$ such that it is **NP**-hard to decide whether a maximum (induced) $\mathbf{P}_k$-packing is of size n or of a size upper bounded by $(1 - \varepsilon_k)n$. On the other hand, all these problems trivially become polynomial-time calculable both in graphs of maximum degree 2 and in forests.

Where these problems are intractable, what about their approximation level? We recall that a given problem is said to be ε-approximable if it admits an algorithm that polynomially calculates on any instance a solution that is at least (if maximizing, at most if minimizing) ε times the optimum value. To our knowledge, there is no specific approximation result for either MINk-PATHPARTITION, or MAXW$\mathbf{P}_k$PACKING, in general graphs. Nevertheless, we can find some approximation results for the k-path partition problem where the objective consists of maximizing the number of edges of the paths that participate in the solution (see [VIS 92] for the general case, [CSA 02] for dense graphs). Concerning MAXW$\mathbf{P}_k$PACKING, using approximation results for the maximum weighted k-packing problem (mainly based on local search techniques), [ARK 98], we can obtain a $(\frac{1}{k-1} - \varepsilon)$-approximation; in particular, MAXW$\mathbf{P}_3$PACKING is $(\frac{1}{2} - \varepsilon)$-approximable.

In the case of complete graphs, MAX$\mathbf{P}_k$P is standard-approximable for any k, [HAS 97]. In particular, MAX$\mathbf{P}_3$P and MAX$\mathbf{P}_4$P are respectively $35/67 - \varepsilon$,

[HAS 06] and $3/4$, [HAS 97] approximable. Note that for $k = 2$, a P_2-partition is a perfect matching and hence, MINP_2P and MAXP_2P are both polynomial-time calculable. The minimum case is trickier: from the fact that $\mathbf{P}_k$PARTITION is **NP**-complete in general graphs, it is **NP**-hard to approximate MIN$\mathbf{P}_k$P within $2^{p(n)}$ for any polynomial p, for any $k \geqslant 3$. Nevertheless, we could expect that the metric instances are constant-approximable, even though no approximation rate (to our knowledge) has been established so far for MINMETRIC$\mathbf{P}_k$P.

Here, we provide new approximation results for MIN3-PATHPARTITION, MAXW-$\mathbf{P}_3$PACKING and $\mathbf{P}_k$P. Concerning the two former problems, we propose a $3/2$-approximation for MIN3-PATHPARTITION in general graphs and a $1/3$ (resp., a $1/2$)-approximation for MAXW$\mathbf{P}_3$PACKING in general (resp., bipartite) graphs of maximum degree 3. But we focus more on $\mathbf{P}_k$P and more specifically on $\mathbf{P}_4$P, by analyzing the performance of a specific algorithm proposed by Hassin and Rubinstein, [HAS 97], under different assumptions on the input. By doing so, we put to the fore the effectiveness of this algorithm by proving that it provides new approximation ratios for both standard and differential measures, for both maximization and minimization versions of the problem. But, before going so far, we briefly recall the basis of approximation theory, introduce some notations and then give this outline of the chapter.

18.1.3. *Theoretical framework, notations and organization*

Consider an instance I of an **NP**-hard optimization problem Π and a polynomial-time algorithm A that calculates feasible solutions for Π. Denote by $\mathrm{apx}_\Pi(I)$ the value of a solution calculated by A on I, by $\mathrm{opt}_\Pi(I)$ the value of an optimal solution and by $\mathrm{wor}_\Pi(I)$ the value of a worst solution (that corresponds to the optimum value when reversing the optimization goal). The quality of A is expressed by means of approximation ratios that somehow compare the approximate value to the optimum one. So far, two measures stand out from other works: the *standard* ratio [AUS 99] (the most widely used) and the *differential* ratio [AUS 80, BEL 95, DEM 96, HAS 01]. The standard ratio is defined by $\rho_\Pi(I, \mathtt{A}) = \mathrm{apx}_\Pi(I)/\mathrm{opt}_\Pi(I)$ if Π is a maximization problem and by $\rho_\Pi(I, \mathtt{A}) = \mathrm{opt}_\Pi(I)/\mathrm{apx}_\Pi(I)$ otherwise, whereas the differential ratio is defined by $\delta_\Pi(I, \mathtt{A})$= $(\mathrm{wor}_\Pi(I) - \mathrm{apx}_\Pi(I))/(\mathrm{wor}_\Pi(I) - \mathrm{opt}_\Pi(I))$. In other words, the standard ratio divides the approximate value by the optimum one, whereas the differential ratio divides the distance from a worst solution to the approximate value by the instance diameter.

Within the worst case analysis framework and given a universal constant $\varepsilon \leqslant 1$ (resp., $\varepsilon \geqslant 1$), an algorithm A is said to be an ε-standard approximation for a maximization (resp., a minimization) problem Π if $\rho_{I,\mathtt{A}_\Pi}(I) \geqslant \varepsilon \ \forall I$ (resp., $\rho_{\mathtt{A}_\Pi}(I) \leqslant \varepsilon$ $\forall I$). With respect to differential approximation, A is said to be ε-differential approximate for Π if $\delta_{\mathtt{A}_\Pi}(I) \geqslant \varepsilon, \ \forall I$, for a universal constant $\varepsilon \leqslant 1$. Equivalently, seeing any solution value as a convex combination of the two values $\mathrm{wor}_\Pi(I)$ and $\mathrm{opt}_\Pi(I)$,

an approximate solution value $\mathrm{apx}_\Pi(I)$ will be ε-differential approximate if for any instance I, $\mathrm{apx}_\Pi(I) \geqslant \varepsilon \times \mathrm{opt}_\Pi(I) + (1-\varepsilon) \times \mathrm{wor}_\Pi(I)$ (for the maximization case; reverse the sense of the inequality when minimizing). For both measures, a given problem Π is said to be constant approximable if there exists a polynomial-time algorithm $\mathtt{A}$ and a universal constant ε such that $\mathtt{A}$ is an ε- approximation for Π. The class of problems that are standard- (resp., differential-) constant-approximable is denoted by **APX** (resp., by **DAPX**). If Π admits a polynomial-time approximation scheme, i.e., a whole algorithm family $(\mathtt{A}_\varepsilon)_{(\varepsilon)}$ such that $\mathtt{A}_\varepsilon$ is ε-approximate for any ε (note that the time-complexity of $\mathtt{A}_\varepsilon$ may be exponential in $1/|1-\varepsilon|$), then Π belongs to the class **PTAS** (resp., **DPTAS**).

The notations that will be used are the usual ones according to graph theory. Moreover, we exclusively work in undirected simple graphs. In this chapter, we often identify a path P of length $k-1$ with $\mathbf{P}_k$, even if P contains a chord. However, when dealing with INDUCED $\mathbf{P}_k$PARTITION, the paths that are considered are chordless. Finally, when no ambiguity occurs on the problem that is concerned, we will omit the reference to Π to denote the values $\mathrm{apx}(I)$, $\mathrm{opt}(I)$ and $\mathrm{wor}(I)$. For a better understanding of what follows, we recall some basic concepts of graph theory: a simple graph $G = (V, E)$ is said to be bipartite (or, equivalently, 2-colorable) if there exists a partition L, R of its vertex set such that E is contained in $L \times R$. A graph is planar if it can be drawn in the plane so that no edges intersect. A path (resp., a cycle) $\Gamma = \{v_{j_1}, \ldots, v_{j_q}\} \subseteq E$ in G of length at least 2 (resp., of length at least 4) is chordless if there is in E no other edge than those of Γ linking two vertices of Γ. G is chordal if none of its cycle of length at least 4 is chordless. G is an interval graph if we can associate with each vertex $v_j \in V$ an interval $[a_j, b_j]$ on the real line such that two intervals $[a_j, b_j]$ and $[a_\ell, b_\ell]$ intersect if and only if the edge $[v_j, v_\ell]$ belongs to E; note that interval graphs are special cases of chordal graphs.

This chapter is organized as follows: the next two sections are dedicated to the study of (INDUCED) $\mathbf{P}_k$PARTITION, MAX(INDUCED)$\mathbf{P}_k$PACKING and MINk-PATHPARTITION. Section 18.2 focuses on the complexity status of those problems in bipartite graphs, whereas section 18.3 proposes some approximation results for MAXW$\mathbf{P}_3$PACKING and MIN3-PATHPARTITION. The fourth section is then dedicated to both standard and differential approximations of $\mathbf{P}_k$P. Section 18.4.1 provides a differential approximation for $\mathbf{P}_k$P while bridging some gap between differential approximation of TSP and differential approximation of $\mathbf{P}_k$P. Finally, section 18.4.2, which constitutes the main part of section 18.4, leads a complete analysis of the approximation level of an algorithm proposed by Hassin and Rubinstein [HAS 97], depending on the approximation measure that is considered and the characteristics of the input weight function.

The two main points of the chapter are, on the one hand, the establishment of new complexity results concerning $\mathbf{P}_k$PARTITION and related problems in bipartite graphs by means of reductions (section 18.2) and, on the other hand, the way the algorithm

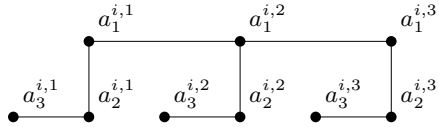


Figure 18.1. *The gadget $H(c_i)$ when c_i is a 3-tuple*

that is addressed in section 18.4.2 appears to be robust, in the sense that this latter provides good quality solutions (the best known so far) irrespective of the version of the problem we deal with and irrespective of the approximation framework within which we estimate the approximate solutions.

18.2. Complexity of $\mathbf{P}_k$PARTITION and related problems in bipartite graphs

18.2.1. *Negative results from the k-dimensional matching problem*

18.2.1.1. *k-dimensional matching problem*

The negative results we present are all based on a transformation from the k-dimensional matching problem, kDM, which is known to be **NP**-complete [GAR 79].

An instance of kDM consists of a subset $\mathcal{C} = \{c_1, \ldots, c_m\} \subseteq X_1 \times \ldots \times X_k$ of k-tuples, where $X_1, \ldots, X_k$ are k pairwise disjoint sets of size n. A matching is a subset $M \subseteq \mathcal{C}$ such that no two elements in M agree in any coordinate and the purpose of kDM is to answer the question: does there exist a perfect matching M on $\mathcal{C}$, i.e., a matching of size n? In its optimization version, the maximum k-dimensional matching problem (MAXkDM) addresses the question of calculating a matching that is of maximal size.

18.2.1.2. *Transforming an instance of kDM into an instance of $\mathbf{P}_k$*PACKING

Let $I = (X_1, \ldots, X_k; \mathcal{C})$ be an instance of kDM, where $|X_q| = n,\ \forall q$ and $|\mathcal{C}| = m$. We denote by X the union of the element sets $X_1, \ldots, X_k$. Furthermore, for each element $e_j \in X$, we denote by d^j its degree, where the degree of an element e_j is defined as the number of k-tuples $c_i \in \mathcal{C}$ that contain e_j. We build an instance $G = (V, E)$ of INDUCED $\mathbf{P}_k$PACKING, where G is a bipartite graph of maximum degree 3, by associating a k-tuple gadget $H(c_i)$ with each k-tuple $c_i \in \mathcal{C}$, an element gadget $H(e_j)$ to each element $e_j \in X$ and then by linking the two gadget families by some edges. Our construction (more precisely, the element gadgets) depends on the parity of k.

1) *The element gadget $H(c_i)$.* For any k-tuple $c_i \in \mathcal{C}$, the gadget $H(c_i)$ consists of a collection $\left\{P^{i,1}, \ldots, P^{i,k}\right\}$ of k vertex-disjoint $\mathbf{P}_k$ with $P^{i,q} = \left\{a_1^{i,q}, \ldots, a_k^{i,q}\right\}$

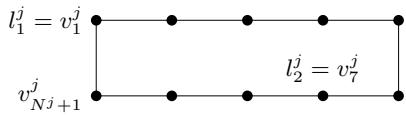


Figure 18.2. *The gadget $H(e_j)$ for $k = 3$ and $d^j = 2$*

for $q = 1, \ldots, k$, plus the edges $[a_1^{i,q}, a_1^{i,q+1}]$ for $q = 1$ to $k-1$. Hence, $H(c_i)$ contains the k initial paths $P^{i,1}, \ldots, P^{i,k}$, plus the additional path $\left\{a_1^{i,1}, \ldots, a_1^{i,k}\right\}$. Figure 18.1 proposes an illustration of the k-tuple gadget when $k = 3$.

2) *The element gadget $H(e_j)$.* Let $e_j \in X$ be an element, with degree d^j. We distinguish two cases according to the parity of k:

– Odd values of k. $H(e_j)$ is defined as a cycle $\left\{v_1^j, \ldots, v_{N^j+1}^j, v_1^j\right\}$ on $N^j + 1$ vertices, where $N^j = k(2d^j - 1)$. Moreover, for $p = 1$ to d^j, we denote by l_p^j the vertex of index $2k(p-1)+1$. Thus, the element gadget is a cycle on a number of vertices that is a multiple of k plus 1, with d^j remarkable vertices l_p^j that will be linked to the k-tuple gadgets.

– Even values of k. In this case, N^j is also even and thus, a cycle on N^j+1 vertices may not be part of a bipartite graph. In order to fix that problem, we define $H(e_j)$ as a cycle $\left\{v_1^j, \ldots, v_{N^j}^j, v_1^j\right\}$ on N^j vertices, plus an additional edge $[v_{N^j}^j, v_{N^j+1}^j]$. The special vertices l_p^j are still defined as $l_p^j = v_{2k(p-1)+1}^j$ for $p = 1, \cdots, d^j$ (note that $l_{d^j}^j$ never matches $v_{N^j}^j$). Figures 18.2 and 18.3 illustrate $H(e_j)$ for the couple of values $k = 3,\ d^j = 2$ and $k = 4,\ d^j = 2$, respectively.

3) *Linking element gadgets to k-tuple gadgets.* For any couple (e_j, c_i) such that e_j is the value of c_i on the q-th coordinate, the two gadgets $H(c_i)$ and $H(e_j)$ are connected using one of the edges $[a_2^{i,q}, l_{p_i}^j],\ p_i \in \{1, \ldots, d^j\}$. The vertices $l_{p_i}^j$ that will be linked to a given gadget $H(c_i)$ must be chosen so that each vertex l_p^j from any gadget $H(e_j)$ will be connected to exactly one gadget $H(c_i)$.

The described construction obviously leads to a graph $G = (V, E)$ that is bipartite, of maximum degree 3 and such that each of the subgraphs $\mathbf{P}_k$ it contains is chordless. Its number of vertices is $|V| = 3k^2m + (1-k)kn$: consider, on the one hand, that each gadget $H(c_i)$ is a graph on k^2 vertices and, on the other hand, that $\sum_{j=1}^{kn} d^j = km$ (without loss of generality, we may assume that each element e_j appears at least once in $\mathcal{C}$).

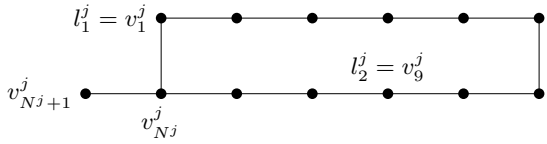


Figure 18.3. *The gadget $H(e_j)$ for $k = 4$ and $d^j = 2$*

18.2.1.3. *Analyzing the obtained instance of* $\mathbf{P}_k$PACKING

Let us define on G some remarkable $\mathbf{P}_k$ packings on the vertex subsets $V(H(c_i))$ and $V(H(e_j))$:

$\mathbf{P}_k$ packings on $V(H(c_i))$, for $i = 1, \ldots, m$:

$$\begin{cases} \mathcal{P}^i = \cup_{q=1}^{k} P^{i,q} \cup \left\{a_1^{i,1}, a_1^{i,2}, \ldots, a_1^{i,k}\right\} & \text{with } P^{i,q} = \left\{a_k^{i,q}, \ldots, a_2^{i,q}, l_{i,q}\right\} \ \forall q \\ \mathcal{Q}^i = \cup_{q=1}^{k} Q^{i,q} & \text{with } Q^{i,q} = \left\{a_k^{i,q}, \ldots, a_2^{i,q}, a_1^{i,q}\right\} \forall q \end{cases}$$

(where $l_{i,q}$ denotes the vertex from some $H(e_j)$ that is linked to $a_2^{i,q}$)

$\mathbf{P}_k$ packings on $V(H(e_j))$, for $j = 1, \ldots, kn$:

$\forall p = 1, \ldots, d^j$, $\mathcal{P}_p^j$ is defined as the only possible $\mathbf{P}_k$ partition of $V(H(e_j))\backslash\{l_p^j\}$

Note that these collections are of size $|\mathcal{P}^i| = k + 1 \ \forall i$, $|\mathcal{Q}^i| = k \ \forall i$ and $|\mathcal{P}_p^j| = 2d^j - 1 \ \forall j \ \forall p \in \{1, \ldots, d^j\}$. With the help of these packings, we now put to the fore three properties that will be key to our further argumentation.

PROPERTY 18.1.–

(i) For any i, $\mathcal{P}^i$ and $\mathcal{Q}^i$ are the only two possible $\mathbf{P}_k$ partitions of $V(H(c_i))$.

(ii) Within a $\mathbf{P}_k$ partition of V and for any $j = 1, \ldots, kn$, the collections $\mathcal{P}_1^j, \ldots, \mathcal{P}_{d^j}^j$ are the only possible $\mathbf{P}_k$ partitions of $V(H(e_j))$.

(iii) Let $\mathcal{P}^*$ be a maximum $\mathbf{P}_k$ packing on G; we can always assume the following:

$(iii.a)$ for any i, $\mathcal{P}^*$ contains either the packing $\mathcal{P}^i$, or the packing $\mathcal{Q}^i$;

$(iii.b)$ for any j, $\mathcal{P}^*$ contains one of the packings $\mathcal{P}_p^j$, for some p.

Proof. For the sake of simplicity, we assume that k is odd, even though the arguments also hold for even values of k.

For (i). Quite immediate, from the observation that a given vertex $a_k^{i,q}$ may only be covered by either $P^{i,q}$ or $Q^{i,q}$.

For (ii). Let $\mathcal{P}$ be a $\mathbf{P}_k$ partition of V and consider an element e_j; since $H(e_j)$ contains $N^j = k(2d^j - 1) + 1$ vertices, at least one edge e of some P_ℓ in $\mathcal{P}$ links $H(e_j)$ to a given $H(c_i)$, using an l_p^j vertex. We deduce from the previous point that P_ℓ is some $P^{i,q}$ path and thus that l_p^j is the only vertex of P_ℓ that intersects $H(e_j)$. Consider now any two vertices l_p^j and $l_{p'}^j$, $p < p'$, from $H(e_j)$; the $2k(p' - p) - 1$ vertices that separate l_p^j and $l_{p'}^j$ might not be covered by any collection of $\mathbf{P}_k$. Hence, exactly one l_p^j vertex of $H(e_j)$ is covered by some $P^{i,q}$ and thus, $\mathcal{P}$ contains the corresponding $\mathbf{P}_k$ packing $\mathcal{P}_p^j$.

For $(iii.a)$. Any maximal size $\mathbf{P}_k$ packing must use (at least) one of the two vertices $a_1^{i,q}$ and $l_{i,q}$, for any couple (i, q), where $l_{i,q}$ denotes the vertex from some $H(e_j)$ that is linked to $a_2^{i,q}$. Suppose the reverse, for some (i, q): then, none of the vertices $l_{i,q}, a_1^{i,q}, a_2^{i,q}, \ldots, a_k^{i,q}$ may be part of a path from $\mathcal{P}^*$ and thus, $P^{i,q}$ or $Q^{i,q}$ could be added to $\mathcal{P}^*$, which would contradict the optimality of $\mathcal{P}^*$. If the edge $[a_1^{i,q}, a_2^{i,q}]$ (resp., $[a_2^{i,q}, l_{i,q}]$ and not $[a_1^{i,q}, a_2^{i,q}]$) is used by some path $P \in \mathcal{P}^*$, then P can be replaced in $\mathcal{P}^*$ by the path $Q^{i,q}$ (resp., by $P^{i,q}$). If none of the edges $[a_1^{i,q}, a_2^{i,q}]$ and $[a_2^{i,q}, l_{i,q}]$ are used by $\mathcal{P}^*$, replace by $P^{i,q}$ (resp., by $Q^{i,q}$) the path from $\mathcal{P}^*$ that uses $l_{i,q}$ (resp., $a_1^{i,q}$ and not $l_{i,q}$). At that point, the collection $\mathcal{P}^*$ contains for any k-tuple c_i at least k paths $P^{i,q}$ and $Q^{i,q}$ (one for each coordinate $q = 1, \ldots, k$). Now, each time $\mathcal{P}^*$ does not contain the packing $\mathcal{P}^i$, we replace these paths by the whole collection $\mathcal{Q}^i$.

For $(iii.b)$. Assume the reverse, for some element e_j; that means that at least 2 vertices $l_{p_i}^j$ and $l_{p_{i'}}^j$ of $H(e_j)$ are used in $\mathcal{P}^*$ by paths $P^{i,q}$ and $P^{i',q'}$, with $p_i < p_{i'}$ (or $\mathcal{P}^*$ would not be of maximal size). Choose two such consecutive vertices, in the sense that $\mathcal{P}^*$ does not use any of the paths $P^{i'',q''}$ for $l_{p_{i''}}^j$ such that $p_i < p_{i''} < p_{i'}$. Since there are $2k(p_{i'} - p_i) - 1$ vertices of $H(e_j)$ between $l_{p_i}^j$ and $l_{p_{i'}}^j$, we can replace $P^{i,q}$, $P^{i',q'}$ and the paths of $\mathcal{P}^*$ between vertices $l_{p_i}^j$ and $l_{p_{i'}}^j$ by $P^{i,q}$ and $2(p_{i'} - p_i)$ paths using vertices between $l_{p_i}^j$ and $l_{p_{i'}}^j$, plus $l_{p_{i'}}^j$. Observe that, in such a case, the packing $\mathcal{P}^{i'}$ will be replaced in $\mathcal{P}^*$ by the packing $\mathcal{Q}^{i'}$, according to the previous property. By repeating this procedure, we obtain a maximal size $\mathbf{P}_k$ packing that fulfills the requirements of items $(iii.a)$ and $(iii.b)$. ∎

18.2.1.4. *NP-completeness and APX-hardness*

The following theorem holds.

THEOREM 18.1.– $\mathbf{P}_k$PARTITION *and* INDUCED $\mathbf{P}_k$PARTITION *are* ***NP****-complete in bipartite graphs of maximum degree 3, for any* $k \geqslant 3$. *As a consequence,* MAX(INDUCED)$\mathbf{P}_k$PACKING *and* MINk-PATHPARTITION *are* ***NP****-hard in bipartite graphs with maximum degree 3, for any* $k \geqslant 3$.

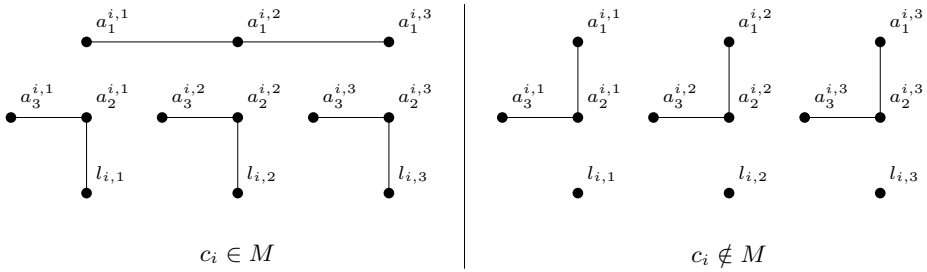


Figure 18.4. *A vertex partition of a $H(c_i)$ gadget into 2-edge paths*

Proof. Let $I = (X_1, \ldots, X_k; \mathcal{C})$ and $G = (V, E)$ be an instance of kDM and the graph produced by construction described is section 18.2.1.2, respectively. First, we recall that any path of length $k-1$ in G is chordless; thus, the result holds for both $\mathbf{P}_k$PARTITION and INDUCED $\mathbf{P}_k$PARTITION. We claim that there exists a perfect matching $M \subseteq \mathcal{C}$ on I *iff* there exists a partition $\mathcal{P}$ of G into $\mathbf{P}_k$.

Let $\mathcal{P}$ be such a partition on G; from Property 18.1 item (i), we know that each gadget $H(c_i)$ is covered by either $\mathcal{P}^i$ or $\mathcal{Q}^i$. Moreover, Property 18.1 item (ii) indicates that every gadget $H(e_j)$ is covered by some $\mathcal{P}_p^j$ collection; those two facts ensure that exactly one $H(c_i)$ gadget for some k-tuple that contains e_j is covered by a $\mathcal{P}^i$ collection and therefore the set $M = \left\{c_i \mid \mathcal{P}^i \subseteq \mathcal{P}\right\}$ defines a perfect matching on I.

Conversely, let M be a perfect matching on $\mathcal{C}$; we build a packing $\mathcal{P}$ by applying the following rule: if a given element c_i belongs to M, then use $\mathcal{P}^i$ to cover $H(c_i)$ and use $\mathcal{Q}^i$ otherwise (Figure 18.4 illustrates this construction for 3DM). Since M is a perfect matching, exactly one vertex l_p^j per gadget $H(e_j)$ is covered by some $P^{i,q}$. Thus, on a given cycle $H(e_j)$, the $N^j = k(2d^j - 1)$ vertices that remain uncovered can be covered using the corresponding collection $\mathcal{P}_p^j$. ∎

Thus, the construction is a Karp reduction and, from the **NP**-completeness of kDM, [GAR 79], we deduce the **NP**-completeness of (INDUCED) $\mathbf{P}_k$PARTITION in bipartite graphs of maximum degree 3. However, by a more accurate observation, we actually may obtain a stronger result, for $k = 3$; namely, (INDUCED) $\mathbf{P}_3$PARTITION **NP**-completeness still holds when restricting ourselves to planar instances. Indeed, on the one hand, the restriction PLANAR 3DM of 3-dimensional matching to planar instances is still **NP**-complete [DYE 86]; on the other hand, if the initial instance I of kDM is planar, then the graph G is also planar for an appropriate choice of the linking edges $[a_2^{i,q}, l_{i,q}]$.

THEOREM 18.2.– $\mathbf{P}_3$PARTITION *and* INDUCED $\mathbf{P}_3$PARTITION *are* ***NP****-complete in planar bipartite graphs with maximum degree 3. As a consequence,* MAX(INDUCED)$\mathbf{P}_3$PACKING *and* MIN3-PATHPARTITION *are* ***NP****-hard in planar bipartite graphs with maximum degree 3.*

If we now turn to the optimization problems, we can observe that the construction described in section 18.2.1.2 also makes it possible to establish an **APX**-hardness result for the maximization problems MAX$\mathbf{P}_k$PACKING and MAX (INDUCED) $\mathbf{P}_k$PACKING. We consider the optimization version of kDM, denoted by MAXkDM, and the following inapproximability result: for any $k \geqslant 3$, there is a constant $\varepsilon'_k > 0$ such that $\forall I = (X_1, \ldots, X_k; \mathcal{C})$ instance of kDM with $|X_1| = \cdots = |X_k| = n$ and it is **NP**-hard to decide between $\mathrm{opt}(I) = n$ and $\mathrm{opt}(I) \leqslant (1 - \varepsilon'_k)n$, where $\mathrm{opt}(I)$ is the value of a maximum matching on $\mathcal{C}$. This result also holds if we restrict ourselves to instances with bounded degree, i.e., to instances I satisfying: $\forall j = 1, \ldots, kn,\ d^j \leqslant f(k)$, where $f(k)$ is a constant; we refer to [PET 94] for $k = 3$ (where the result is proved with $f(3) = 3$), to [KAR 06] for other values of k.

THEOREM 18.3.– *For any $k \geqslant 3$, there is a constant $\varepsilon_k > 0$, such that $\forall G = (V, E)$ instance of* MAX(INDUCED)$\mathbf{P}_k$PACKING *where G is a bipartite graph of maximum degree 3 and it is* ***NP****-hard to decide between $\mathrm{opt}(G) = \frac{|V|}{k}$ and $\mathrm{opt}(G) \leqslant (1 - \varepsilon_k)\frac{|V|}{k}$, where $\mathrm{opt}(G)$ is the value of a maximum (induced) $\mathbf{P}_k$-Packing on G.*

Proof. Let $I = (X_1, \ldots, X_k; \mathcal{C})$ be an instance of kDM, with $|X_q| = n\ \forall q$ and $|\mathcal{C}| = m$, such that the degree d^j of any element e_j is bounded above by $f(k)$. Consider the graph $G = (V, E)$ produced by the construction described in section 18.2.1.2; we recall that $|V| = 3k^2m - k^2n + kn$. Let (M^*, $\mathcal{P}^*$) be a couple of optimal solutions on I and G, with values $\mathrm{opt}(I)$ and $\mathrm{opt}(G)$, respectively. From Property 18.1 items $(iii.a)$ and $(iii.b)$, we can assume that $\mathcal{P}^*$ satisfies the following:

– for any i, $\mathcal{P}^*$ contains either the packing $\mathcal{P}^i$, or the packing $\mathcal{Q}^i$;

– for any j, $\mathcal{P}^*$ contains one of the packings $\mathcal{P}^j_1, \ldots, \mathcal{P}^j_{d^j}$.

Hence, the set $M = \{c_i \in \mathcal{C} :\ \mathcal{P}^i \in \mathcal{P}^*\}$ of k-tuples c_i such that $\mathcal{P}^*$ contains $\mathcal{P}^i$ defines a matching on I; moreover, the value $\mathrm{opt}(G)$ of $\mathcal{P}^*$ can be expressed as:

$$\mathrm{opt}(G) = (km + |M|) + \sum_{j=1}^{kn} \left(2d^j - 1\right) = 3km - kn + |M|$$

From $|M| \leqslant |M^*|$, we then deduce: $\mathrm{opt}(G) \leqslant \mathrm{opt}(I) + 3km - kn$.

If $\mathrm{opt}(I) = n$: we know from Theorem 18.1 that I has a perfect matching if and only if G admits a $\mathbf{P}_k$Partition, i.e., $\mathrm{opt}(I) = n$ *iif* $\mathrm{opt}(G) = \frac{|V|}{k} = 3km - kn + n$. Suppose now that $\mathrm{opt}(I) \leqslant (1-\varepsilon'_k)n$. Then, necessarily: $\mathrm{opt}(G) \leqslant 3km - kn + (1 -$

$\varepsilon'_k)n = (3km - kn + n) - \varepsilon'_k n$. By setting $\varepsilon_k = \frac{n}{3km-kn+n}\varepsilon'_k$, we obtain $\mathrm{opt}(G) \leqslant (1-\varepsilon_k)(3km-kn+n)$. Finally, since $d^j \leqslant f(k)$, we deduce that $km \leqslant kf(k)n$ and then, that $\varepsilon_k \geqslant \frac{1}{3f(k)k-k+1}\varepsilon'_k = \mathcal{O}(1)$. In conclusion, deciding between $\mathrm{opt}(G) = |V|/n$ and $\mathrm{opt}(G) \leqslant (1-\varepsilon_k)|V|/n$ (or $\mathrm{opt}(G) \leqslant (1 - \frac{1}{3f(k)k-k+1}\varepsilon'_k)|V|/n)$) on G would make it possible to decide between $\mathrm{opt}(I) = n$ and $\mathrm{opt}(I) \leqslant (1-\varepsilon'_k)n$ on I. ∎

18.2.2. *Positive results from the maximum independent set problem*

If we decrease the maximum degree of the graph down to 2, we can easily prove that $\mathbf{P}_k$PARTITION, INDUCED $\mathbf{P}_k$PARTITION, MAX$\mathbf{P}_k$PACKING and MINk-PATH-PARTITION are polynomial-time calculable. The same fact holds for MAXW$\mathbf{P}_k$PACKING (what remains true in forests), although it is a little bit complicated: the proof consists of a reduction from MAXW$\mathbf{P}_k$PACKING in graphs with maximum degree 2 (resp., in a forest) to the problem of calculating a maximum weight independent set in an interval (resp., a chordal) graph, which is known to be polynomial, [FRA 76].

PROPOSITION 18.1.– MAXW$\mathbf{P}_k$PACKING *is polynomial in graphs with maximum degree 2 and in forests, for any* $k \geqslant 3$.

Proof. Let $I = (G, w)$ be an instance of MAXW$\mathbf{P}_k$PACKING where $G = (V, E)$ is a graph with maximum degree 2. Hence, G is a collection of disjoint paths or cycles and thus, each connected component may be separately solved. Moreover, without loss of generality, we may assume that each connected component G^ℓ of G is a path. Otherwise, a given cycle $G^\ell = \{v_1, \ldots, v_{N_\ell}, v_1\}$ might be solved by picking the best solution among the solutions calculated on the k instances $G^\ell \setminus \{[v_1, v_2]\}, \ldots, G^\ell \setminus \{[v_k, v_{k+1}]\}$.

Thus, let $G^\ell = \left\{v^\ell_1, \ldots, v^\ell_{N^\ell}\right\}$ be such a path; we build the instance (H^ℓ, w^ℓ) of MAXWIS where the vertex set of H^ℓ corresponds to the paths of length $k-1$ in G^ℓ: a vertex v is associated with each path P_v, with weight $w^\ell(v) = w(P_v)$. Moreover, two vertices $u \neq v$ are linked in H^ℓ if and only if the corresponding paths P_u and P_v share at least one common vertex in the initial graph.

We deduce that the set of independent sets in H^ℓ corresponds to the set of $\mathbf{P}_k$ in G^ℓ. Observe that H^ℓ is an interval graph (even a unit interval graph), since each path can be viewed as an interval of the line $\{1, \cdots, N^\ell\}$; hence, H^ℓ is chordal. If G is a forest, then any of the graphs H^ℓ that correspond to a tree of G are chordal graphs. ∎

18.3. Approximating MAXW$\mathbf{P}_3$PACKING and MIN3-PATHPARTITION

We present some approximation results for MAXW$\mathbf{P}_3$PACKING and MIN3-PATH-PARTITION, that are mainly based on matching and spanning tree heuristics.

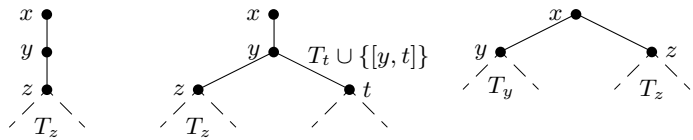
18.3.1. MAXWP$_3$PACKING *in graphs of maximum degree 3*

For this problem, the best approximate algorithm known so far provides a ratio of $(\frac{1}{2}-\varepsilon)$, within high (but polynomial) time complexity. This algorithm is deduced from that proposed in [ARK 98] to approximate the weighted k-set packing problem for sets of size 3. Furthermore, a simple greedy $1/k$-approximation of MAXW$\mathbf{P}_k$PACKING consists of iteratively picking a path of length $k-1$ that is of maximum weight.

For $k = 3$ and in graphs of maximum degree 3, the time complexity of this algorithm is between $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^2)$ (depending on the encoding structure). Actually, in such graphs, we may reach a $1/3$-approximate solution, even in time $\mathcal{O}(\alpha(n,m)n)$, where α is the inverse Ackerman's function and $m \leqslant 3n/2$.

THEOREM 18.4.– MAXW$\mathbf{P}_3$PACKING *is 1/3 approximable within* $\mathcal{O}(\alpha(n,3n/2)n)$ *time complexity in graphs of maximum degree 3; this ratio is tight for the algorithm we analyze.*

Proof. The argument uses the following observation: for any spanning tree of maximum degree 3 containing at least 3 vertices, we can build a cover of its edge set into 3 packings of $\mathbf{P}_3$ within linear time. Hence, by calculating a maximum-weight spanning tree $T = (V, E_T)$ on G in $\mathcal{O}(\alpha(n,3n/2)n)$ time [CHA 00] and by picking the best $\mathbf{P}_3$-packing among the cover, we obtain a 1/3 approximate solution within an overall time complexity dominated by $\mathcal{O}(\alpha(n,3n/2)n)$.



Steps 3.1 and 3.2 Steps 3.1, 3.2 and 3.3.1 Steps 4.1 to 4.3

Figure 18.5. *The main configurations of the algorithm* `SubProcess`

The construction of the 3 packings $\mathcal{P}^1, \mathcal{P}^2, \mathcal{P}^3$ is done in the following way: we start with three empty collections $\mathcal{P}^1, \mathcal{P}^2, \mathcal{P}^3$ and a tree T rooted at r; according to the degree of r and to the degree of its children, we add some $\mathbf{P}_3$ path P that contains r to the packing $\mathcal{P}^1$, remove the edges of P from T and then recursively repeat this process on the remaining subtrees, alternatively invoking $\mathcal{P}^2$ and $\mathcal{P}^1$. This procedure is formally described in the algorithms `SubProcess` (the recursive process) and `Tree-P`$_3$`PackingCover` (the whole process).

Algorithm `Tree-P`$_3$`PackingCover` makes an initial call to `SubProcess`, on the whole tree T, rooted on a vertex r that is of degree at most 2 in T. The stopping

criteria of the recursive procedure `SubProcess` are the following: the current tree has no edge (then stop), or the current tree is a lonely edge $[x, y]$; then add $\{r_x, x, y\}$ to $\mathcal{P}^3$, where r_x denotes the father of x in T. Concerning the three main configurations of `SubProcess`, they are illustrated in Figure 18.5, where T_v denotes the subtree of T rooted at v; the edges in rigid lines represent the path that is added to the current packing, and the subtrees that are invoked by the recursive calls are indicated.

`Tree-P`$_3$`PackingCover`

Input: $T = (V_T, E_T)$ spanning tree of maximum degree 3 containing at least 3 vertices and rooted at r such that $d_T(r) \leqslant 2$.

1 Set $\mathcal{P}^1 = \mathcal{P}^2 = \mathcal{P}^3 = \{\}$;

2 Call `SubProcess`(T_r,$\mathcal{P}^1$,$\mathcal{P}^2$,$\mathcal{P}^3$,1);

3 `Repair`($\mathcal{P}^1, \mathcal{P}^2, \mathcal{P}^3$);

Output ($\mathcal{P}^1$, $\mathcal{P}^2$, $\mathcal{P}^3$).

`SubProcess`(T_x, $\mathcal{P}^1$,$\mathcal{P}^2$,$\mathcal{P}^3$, i)

1 If $E_{T_x} = \{\}$ then **exit**;

Pick y a child of x in T_x;

2 If $E_{T_x} = \{\{x, y\}\}$

Pick r_x the father of x in T_r;

2.1 $\mathcal{P}^3 \longleftarrow \mathcal{P}^3 \cup \{\{r_x, x, y\}\}$; **exit**;

3 If x is of degree 1 in T_x

Pick z a child of y in T_x;

3.1 $\mathcal{P}^i \longleftarrow \mathcal{P}^i \cup \{\{x, y, z\}\}$;

3.2 Call `SubProcess`(T_z, $\mathcal{P}^1$,$\mathcal{P}^2$,$\mathcal{P}^3$,3-i);

3.3 If y is of degree 3 in T_x

Pick t the second child of y in T_x;

3.3.1 Call `SubProcess`($\{\{y, t\}\} \cup T_t$, $\mathcal{P}^1$,$\mathcal{P}^2$,$\mathcal{P}^3$, 3-i);

4 Else If x is of degree 2 in T_x

Pick z the second child of x in T_x;

4.1 $\mathcal{P}^i \longleftarrow \mathcal{P}^i \cup \{\{y, x, z\}\}$;

4.2 Call `SubProcess`(T_y, $\mathcal{P}^1$,$\mathcal{P}^2$,$\mathcal{P}^3$,3-i);

4.3 Call `SubProcess`(T_z, $\mathcal{P}^1$,$\mathcal{P}^2$,$\mathcal{P}^3$,3-i);

At the end of the initial call to `SubProcess` (i.e., when the step 2 of `Tree-P`$_3$`PackingCover` has been achieved), $\mathcal{P}^1$ and $\mathcal{P}^2$ both are packings: we can easily see that the paths that are added to $\mathcal{P}^i$ (where $i = 1$ or $i = 2$) at a given time t and those that are added again to $\mathcal{P}^i$ at time $t + 2$ do not share any common vertex. On the other hand, $\mathcal{P}^3$ might not be a packing. Let $\{r_x, x, y\}$ and $\{r_{x'}, x', y'\}$ be two paths from $\mathcal{P}^3$ such that $\{r_x, x, y\} \cap \{r_{x'}, x', y'\} \neq \{\,\}$; then, either $r_x = r_{x'}$, or $r_x = x'$. If the first case occurs, $\{x, r_x, x'\}$ has been added to $\mathcal{P}^i$ (for $i = 1$ or $i = 2$), then set: $\mathcal{P}^i = \mathcal{P}^i \backslash \{\{x, r_x, x'\}\} \cup \{\{r_x, x, y\}\}$ and $\mathcal{P}^3 = \mathcal{P}^3 \backslash \{\{r_x, x, y\}\}$. Otherwise, $r_{x'}$ is the father of r_x in T_r and we have $\{r_{x'}, r_x, x\} \in \mathcal{P}^i$ (for $i = 1$ or $i = 2$); then set: $\mathcal{P}^i = \mathcal{P}^i \backslash \{\{r_{x'}, r_x, x\}\} \cup \{\{r_{x'}, x', y'\}\}$ and $\mathcal{P}^3 = \mathcal{P}^3 \backslash \{\{r_{x'}, x', y'\}\}$. These repairing operations are made by the algorithm `Repair`, during step 3 of `Tree-P`$_3$`PackingCover`.

Figure 18.6 provides two examples of the construction of $\mathcal{P}^1$, $\mathcal{P}^2$ and $\mathcal{P}^3$. The overall time complexity of `Tree-P`$_3$`PackingCover` is linear: first, the number of recursive calls to `SubProcess` may not exceed $2/3n$ and second, $|\mathcal{P}^3|$ is at most $\mathcal{O}(\log n)$.

`Repair(`$\mathcal{P}^1$`,`$\mathcal{P}^2$`,`$\mathcal{P}^3$`)`

1 For any $(P = \{r_x, x, y\} \neq P' = \{r_{x'}, x', y'\}) \in \mathcal{P}^3$ s.t. $r_x = r_{x'}$
  Set $i \in \{1, 2\}$ s.t. $\{x, r_x, x'\} \in \mathcal{P}^i$;
  1.1 $\mathcal{P}^i \longleftarrow \mathcal{P}^i \backslash \{\{x, r_x, x'\}\} \cup \{\{r_x, x, y\}\}$; $\mathcal{P}^3 \longleftarrow \mathcal{P}^3 \backslash \{\{r_x, x, y\}\}$;

2 For any $(P = \{r_x, x, y\} \neq P' = \{r_{x'}, x', y'\}) \in \mathcal{P}^3$ s.t. $r_x = x'$
  Set $i \in \{1, 2\}$ s.t. $\{r_{x'}, r_x, x\} \in \mathcal{P}^i$;
  2.1 $\mathcal{P}^i \longleftarrow \mathcal{P}^i \backslash \{\{r_{x'}, r_x, x\}\} \cup \{\{r_{x'}, x', y'\}\}$; $\mathcal{P}^3 \longleftarrow \mathcal{P}^3 \backslash \{\{r_{x'}, x', y'\}\}$;

Output ($\mathcal{P}^1$, $\mathcal{P}^2$, $\mathcal{P}^3$).

We can now deduce an approximate algorithm `MaxW`$\mathbf{P}_3$`Packing` that consists of calculating a $\mathbf{P}_3$-packing cover $(\mathcal{P}^1, \mathcal{P}^2, \mathcal{P}^3)$ of a maximum spanning tree of G, and then picking the best collection among ($\mathcal{P}^1$, $\mathcal{P}^2$, $\mathcal{P}^3$). This algorithm provides a 1/3-approximation within $\mathcal{O}(\alpha(n, 3n/2)n)$ time complexity (the overall complexity of the algorithm is dominated by that of calculating the initial spanning tree). Concerning the approximation level, consider that the weight $w(T)$ of a maximum spanning tree T is at least the weight of an optimal $\mathbf{P}_3$-packing, since any $\mathbf{P}_3$ packing can be completed into a spanning tree (if the input graph is connected). Then the result is trivial (let $\mathcal{P}^*$ denote an optimal solution):

$$w(\mathcal{P}) \geqslant 1/3 \left(w(\mathcal{P}^1) + w(\mathcal{P}^2) + w(\mathcal{P}^3)\right) \geqslant 1/3 w(T) \geqslant 1/3 w(\mathcal{P}^*)$$

The proof of tightness is omitted. ∎

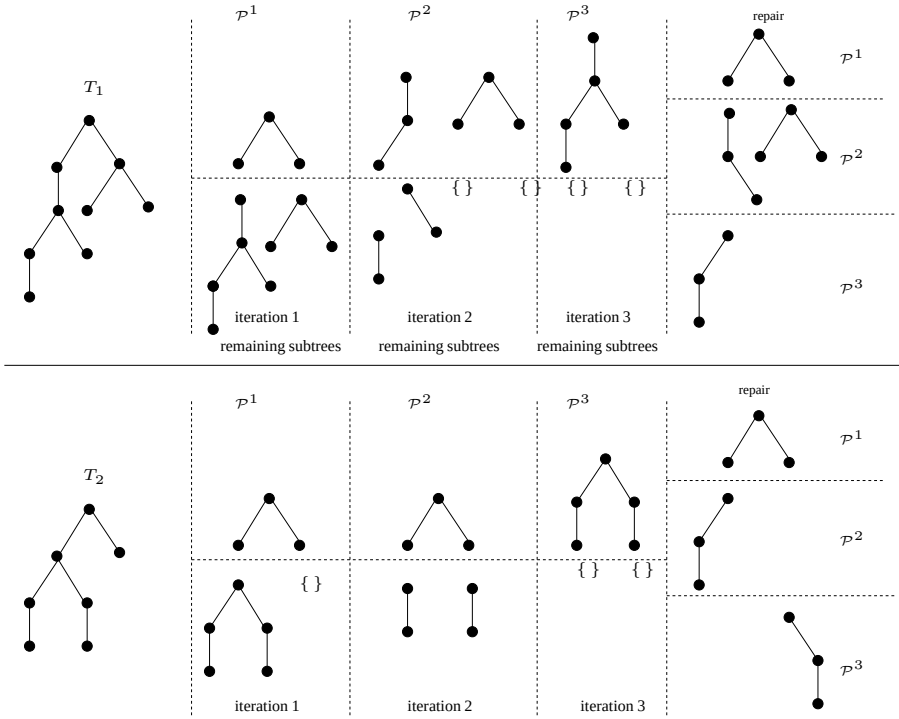


Figure 18.6. *Two examples of the construction of the 3 packings $\mathcal{P}^i$ for $i = 1, 2, 3$*

18.3.2. MAXWP$_3$PACKING *in bipartite graphs of maximum degree 3*

If we restrict ourselves to bipartite graphs, we slightly improve the ratio of $\frac{1}{2} - \varepsilon$, [ARK 98] up to $\frac{1}{2}$. We then show that, in the unweighted case, this result holds without any constraint on the graph maximum degree. The key idea here is to transform the problem of finding a $\mathbf{P}_3$Packing in the initial bipartite graph $G = (L, R; E)$ into the problem of finding a maximum matching in two graphs G_L and G_R, where G_L (resp., G_R) contains the representative edge of the $\mathbf{P}_3$ of the initial graph with their two extremities in L (resp., in R). Formally, from an instance $I = (G, w)$ of MAXW$\mathbf{P}_3$PACKING, where $G = (L, R; E)$ is a bipartite graph of maximum degree 3, we build two weighted graphs (G_L, w_L) and (G_R, w_R), where $G_L = (L, E_L)$ and $G_R = (R, E_R)$. Two vertices $x \neq y$ from L are linked in G_L if and only if there exists in G a path $P_{x,y}$ of length 2 from x to y: $[x, y] \in E_L$ if and only if $\exists z \in R$ s.t. $[x, z], [z, y] \in E$. The weight $w_L(x, y)$ is defined as $w_L(x, y) = \max\{w(x, z) + w(z, y) | [x, z], [z, y] \in E\}$. The weighted graph (G_R, w_R) is defined

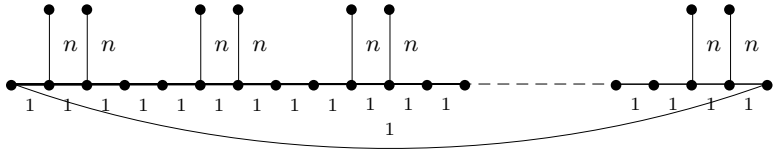


Figure 18.7. *Tightness of* `Weighted` P_3*-*`Packing` *analysis.*

by considering R instead of L. If G is of maximum degree 3, then the following fact holds:

PROPERTY 18.2.– From any matching M on G_L (resp., on G_R), we can deduce a $\mathbf{P}_3$ packing $\mathcal{P}_M$ of weight $w(\mathcal{P}_M) = w_L(M)$ (resp., $w(\mathcal{P}_M) = w_R(M)$), where G is of degree at most 3.

Proof. Let M be a matching on G_L and $\mathcal{P}_M$ the corresponding $\mathbf{P}_3$ collection on G. Suppose that two paths $P_{x,y} \neq P_{x',y'} \in \mathcal{P}_M$ share a common vertex t. Because M is a matching, we have $\{x, y\} \cap \{x', y'\} = \{\}$; hence, the vertex t belongs to R and is the internal vertex of both $P_{x',y'}$ and $P_{x',y'}$, which contradicts the assumption on the graph maximum degree. ∎

In light of this fact, we propose the algorithm `Weighted` P_3`-Packing` that consists of calculating two maximum matchings on G_L and G_R, and then picking the best corresponding packing in G. The time complexity of this algorithm is mainly the time complexity of calculating a maximum weight matching in graphs of maximum degree 9, i.e., $\mathcal{O}(|V|^2 \log |V|)$, [LOV 86].

`Weighted` P_3`-Packing`

1 Build the weighted graphs (G_L, w_L) and (G_R, w_R);

2 Calculate a maximum weight matching M_L^* (resp., M_R^*) on (G_L, w_L) (resp., on (G_R, w_R));

3 Deduce from M_L^* (resp., from M_R^*) a $\mathbf{P}_3$ packing $\mathcal{P}_L$ (resp., $\mathcal{P}_R$) according to Property 18.2;

4 Output the best packing $\mathcal{P}$ among $\mathcal{P}_L$ and $\mathcal{P}_R$.

THEOREM 18.5.– *`Weighted` P_3`-Packing` provides a $1/2$-approximation for* MAXW$\mathbf{P}_3$PACKING *in bipartite graphs with maximum degree 3 and this ratio is tight.*

Proof. Let $\mathcal{P}^*$ be an optimum $\mathbf{P}_3$-packing on $I = (G, w)$ we denote by $\mathcal{P}_L^*$ (resp., by $\mathcal{P}_R^*$) the paths of $\mathcal{P}^*$ whose two endpoints belong to L (resp., to R); thus, $\mathrm{opt}(I) =$

$w(\mathcal{P}_L^*) + w(\mathcal{P}_L^*)$. For any path $P = P_{x,y} \in \mathcal{P}_L^*$, $[x, y]$ is an edge from E_L, of weight $w_L(x, y) \geqslant w(P_{x,y})$. Hence, $M_L = \{[x, y] | P_{x,y} \in \mathcal{P}_L^*\}$ is a matching on G_L that satisfies:

$$w_L(M_L) \geqslant w(\mathcal{P}_L^*) \qquad [18.1]$$

Moreover, since M_L^* is a maximum weight matching on G_L, we have $w_L(M_L) \leqslant w_L(M_L^*)$. Thus, using inequality [18.1] and Property 18.2 (and by applying the same arguments on G_R), we deduce:

$$w(\mathcal{P}_L) \geqslant w(\mathcal{P}_L^*),\ w(\mathcal{P}_R) \geqslant w(\mathcal{P}_R^*) \qquad [18.2]$$

Finally, the solution output by the algorithm satisfies $w(\mathcal{P}) \geqslant 1/2(w(\mathcal{P}_L) + w(\mathcal{P}_R))$ and we directly deduce from inequalities [18.2] the expected result. The instance $I = (G, w)$ that provides the tightness is depicted in Figure 18.7. It consists of a graph on $12n$ vertices on which we can easily observe that $w(\mathcal{P}_L) = w(\mathcal{P}_R) = 2n(n + 2)$ and $w(\mathcal{P}^*) = 2n(2n + 2)$. ∎

Concerning the unweighted case, we may obtain the same performance ratio without the restriction on the graph maximum degree. The main differences compared to the previous algorithm lie in the construction of the two graphs G_L, G_R: starting from G, we duplicate each vertex $r_i \in R$ by adding a new vertex r_i' with the same neighborhood as r_i (this operation, often called *multiplication of vertices* in other works, is used in the characterization of perfect graphs). We then add the edge $[r_i, r_i']$. If R_L denotes the vertex set $\{r_i, r_i' | r_i \in R\}$, the following properties hold:

PROPERTY 18.3.–

(i) From any matching M on G_L, we can deduce a matching M' of cardinality $|M'| \geqslant |M|$ on G_L that saturates R_L.

(ii) From any matching M on G_L (resp., on G_R) that saturates R_L (resp., L_R), we can deduce a $\mathbf{P}_3$ packing $\mathcal{P}_M$ on G of size $|\mathcal{P}_M| = |M| - |R|$.

Proof. For (i). Let M be a matching on G_L and consider a given vertex $r_i \in R$. If M contains no edge incident to $\{r_i, r_i'\}$, then add $[r_i, r_i']$ to M; if M contains an edge e incident to r_i (resp., to r_i'), but no edge incident to r_i' (resp., to r_i), then set $M = M \backslash \{e\} \cup \{[r_i, r_i']\}$.

For (ii). Let M be a matching on G_L that saturates R_L. We respectively denote by J the set of vertices $r_i \in R$ such that $[r_i, r_i'] \in M$ and by $p = |J|$ its cardinality. We consider the matching M' deduced from M by deleting the edges $[r_i, r_i']$; hence,

$|M'| = |M| - p$. From the fact that M saturates R_L, we first deduce that $|M| = |R_L| - p = 2|R| - p$; we then observe that, for any vertex $r_i \notin J$, there are two edges $[l_i^1, r_i]$ and $[l_i^2, r'i]$ in M', that define the $\mathbf{P_3}$ $P_i = \{l_i^1, r_i, l_i^2\}$ of the initial graph G. The collection $\mathcal{P}_M = \cup_{r_i \notin J}\{P_i\}$ obviously is a $\mathbf{P_3}$ packing of size $|M'|/2$ on G. We just have to obverse that $|M'| = 2|R| - 2p = 2(|M| - |R|)$ in order to conclude. ∎

P$_3$-Packing

1 Build the graph G_L (resp., G_R) obtained from G by *multiplication of vertices* on R (resp., on L);

2 Compute a maximum size matching M_L (resp., M_R) on G_L (resp., on G_R); According to Property 18.3 item (i), deduce from M_L (resp., from M_R) a maximum size matching M_L^* (resp., M_R^*) that saturates R_L (resp., L_R);

3 According to Property 18.3 item (ii), deduce from M_L^* (resp., from M_R^*) a $\mathbf{P}_3$ packing $\mathcal{P}_L$ (resp., $\mathcal{P}_R$) of size $|M_L^*| - |R|$ (resp., $|M_R^*| - |L|$);

4 Output the best packing $\mathcal{P}$ among $\mathcal{P}_L$ and $\mathcal{P}_R$.

The approximate algorithm P$_3$-Packing works as previously, except that we calculate a maximum (size) matching M_L^* (resp., M_R^*) on G_L (resp., G_R) that saturates R_L (resp., L_R) in step 2, and that the $\mathbf{P}_3$ packing $\mathcal{P}_L$ (resp., $\mathcal{P}_R$) is obtained from M_L^* (resp., M_R^*) by deleting the edges $[r_i, r'_i]$ (resp., $[l_i, l'_i]$) in step 3.

THEOREM 18.6.– P$_3$-*Packing provides a* $1/2$*-approximation for* MAX$\mathbf{P}_3$PACKING *in bipartite graphs and this ratio is tight. The time complexity of this algorithm is* $\mathcal{O}(m\sqrt{n})$.

Proof. Let $\mathcal{P}_L^* = \{P_1, \cdots, P_q\}$ be the set of paths from the optimal solution having their two endpoints in L; $\mathcal{P}_L^*$ can easily be converted on G_L into a matching M of size $|M| = 2q + (|R| - q) = |\mathcal{P}_L^*| + |R|$. From the optimality of M_L^* on G_L, we deduce that $|M_L^*| \geqslant |M|$ and hence, that $|\mathcal{P}_L| \geqslant |\mathcal{P}_L^*|$. The same obviously holds for $\mathcal{P}_R^*$ and the result is immediate. The time complexity of the unweighted version of the algorithm is still dominated by that of calculating a maximum (size) matching, i.e., $\mathcal{O}(m\sqrt{n})$, [LOV 86]. The proof of tightness is omitted. ∎

18.3.3. MIN3-PATHPARTITION *in general graphs*

To our knowledge, the approximability of MINk-PATHPARTITION (or MINPATH-PARTITION) has not been studied so far. Here, we propose a $3/2$-approximation for MIN3-PATHPARTITION. Although this problem can be viewed as an instance of 3-set cover (view the set of all paths of length 0, 1 or 2 in G as sets on V), MIN3-PATHPARTITION and the minimum 3-set cover problem are different. For instance,

consider a star $K_{1,2n}$; the optimum value of the corresponding 3-set cover instance is n, whereas the optimum value of the 3-path partition is $2n-1$.

Note that, concerning MINPATHPARTITION (i.e., the approximation of $\rho(G)$), we can trivially see that it is not $(2-\varepsilon)$-approximable, from the fact that deciding whether $\rho(G)=1$ or $\rho(G)\geqslant 2$ is **NP**-complete. Actually, we can more generally establish that $\rho(G)$ is not in **APX**: otherwise, we could obtain a **PTAS** for the traveling salesman problem with weight 1 and 2 when $\mathrm{opt}(I)=n$, which is not possible, unless **P**=**NP**.

The algorithm `Minimum 3Path Partition` we propose runs in two phases: first, it calculates a maximum matching M_1^* on the input graph $G=(V,E)$; then, it matches through M_2^* a maximum number of edges from M_1^* to vertices from $V\backslash M_1^*$. Those two matchings define the $\mathbf{P}_3$ and the $\mathbf{P}_2$ of the approximate solution.

THEOREM 18.7.– *`Minimum 3Path Partition` provides a $3/2$-approximation for* MIN3-PATHPARTITION *in general graphs within $\mathcal{O}(nm+n^2\log n)$ time and this ratio is tight.*

Proof. Let $G=(V,E)$ be an instance of MIN3-PATHPARTITION. Let $\mathcal{P}^*=(\mathcal{P}_2^*,\mathcal{P}_1^*,\mathcal{P}_0^*)$ and $\mathcal{P}'=(\mathcal{P}_2',\mathcal{P}_1',\mathcal{P}_0')$ respectively be an optimal solution and the approximate 3-path partition on G, where $\mathcal{P}_i^*$ and $\mathcal{P}_i'$ denote for $i=0,1,2$ the set of paths of length i. By construction of the approximate solution, we have:

$$\mathrm{apx}(I)=|V|-|M_1^*|-|M_2^*| \qquad [18.3]$$

Let $V_0=(V\backslash V(M_1^*))\backslash\mathcal{P}_0^*$, we consider a subgraph $G_2'=(L,R';E_2')$ of G_2, where R' and E_2' are defined as: $R'=\{r_v\in R|v\in V_0\}$ and E_2' contains the edge $[l_e,r_v]\in E_2'$ if and only if there is an edge of $\mathcal{P}^*$ that links v to an endpoint of e. By definition of V_0, we deduce that $d_{G_2'}(r_v)\geqslant 1$ for any $v\in V_0$ (V_0 is an independent set of G). Moreover, we have $d_{G_2'}(l_e)\leqslant 2$ for any $e\in M_1^*$ (M_1^* is an optimal matching). Thus, we get:

$$|M_2^*|\geqslant 1/2|R'|=1/2\left(|V|-2|M_1^*|-|\mathcal{P}_0^*|\right) \qquad [18.4]$$

From relations [18.3] and [18.4], we deduce:

$$\mathrm{apx}(I)=|V|-|M_1^*|-|M_2^*|\leqslant 1/2\left(|V|+|\mathcal{P}_0^*|\right) \qquad [18.5]$$

Now, consider the optimal solution. From $|V|=3|\mathcal{P}_2^*|+2|\mathcal{P}_1^*|+|\mathcal{P}_0^*|$, we trivially have:

$$\mathrm{opt}(I)=|\mathcal{P}_2^*|+|\mathcal{P}_1^*|+|\mathcal{P}_0^*|\geqslant 1/3\left(|V|+|\mathcal{P}_0^*|\right) \qquad [18.6]$$

Thus, we obtain the expected result. The proof of tightness is omitted. Concerning the time complexity, we refer again to [LOV 86]. ∎

`Minimum 3Path Partition`

1 Calculate a maximum matching M_1^* on G;

2 Build a bipartite graph $G_2 = (L, R; E_2)$ where $L = \{l_e | e \in M_1^*\}$, $R = \{r_v | v \in V \setminus V(M_1^*)\}$, and $[l_e, r_v] \in E_2$ if and only if the corresponding isolated vertex $v \notin V(M_1^*)$ is adjacent in G to the edge $e \in M_1^*$;

3 Calculate a maximum matching M_2^* on G_2;

4 Output $\mathcal{P}'$ the 3-paths partition deduced from M_1^*, M_2^*, and $V \setminus V(M_1^* \cup M_2^*)$. Precisely, if $M_1' \subseteq M_1^*$ is the set of edges adjacent to M_2^*, then the paths of length 2 are given by $M_1' \cup M_2^*$, the paths of length 1 are given by $M_1^* \setminus M_1'$, and the paths of length 0 (i.e., the isolated vertices) are given by $V \setminus V(M_1^* \cup M_2^*)$.

18.4. Standard and differential approximation of $\mathbf{P}_k\mathbf{P}$

From now, we will exclusively deal with the approximability of MAX$\mathbf{P}_k$P and MIN$\mathbf{P}_k$P, from both standard and differential points of view. We recall that $\mathbf{P}_k$P is the special case of MAXW$\mathbf{P}_k$PACKING where the graph is complete on kn vertices. We first discuss the differential approximability of $\mathbf{P}_k$P, for any constant value k, by connection to the differential approximability of the traveling salesman problem. The second part of this section then focuses on the special case where $k = 4$, in the aim of extensively analyzing the approximate algorithm proposed by Hassin and Rubinstein, which is described in section 18.4.2.1. We first considerer, on the one hand, general and metric instances for the standard ratio (section 18.4.2.2) and, on the other hand, general instances for the differential ratio (section 18.4.2.3). We then switch to bi-valuated instances, i.e., $\{1, 2\}$-instances for the standard ratio (section 18.4.2.4) and $\{a, b\}$-instances for the differential ratio (section 18.4.2.5).

18.4.1. *Differential approximation of $\mathbf{P}_k\mathbf{P}$ from the traveling salesman problem*

A common technique in order to obtain an approximate solution for MAX$\mathbf{P}_k$P from a Hamiltonian cycle is called the *deleting and turning around* method (see for instance [HAS 97, HAS 06, FRE 78]). Starting from a tour, this method builds k solutions of MAX$\mathbf{P}_k$P and picks the best among them, where the ith solution is obtained by deleting every kth edge from the input cycle, starting from its ith edge. The quality of the output $\mathcal{P}'$ obviously depends on the quality of the initial tour; in this way, it is proven in [HAS 97, HAS 06], that any ε-standard approximation for MAXTSP provides a $\frac{k-1}{k}\varepsilon$-standard approximation for MAX$\mathbf{P}_k$P. From a differential point of view,

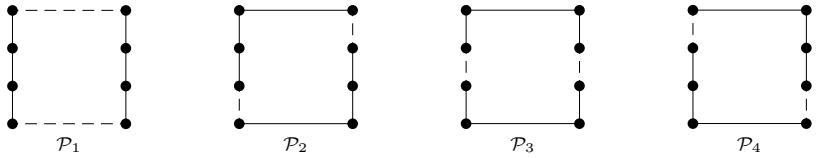


Figure 18.8. *An example of the 4 solutions $\mathcal{P}_1, \ldots, \mathcal{P}_4$*



Figure 18.9. *A worst solution and an optimal solution when $n = 1$*

things are less optimistic: even for $k = 4$, there exists an instance family $(I_n)_{n \geqslant 1}$ that verifies $\mathrm{apx}(I_n) = \frac{1}{2}\mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I_n) + \frac{1}{2}\mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I_n)$. This instance family is defined as $I_n = (K_{8n}, w)$ for $n \geqslant 1$, where the vertex set $V(K_{8n})$ may be partitioned into two sets $L = \{\ell_1, \ldots, \ell_{4n}\}$ and $R = \{r_1, \ldots, r_{4n}\}$ so that the associated weight function w is 0 on $L \times L$, 2 on $R \times R$ and 1 on $L \times R$. Thus, for any $n \geqslant 1$, the following property holds:

PROPERTY 18.4.– $\mathrm{apx}(I_n) = 6n$, $\mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I_n) = 8n$, $\mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I_n) = 4n$.

Proof. If the initial tour is described as $\Gamma = \{e_1, \ldots, e_n, e_1\}$, then the *deleting and turning around* method produces 4 solutions $\mathcal{P}_1, \ldots, \mathcal{P}_4$ where $\mathcal{P}_i = \cup_{j=0}^{n-1}\{\{e_{j+i}, e_{j+i+1}, e_{j+i+2}\}\}$ for $i = 1, \ldots, 4$ (indices are considered mod n). Figure 18.8 provides an illustration of this process (the dashed lines correspond to the edges from $\Gamma \setminus \mathcal{P}_i$).

Observe that any optimal tour Γ on I_n has total weight $8n$ (consider that any tour contains as many edges with their two endpoints in L as edges with their two endpoints in R). Hence, starting from the optimal cycle $\Gamma^* = [r_1, \ldots, r_{4n}, l_1, \ldots, l_{4n}, r_1]$, any of the four solutions $\mathcal{P}_1, \ldots, \mathcal{P}_4$ output by the algorithm (see Figure 18.8) has value $w(\mathcal{P}_i) = 6n$, while an optimal solution $\mathcal{P}^*$ and a worst solution $\mathcal{P}_*$ are of total weight respectively $8n$ and $4n$ (see Figure 18.9). Indeed, because any $\mathbf{P}_4$-partition $\mathcal{P}$ is a $2n$ edge cut down tour, we get, on the one hand, $\mathrm{opt}_{\mathrm{MAXTSP}}(I_n) \geqslant w(\mathcal{P})$ and, on the other hand, $w(\mathcal{P}) \geqslant 8n - 4n = 4n$, which concludes this argument. ∎

Nevertheless, the deleting and turning around method leads to the following weaker differential approximation relation:

LEMMA 18.1.– *From an ε-differential approximation of* MAXTSP*, we can polynomially calculate an $\frac{\varepsilon}{k}$-differential approximation of* MAX$\mathbf{P}_k$P*. In particular, we deduce from [HAS 01, MON 02b] that* MAX$\mathbf{P}_k$P *is $\frac{2}{3k}$-differential approximable.*

Proof. Let us show that the following inequality holds for any instance $I = (K_{kn}, w)$ of MAX$\mathbf{P}_k$P:

$$\mathrm{opt}_{\text{MAXTSP}}(I) \geqslant \frac{1}{k-1}\mathrm{opt}_{\text{MAX}\mathbf{P}_k\text{P}}(I) + \mathrm{wor}_{\text{MAX}\mathbf{P}_k\text{P}}(I) \qquad [18.7]$$

Let $\mathcal{P}^*$ be an optimal solution of MAX$\mathbf{P}_k$P, then arbitrarily add some edges to $\mathcal{P}^*$ in order to obtain a tour Γ. From this latter, we can deduce $k-1$ solutions $\mathcal{P}_i$ for $i = 1, \ldots, k-1$, by applying the deleting and turning around method in such a way that any of the solutions $\mathcal{P}_i$ contain $(\Gamma \setminus \mathcal{P}^*)$. Thus, we get $(k-1)\mathrm{wor}_{\text{MAX}\mathbf{P}_k\text{P}}(I) \leqslant \sum_{i=1}^{k-1} w(\mathcal{P}_i) = (k-1)w(\Gamma) - \mathrm{opt}_{\text{MAX}\mathbf{P}_k\text{P}}(I)$. Hence, consider that we also have $w(\Gamma) \leqslant \mathrm{opt}_{\text{MAXTSP}}(I)$ and the result follows. By applying again the deleting and turning around method, but this time from a worst tour, we may obtain k approximate solutions of MAX$\mathbf{P}_k$P, which allows us to deduce:

$$\mathrm{wor}_{\text{MAXTSP}}(I) \geqslant \frac{k}{k-1}\mathrm{wor}_{\text{MAX}\mathbf{P}_k\text{P}}(I) \qquad [18.8]$$

Finally, let Γ' be an ε-differential approximation of MAXTSP and we deduce from Γ' k approximate solutions of MAX$\mathbf{P}_k$P. If $\mathcal{P}'$ is set to the best one, we get $w(\mathcal{P}') \geqslant \frac{k}{k-1}w(\Gamma')$ and thus:

$$\mathrm{apx}(I) \geqslant \frac{k}{k-1}w(\Gamma') \geqslant \frac{k}{k-1}\big(\varepsilon\mathrm{opt}_{\text{MAXTSP}}(I) + (1-\varepsilon)\mathrm{wor}_{\text{MAXTSP}}(I)\big) \qquad [18.9]$$

Using inequalities [18.7], [18.8] and [18.9], we get $\mathrm{apx}(I) \geqslant \frac{\varepsilon}{k}\mathrm{opt}_{\text{MAX}\mathbf{P}_k\text{P}}(I) + (1-\frac{\varepsilon}{k})\mathrm{wor}_{\text{MAX}\mathbf{P}_k\text{P}}(I)$ and the proof is complete. ∎

To conclude with the relationship between $\mathbf{P}_k$P and TSP with respect to their approximability, observe that the minimization case with respect to standard approximation is also trickier. Notably, if we consider MINMETRIC$\mathbf{P}_4$P, then the instance family $I'_n = (K_{8n}, w')$ built as the same as I_n with a distinct weight function defined as $w'(\ell_i, \ell_j) = w'(r_i, r_j) = 1$ and $w'(\ell_i, r_j) = n^2 + 1$ for any i, j, then we have: $\mathrm{opt}_{\text{TSP}}(I'_n) = 2n^2 + 8n$ whereas $\mathrm{opt}_{\mathbf{P}_4\text{P}}(I'_n) = 6n$.

18.4.2. *Approximating* $\mathbf{P}_4\mathbf{P}$ *by means of optimal matchings*

Here starts the analysis, from both a standard and a differential point of view, of an algorithm proposed by Hassin and Rubinstein in [HAS 97], where the authors show that the approximate solution is a 3/4-standard approximation for MAX$\mathbf{P}_4$P. We prove that, with respect to the standard ratio, this algorithm provides new approximation ratios for METRIC$\mathbf{P}_4$P, namely: the approximate solution respectively achieves a 3/2, a 7/6 and a 9/10-standard approximation for MINMETRIC$\mathbf{P}_4$P, MIN$\mathbf{P}_4\mathrm{P}_{1,2}$ and MAX$\mathbf{P}_4\mathrm{P}_{1,2}$. As a corollary of a more general result, we also obtain an alternative proof of the result of [HAS 97]. We then prove that, under differential ratio, the approximate solution is a 1/2-approximation for general $\mathbf{P}_4$P and a 2/3-approximation for $\mathbf{P}_4\mathrm{P}_{a,b}$. In addition to the new approximation bounds that they provide, the obtained results establish the robustness of the algorithm that is addressed here, since this latter provides good quality solutions irrespective of the version of the problem we deal with and irrespective of the approximation framework within which we estimate the approximate solutions.

Note that the gap between differential and standard approximation levels that might be reached for a maximization problem comes from the fact that, within the differential framework, the approximate value is located within the tighter interval $[\mathrm{wor}(I), \mathrm{opt}(I)]$, instead of $[0, \mathrm{opt}(I)]$ for the standard measure. That is the aim of differential approximation: the reference to $\mathrm{wor}(I)$ makes this measure both more precise (relevant with respect to the notion of guaranteed performance) and more robust (in the sense that minimizing and maximizing tend to be equivalent and, more generally, differential ratio is invariant under affine transformation of the objective function).

18.4.2.1. *Description of the algorithm*

The algorithm proposed in [HAS 97] runs in two stages: first, it calculates an optimum weight perfect matching M on $I = (K_{4n}, w)$; then, it builds on the edges of M a second optimum weight perfect matching R in order to complete the solution (note that "optimum weight" signifies "*maximum weight*" if the goal is to maximize and "*minimum weight*" if the goal is to minimize). Precisely, we define the instance $I' = (K_{2n}, w')$ (having a vertex v_e in K_{2n} for each edge $e \in M$), where the weight function w' is defined as follows: for any edge $[v_{e_1}, v_{e_1}]$ on I', $w'(v_{e_1}, v_{e_2})$ is set to the weight of the heaviest edge that links e_1 and e_2 in I, i.e., if $e_1 = [x_1, y_1]$ and $e_2 = [x_2, y_2]$, then $w'(v_{e_1}, v_{e_2}) = \max\{w(x_1, x_2), w(x_1, y_2), w(y_1, x_2), w(y_1, y_2)\}$ (when dealing with the minimization version of the problem, set the weight to the lightest). We thus build on (K_{2n}, w') an optimum weight matching R, which is then transposed to the initial graph (K_{4n}, w) by selecting on K_{4n} the edges that realizes the same weight. Since the calculation of an optimum weight perfect matching is polynomial, the whole algorithm runs in polynomial time, whether the goal is to minimize or to maximize.

18.4.2.2. *General* $\mathbf{P}_4$P *within the standard framework*

For any solution $\mathcal{P}$, we denote respectively by $M_{\mathcal{P}}$ and $R_{\mathcal{P}}$ the set of the end edges and the set of the middle edges of its paths. Furthermore, we consider for any path $P = \{x, y, z, t\}$ of the solution the edge $[t, x]$ that completes P into a cycle. If $\overline{R}_{\mathcal{P}}$ denotes the set of these edges, we observe that $R_{\mathcal{P}} \cup \overline{R}_{\mathcal{P}}$ forms a perfect matching. Finally, for any edge $e \in \mathcal{P}$, we will denote by $P_{\mathcal{P}}(e)$ the $\mathbf{P}_4$ from the solution that contains e and by $C_{\mathcal{P}}(e)$ the 4-edge cycle that contains $P_{\mathcal{P}}(e)$.

LEMMA 18.2.– *For any instance $I = (K_{4n}, w)$ with optimal solution $\mathcal{P}^*$ and for any perfect matching M, there are four pairwise disjoint edge sets A, B, C and D that verify:*

(i) $A \cup B = \mathcal{P}^*$ and $C \cup D = \overline{R}_{\mathcal{P}^*}$.

(ii) $A \cup C$ and $B \cup D$ both are perfect matchings on I.

(iii) $A \cup C \cup M$ is a perfect 2-matching on I whose cycles are of length a multiple of 4.

Proof. Let $\mathcal{P}^* = M_{\mathcal{P}^*} \cup R_{\mathcal{P}^*}$ be an optimal solution, we apply the `Combining perfect matchings` process. At the initialization stage, the connected components of the partial graph induced by $(A \cup C \cup M)$ are either cycles that alternate edges from $(A \cup C)$ and M, or isolated edges from $M_{\mathcal{P}^*} \cap M$. During step 2, at each iteration, the process merges together two connected components of G' into a single cycle that still alternates edges from $(A \cup C)$ and M (an illustration of this merging process is provided in Figure 18.10). Note that all along the process, the sets A, B, C and D define a partition of $\mathcal{P}^* \cup \overline{R}_{\mathcal{P}^*}$ and thus remain pairwise disjoint.

`Combining perfect matchings`

1 $A \longleftarrow M_{\mathcal{P}^*}$, $B \longleftarrow R_{\mathcal{P}^*}$, $C \longleftarrow \{\}$, $D \longleftarrow \overline{R}_{\mathcal{P}^*}$;
Set $G' = (V, A \cup M)$ (consider the simple graph);

2 While $\exists\, e \in R_{\mathcal{P}^*}$ that links two connected components of G', do:

2.1
$A \longleftarrow A \setminus (C_{\mathcal{P}^*}(e) \cap M_{\mathcal{P}^*})$, $B \longleftarrow B \cup (C_{\mathcal{P}^*}(e) \cap M_{\mathcal{P}^*})$;
$B \longleftarrow B \setminus (C_{\mathcal{P}^*}(e) \cap R_{\mathcal{P}^*})$, $A \longleftarrow A \cup (C_{\mathcal{P}^*}(e) \cap R_{\mathcal{P}^*})$;
$D \longleftarrow D \setminus \left(C_{\mathcal{P}^*}(e) \cap \overline{R}_{\mathcal{P}^*}\right)$, $C \longleftarrow C \cup \left(C_{\mathcal{P}^*}(e) \cap \overline{R}_{\mathcal{P}^*}\right)$;

2.2 $G' \longleftarrow (V, A \cup C \cup M)$;

3 Output A, B, C and D.

• For (i): immediate from definition of the process (edges from $\mathcal{P}^*$ are moved from A to B, from B to A, but never out of $A \cup B$; the same holds for $\overline{R}_{\mathcal{P}^*}$ and the two sets C and D).

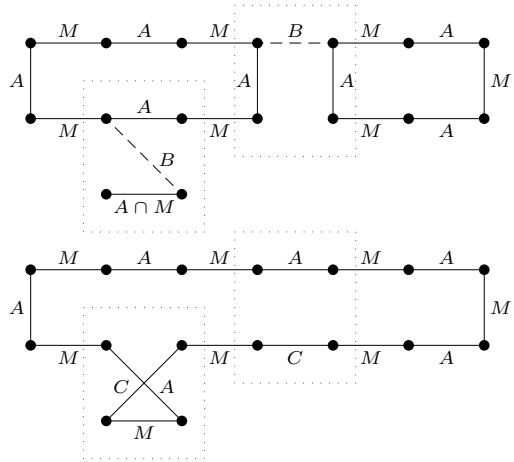


Figure 18.10. *The construction of sets A and C*

• For (ii): at the initialization stage, $A \cup C$ and $B \cup D$ respectively coincide with $M_{\mathcal{P}^*}$ and $R_{\mathcal{P}^*} \cup \overline{R}_{\mathcal{P}^*}$, which are each a perfect matching. More precisely, for any path $P \in \mathcal{P}^*$, if $C(P)$ denotes the associated 4-edge cycle, then $A \cup C$ and $B \cup D$ respectively contain the perfect matching $C(P) \cap M_{\mathcal{P}^*}$ and $C(P) \cap (R_{\mathcal{P}^*} \cup \overline{R}_{\mathcal{P}^*})$ on $V(P)$. Now, at each iteration, the algorithm swaps the perfect matchings that are used in $A \cup C$ or in $B \cup D$ in order to cover the vertices of a given path P and thus, both $A \cup C$ and $B \cup D$ remain perfect matchings.

• For (iii): at the end of the process, the stopping criterion ensures that $(A \cup C) \cap M = \{\}$ and thus, as the union of two perfect matchings, $A \cup C \cup M$ is a perfect 2-matching. Now, consider a cycle Γ of $G' = (V, A \cup C \cup M)$; by definition of step 2, any edge e from $R_{\mathcal{P}^*}$ that is incident to Γ has its two endpoints in $V(\Gamma)$, which means that Γ contains either the two edges of $C_{\mathcal{P}^*}(e) \cap M_{\mathcal{P}^*}$, or the two edges of $C_{\mathcal{P}^*}(e) \cap (R_{\mathcal{P}^*} \cup \overline{R}_{\mathcal{P}^*})$. In other words, if any vertex u from any path $P \in \mathcal{P}^*$ belongs to $V(\Gamma)$, then the whole vertex set $V(P)$ is actually a subset of $V(\Gamma)$ and therefore, we deduce that $|V(\Gamma)| = 4q$, where q is the number of paths $P \in \mathcal{P}^*$ such that Γ contains $V(P)$. ∎

THEOREM 18.8.– *The solution $\mathcal{P}'$ provided by the algorithm achieves a 3/2-standard approximation for* MINMETRIC$\mathbf{P}_4$P *and this ratio is tight.*

Proof. Let $\mathcal{P}^*$ be an optimal solution on $I = (K_{4n}, w)$. Using Lemma 18.2 with the perfect matching $M_{\mathcal{P}'}$ of the solution $\mathcal{P}'$, we obtain four pairwise disjoint sets A, B, C and D. According to property (iii), we can split $A \cup C$ into two sets A_1 and A_2 so that $A_i \cup M_{\mathcal{P}'}$ $(i = 1, 2)$ is a $\mathbf{P}_4$-partition (see Figure 18.11 for an illustration).

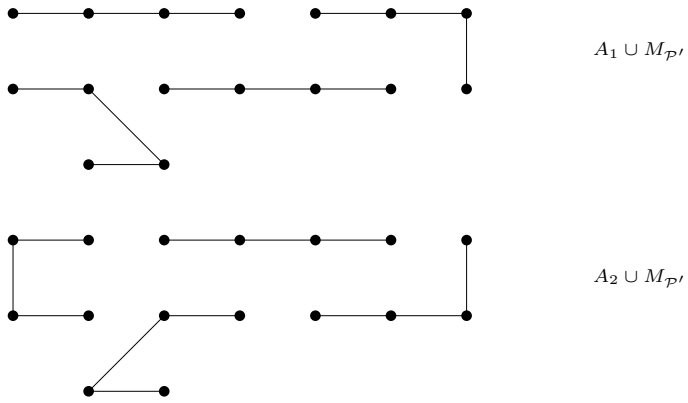


Figure 18.11. *Two possible* $\mathbf{P}_4$ *partitions deduced from* $A \cup C \cup M_{\mathcal{P}'}$

Hence, A_i constitutes an alternative solution for $R_{\mathcal{P}'}$ and because this latter is optimal on $I' = (K_{2n}, w')$, we obtain:

$$2w(R_{\mathcal{P}'}) \leqslant w(A) + w(C) \qquad [18.10]$$

Moreover, item (ii) of Lemma 18.2 states that $B \cup D$ is a perfect matching; since $M_{\mathcal{P}'}$ is a minimum weight perfect matching, we deduce:

$$w(M_{\mathcal{P}'}) \leqslant w(B) + w(D) \qquad [18.11]$$

Hence, by combining inequalities [18.10] and [18.11] (and also considering item (i) of Lemma 18.2), we get:

$$w(M_{\mathcal{P}'}) + 2w(R_{\mathcal{P}'}) \leqslant w(\mathcal{P}^*) + w(\overline{R}_{\mathcal{P}^*}) \qquad [18.12]$$

Inequality [18.12], combined with the observation that $w(\overline{R}_{\mathcal{P}^*}) \leqslant w(\mathcal{P}^*)$ (which is true from the assumption that I satisfies the triangle inequality), leads to the following new inequality:

$$w(M_{\mathcal{P}'}) + 2w(R_{\mathcal{P}'}) \leqslant 2\mathrm{opt}_{\text{MinMetric}\mathbf{P}_4\text{P}}(I) \qquad [18.13]$$

Relation [18.13] together with $w(M_{\mathcal{P}'}) \leqslant w(M_{\mathcal{P}^*}) \leqslant w(\mathcal{P}^*)$ complete the proof. Finally, the tightness is provided by the instance family $I_n = (K_{8n}, w)$ that has been described in Property 18.4. ∎

Concerning the maximization case and using Lemma 18.2, we can also obtain an alternative proof of the result given in [HAS 97].

THEOREM 18.9.– *The solution $\mathcal{P}'$ provided by the algorithm achieves a 3/4-standard approximation for* MAX$\mathbf{P}_4$P.

Proof. The inequality [18.12] becomes:

$$w(M_{\mathcal{P}'}) + 2w(R_{\mathcal{P}'}) \geqslant \mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) + w(\overline{R}_{\mathcal{P}^*}) \qquad [18.14]$$

Since $M_{\mathcal{P}'}$ is a maximum weight perfect matching, the approximate value obviously satisfies $2 \times w(M_{\mathcal{P}'}) \geqslant \mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) + w(\overline{R}_{\mathcal{P}^*})$; hence, we deduce $\mathrm{apx}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) \geqslant \frac{3}{4}\left(\mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) + w(\overline{R}_{\mathcal{P}^*})\right)$. ∎

18.4.2.3. *General* $\mathbf{P}_4$P *within the differential framework*

When dealing with the differential ratio, MIN$\mathbf{P}_4$P, MINMETRIC$\mathbf{P}_4$P and MAX$\mathbf{P}_4$P are equivalent to approximate, since $\mathbf{P}_k$P problems belong to the class *FGNPO*, [MON 02a]. Note that such an equivalence is more generally true for any couple of problems that only differ by an affine transformation of their objective function.

THEOREM 18.10.– *The solution $\mathcal{P}'$ provided by the algorithm achieves a 1/2-differential approximation for* $\mathbf{P}_4$P *and this ratio is tight.*

Proof. We consider the maximization version. First, observe that $\overline{R}_{\mathcal{P}^*}$ is an n-cardinality matching. Let M be any perfect matching of I such that $M \cup \overline{R}_{\mathcal{P}^*}$ forms a $\mathbf{P}_4$-partition we have:

$$w(M) + w(\overline{R}_{\mathcal{P}^*}) \geqslant \mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) \qquad [18.15]$$

Adding inequalities [18.14] and [18.15], and since $w(M_{\mathcal{P}'}) \geqslant w(M)$, we conclude that:

$$2\mathrm{apx}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) = 2\left(w(M_{\mathcal{P}'}) + w(R_{\mathcal{P}'})\right) \geqslant \mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) + \mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I)$$

$$\Rightarrow \frac{\mathrm{apx}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) - \mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I)}{\mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) - \mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I)} \geqslant 1/2$$

In order to establish the tightness of this ratio, we refer again to Property 18.4. ∎

18.4.2.4. *Bi-valued metric* $\mathbf{P}_4$P *with weights 1 and 2 within the standard framework*

As it has been recently done for MINTSP in [BER 06, BLA 05] and because such an analysis enables a keener comprehension of a given algorithm, we now focus on instances where any edge weight is either 1 or 2. Note that, since the $\mathbf{P}_4$-partition problem is **NP**-complete, the problems MAX$\mathbf{P}_4$P$_{1,2}$ and MIN$\mathbf{P}_4$P$_{1,2}$ are still **NP**-hard.

Let us first introduce some more notation. For a given instance $I = (K_{4n}, w)$ of $\mathbf{P}_4$P$_{1,2}$ with $w(e) \in \{1, 2\}$, we denote by $M_{\mathcal{P}',i}$ (resp., by $R_{\mathcal{P}',i}$) the set of edges from $M_{\mathcal{P}'}$ (resp., from $R_{\mathcal{P}'}$) that are of weight i. If we aim at maximizing, then p (resp., q) indicates the cardinality of $M_{\mathcal{P}',2}$ (resp., of $R_{\mathcal{P}',2}$); otherwise, it indicates the quantity $|M_{\mathcal{P}',1}|$ (resp., $|R_{\mathcal{P}',1}|$). In any case, p and q respectively count the number of "*optimum weight edges*" in the sets $M_{\mathcal{P}'}$ and $R_{\mathcal{P}'}$. With respect to the optimal solution, we define the sets $M_{\mathcal{P}^*,i}$, $R_{\mathcal{P}^*,i}$ for $i = 1, 2$ and the cardinalities p^*, q^* as the same. Without loss of generality, we may assume that the following property always holds for $\mathcal{P}^*$:

PROPERTY 18.5.– For any 3-edge path $P \in \mathcal{P}^*$,
$|P \cap M_{\mathcal{P}^*,2}| \geqslant |P \cap R_{\mathcal{P}^*,2}|$ if the goal is to maximize,
$|P \cap M_{\mathcal{P}^*,1}| \geqslant |P \cap R_{\mathcal{P}^*,1}|$ if the goal is to minimize.

Proof. Assume that the goal is to maximize. If $|P \cap M_{\mathcal{P}^*,2}| < |P \cap R_{\mathcal{P}^*,2}|$, then $\mathcal{P}^*$ would contain a path $P = \{[x, y], [y, z], [z, t]\}$ with $w(x, y) = w(z, t) = 1$ and $w(y, z) = 2$; thus, by swapping P for $P' = \{[y, z], [z, t], [t, x]\}$ within $\mathcal{P}^*$, we could generate an alternative optimal solution. ∎

LEMMA 18.3.– *For any instance $I = (K_{4n}, w)$, if $\mathcal{P}'$ is a feasible solution and $\mathcal{P}^*$ is an optimal solution, then there exists an edge set A that verifies:*

(i) $A \subseteq M_{\mathcal{P}^*,2} \cup R_{\mathcal{P}^*,2}$ (resp., $A \subseteq M_{\mathcal{P}^*,1} \cup R_{\mathcal{P}^*,1}$) and $|A| = q^*$ if the goal is to maximize (resp., to minimize);

(ii) $G' = (V, M_{\mathcal{P}'} \cup A)$ is a simple graph made of pairwise disjoint paths.

Proof. We only prove the maximization case. We now consider G' as the multi-graph induced by $M_{\mathcal{P}'} \cup R_{\mathcal{P}^*,2}$ (the edges from $M_{\mathcal{P}'} \cap R_{\mathcal{P}^*,2}$ appear twice). This graph consists of elementary cycles and paths: its cycles alternate edges from $M_{\mathcal{P}'}$ and $R_{\mathcal{P}^*,2}$ (in particular, the 2-edge cycles correspond to the edges from $R_{\mathcal{P}^*,2} \cap M_{\mathcal{P}'}$); its paths (that may be of length 1) also alternate edges from $M_{\mathcal{P}'}$ and $R_{\mathcal{P}^*,2}$, with the particularity that their end edges all belong to $M_{\mathcal{P}'}$.

Let Γ be a cycle on G' and e be an edge from $\Gamma \cap R_{\mathcal{P}^*,2}$. If $P_{\mathcal{P}^*}(e) = \{x, y, z, t\}$ denotes the path from the optimal solution that contains e, then $e = [y, z]$. The initial vertex x of the path $P_{\mathcal{P}^*}(e)$ is necessarily the endpoint of some path from G': otherwise, the edge $[x, y]$ from $P_{\mathcal{P}^*}(e) \cap M_{\mathcal{P}^*}$ would be incident to 2 distinct edges from

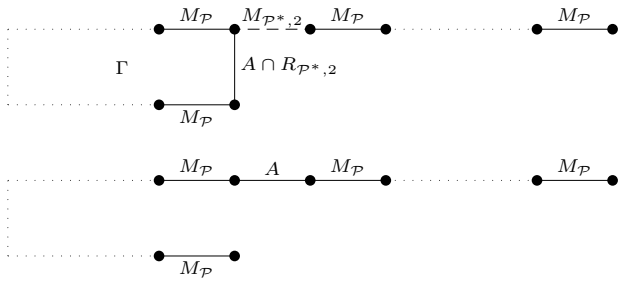


Figure 18.12. *The construction of set A*

$R_{\mathcal{P}^*}$, which would contradict the fact that $\mathcal{P}^*$ is a $\mathbf{P}_4$ partition. The same obviously holds for t. W.l.o.g., we may assume from Property 18.5 that $[x, y] \in M_{\mathcal{P}^*,2}$. In light of these comments and in order to build an edge set A that fulfills the requirements (i) and (ii), we proceed as follows:

`Combining matchings`

1 Set $A = R_{\mathcal{P}^*,2}$; Set $G' = (V, A \cup M_{\mathcal{P}'})$ (consider the multi-graph);
2 While there exists a cycle Γ in G', do:
  2.1 Pick e from $\Gamma \cap R_{\mathcal{P}^*,2}$;
  Pick f from $P_{\mathcal{P}^*}(e) \cap M_{\mathcal{P}^*,2}$;
  $A \longleftarrow A \setminus \{e\} \cup \{f\}$;
  2.2 $G' \longleftarrow (V, A \cup M_{\mathcal{P}'})$;
3 Output A.

By construction, the set A output by the algorithm is of cardinality q^* and contains exclusively edges of weight 2. Furthermore, each iteration of step 2 merges a cycle and a path of $A \cup M_{\mathcal{P}}$ into a path (an illustration of this merging operation is provided by Figure 18.12). Hence, the stopping criterion ensures that, at the end of this step, $G' = (V, A \cup M_{\mathcal{P}})$ is a simple graph whose connected components are elementary paths. Finally, the existence of edge f at step 2.1 directly comes from Property 18.5. ∎

THEOREM 18.11.– *The solution $\mathcal{P}'$ provided by the algorithm achieves a 9/10-standard approximation for* MAX$\mathbf{P}_4$P$_{1,2}$ *and a 7/6-standard approximation for* MIN$\mathbf{P}_4$P$_{1,2}$. *These ratios are tight.*

Proof. Let us consider A as the edge subset of the optimal solution that may be deduced from the application of Lemma 18.3 to the approximate solution. We arbitrarily complete A by means of an edge set B so that $A \cup B \cup M_{\mathcal{P}'}$ constitutes a perfect 2-matching. As we did while proving Theorem 18.8, we split the edge set $A \cup B$ into

two sets A_1 and A_2 in order to obtain two $\mathbf{P}_4$-partitions $M_{\mathcal{P}'} \cup A_1$ and $M_{\mathcal{P}'} \cup A_2$ of $V(K_{4n})$. As both A_1 and A_2 complete $M_{\mathcal{P}'}$ into a $\mathbf{P}_4$-partition and because $R_{\mathcal{P}'}$ is optimal, we deduce that A_i does not contain more "*optimum weight edges*" than $R_{\mathcal{P}'}$, i.e., $q \geqslant |\{e \in A_i : w(e) = 2\}|$ if the goal is to maximize, $q \geqslant |\{e \in A_i : w(e) = 1\}|$ otherwise. Since $A \subseteq A_1 \cup A_2$ and $|A| = q^*$, we immediately deduce:

$$q \geqslant q^*/2 \qquad [18.16]$$

On the other hand, by the optimality of $M_{\mathcal{P}'}$:

$$p \geqslant \max\{p^*, q^*\} \qquad [18.17]$$

Moreover, the quantities p^* and q^* structurally verify:

$$n \geqslant \max\{p^*/2, q^*\} \qquad [18.18]$$

Finally, we can express the value of any solution $\mathcal{P}$ as:

$$w(\mathcal{P}) = 3n + (p+q) \text{ (if goal} = \max\text{)}, 6n - (p+q) \text{ (if goal} = \min\text{)} \qquad [18.19]$$

The claimed results can now be obtained from inequalities [18.16], [18.17], [18.18] and [18.19]:

$$\begin{array}{lcl}
10\mathrm{apx}_{\text{MAX}\mathbf{P}_4\text{P}_{1,2}}(I) & = & 10(3n+p+q) \\
& = & 9(3n) + 3n + 9p + p + 10q \\
& \geqslant & 9(3n) + 3q^* + 9p^* + q^* + 5q^* \\
& = & 9(3n+p^*+q^*) = 9\mathrm{opt}_{\text{MAX}\mathbf{P}_4\text{P}_{1,2}}(I) \\
6\mathrm{apx}_{\text{MIN}\mathbf{P}_4\text{P}_{1,2}}(I) & = & 6(6n-p-q) \\
& = & 6(6n) - 6p - 6q \\
& \leqslant & 6(6n) - 6p^* - 3q^* \\
& \leqslant & 6(6n) - 6p^* - 3q^* + (2n-p^*) + 4(n-q^*) \\
& \leqslant & 7(6n-p^*-q^*) = 7\mathrm{opt}_{\text{MIN}\mathbf{P}_4\text{P}_{1,2}}(I)
\end{array}$$

The tightness for MAX$\mathbf{P}_4$P$_{1,2}$ is established in the instance $I = (K_8, w)$ depicted in Figure 18.13, where the edges of weight 2 are drawn in continuous lines, and the edges of weight 1 on $\mathcal{P}^*$ and $\mathcal{P}'$ are drawn in dotted lines (the other edges are not drawn). We can easily see: $\mathrm{opt}_{\text{MAX}\mathbf{P}_4\text{P}_{1,2}}(I) = 10$ and $\mathrm{apx}_{\text{MAX}\mathbf{P}_4\text{P}_{1,2}}(I) = 9$.

Concerning the minimization case, the ratio is tight on the instance $J = (K_8, w)$ that verifies: $\mathrm{opt}(J) = w(\mathcal{P}^*) = 6$ and $\mathrm{apx}(J) = w(\mathcal{P}') = 7$. $J = (K_8, w)$ is depicted in Figure 18.14 (the 1-weight edges are drawn in continuous lines and the 2-weight edges on $\mathcal{P}^*$ and $\mathcal{P}'$ are drawn in dotted lines). ∎

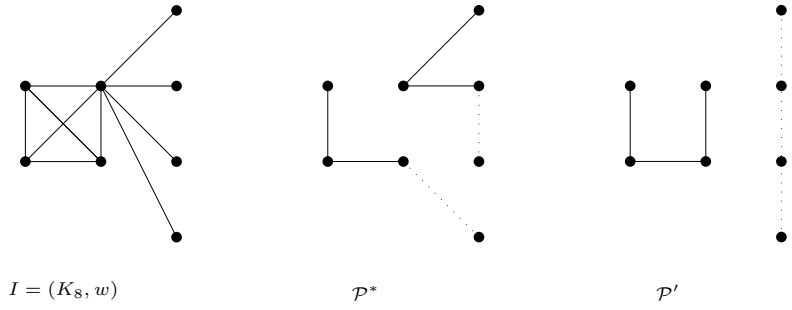


Figure 18.13. *Instance $I = (K_8, w)$ that establishes the tightness for* $\text{MAX}\mathbf{P}_4\text{P}_{1,2}$

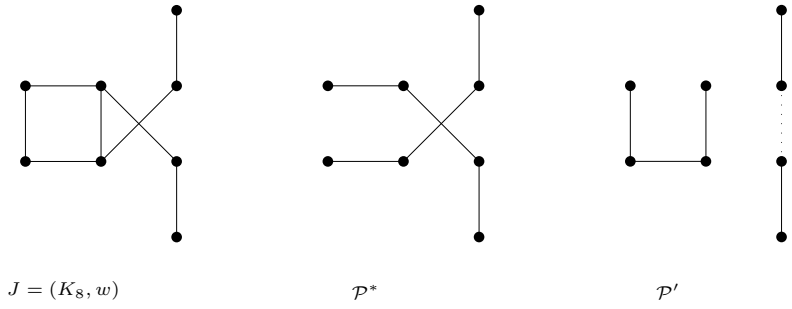


Figure 18.14. *Instance $I = (K_8, w)$ that establishes the tightness for* $\text{MIN}\mathbf{P}_4\text{P}_{1,2}$

18.4.2.5. *Bi-valued metric $\mathbf{P}_4\text{P}$ with weights a and b in the differential framework*

As we have already mentioned, the differential measure is invariant under affine transformation; now, any instance from $\text{MAX}\mathbf{P}_4\text{P}_{a,b}$ or from $\text{MIN}\mathbf{P}_4\text{P}_{a,b}$ can be mapped into an instance of $\text{MAX}\mathbf{P}_4\text{P}_{1,2}$ by way of such a transformation. Thus, proving $\text{MAX}\mathbf{P}_4\text{P}_{1,2}$ is ε-differential approximable actually establishes that $\text{MIN}\mathbf{P}_4\text{P}_{a,b}$ and $\text{MAX}\mathbf{P}_4\text{P}_{a,b}$ are ε-differential approximable for any couple of real values $a < b$. We demonstrate here that Hassin and Rubinstein algorithm achieves a 2/3-differential approximation for $\mathbf{P}_4\text{P}_{1,2}$ and hence, for $\mathbf{P}_4\text{P}_{a,b}$, for any couple of real $a < b$.

Let $I = (K_{4n}, w)$ be an instance of $\text{MAX}\mathbf{P}_4\text{P}_{1,2}$. We recall the notation introduced while proving Theorem 18.11: $p = |M_{\mathcal{P}',2}|$, $p^* = |M_{\mathcal{P}^*,2}|$, $q = |R_{\mathcal{P}',2}|$ and $q^* = |R_{\mathcal{P}^*,2}|$. Furthermore, for $i = 1, 2$, $\mathcal{F}^i$ will refer to the set of paths from $\mathcal{P}'$ whose central edge is of weight i. Note that the paths from $\mathcal{F}^1$ may be of total weight 3, 4 or 5, whereas the paths from $\mathcal{F}^2$ may be of total weight 5 or 6 (at least one extremal edge must be of weight 2, or $M_{\mathcal{P}'}$ is not an optimum weight matching). We will denote by

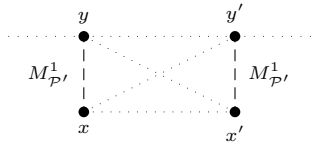


Figure 18.15. *1-weight edges on* $V(M^1_{\mathcal{P}'})$

$\mathcal{F}^2_5$ and $\mathcal{F}^2_6$ the paths from $\mathcal{F}^2$ that are of total weight 5 and 6, respectively. Finally, for $i = 1, 2$, $M^i_{\mathcal{P}'}$ will refer to the set of edges $e \in M_{\mathcal{P}'}$ such that $P_{\mathcal{P}'}(e) \in \mathcal{F}^i$ (i.e., e is element of a path from $\mathcal{P}'$ whose central edge has weight i). According to [18.16] and [18.17], we get:

$$\mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}_{1,2}}(I) \leqslant \min\{3n + p + 2q, 3n + 2p\} \qquad [18.20]$$

To obtain a differential approximation, we also have to produce an efficient bound for $\mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}_{1,2}}(I)$. To do so, we exploit the optimality of $M_{\mathcal{P}'}$ and $R_{\mathcal{P}'}$ in order to exhibit some edges of weight 1 that will enable us to approximate the worst solution. We first consider the vertices from $V(\mathcal{F}^1)$: they are "*easy*" to cover by means of 3-edge paths of total weight 3, since we may immediately deduce from the optimality of $R_{\mathcal{P}'}$ the following property (an illustration is provided by Figure 18.15, where dotted lines indicate edges of weight 1 and dashed lines indicate unspecified weight edges):

PROPERTY 18.6.– $[x, y] \neq [x', y'] \in M^1_{\mathcal{P}'}$ implies that $\forall (u, v) \in \{x, y\} \times \{x', y'\}$, $w(u, v) = 1$.

We now consider the vertices from $V(\mathcal{F}^2_5)$. Let $P = \{x, y, z, t\}$ with $[x, y] \in M_{\mathcal{P}',2}$ be a path from $\mathcal{F}^2_5$ and we deduce from the optimality of $M_{\mathcal{P}'}$ that $w(t, x) = 1$; hence, the 3-edge path $P' = \{y, z, t, x\}$ covers the vertices $\{x, y, z, t\}$ with a total weight 4. Let us assume that $\mathcal{F}^2_6 = \{\}$, then we are able to build a $\mathbf{P}_4$ partition of $V(K_{4n})$ using $3n - |\mathcal{F}^2_5|$ edges of weight 1 and $|\mathcal{F}^2_5|$ edges of weight 2 (one edge of weight 2 is used for each path from $\mathcal{F}^2_5$). Hence, a worst solution costs at most $3n + q$, while the approximate solution is of total weight $3n + p + q$. Thus, using relation [18.20], we would be able to conclude that $\mathcal{P}'$ is a (2/3)-approximation. Of course, there is no reason for $\mathcal{F}^2_6 = \{\}$; nevertheless, this discussion has brought to the fore the following fact: the difficult point of the proof lies in the partitioning of $V(\mathcal{F}^2_6)$ into "*light*" 3-edge paths. In order to deal with these vertices, we first state two more properties that are immediate from the optimality of $M_{\mathcal{P}'}$ and $R_{\mathcal{P}'}$, respectively.

PROPERTY 18.7.– $\left\{ \begin{array}{ll} & [x, y] \in M_{\mathcal{P}',1} \text{ and } [x', y'] \in M_{\mathcal{P}',2} \\ \Rightarrow & \min\{w(x, x'), w(y, y')\} = \min\{w(x, y'), w(y, x')\} = 1 \end{array} \right.$

PROPERTY 18.8.– If $[x, y] \neq [x', y'] \in M^1_{\mathcal{P}'}$ and $P_{\mathcal{P}'} = \{\alpha, \beta, \gamma, \delta\} \in \mathcal{F}^2$, then:
$\left\{ \begin{array}{ll} & \max\{w(u, v) | (u, v) \in \{\alpha, \beta\} \times \{x, y\}\} = 2 \\ \Rightarrow & \max\{w(u, v) | (u, v) \in \{\gamma, \delta\} \times \{x', y'\}\} = 1 \end{array} \right.$

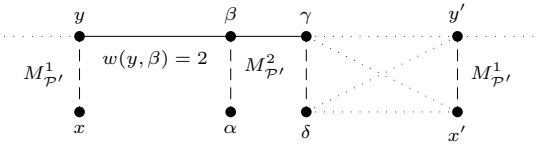


Figure 18.16. *1-weight edges that may be deduced from the optimality of* $R_{\mathcal{P}'}$

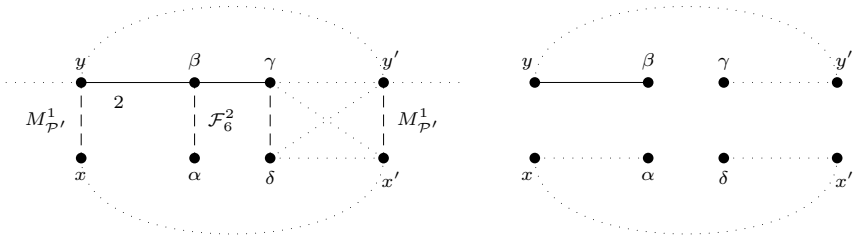


Figure 18.17. *A* $\mathbf{P}_4$ *partition of* $(P, e_1, e_2) \in \mathcal{F}^2_6 \times (M^1_{\mathcal{P}'})^2$ *of total weight at most 7*

An illustration of this latter property is proposed in Figure 18.16, where continuous and dotted lines respectively indicate 2- and 1-weight edges, whereas dashed lines indicate unspecified weight edges. Properties 18.7 and 18.8 give the clue on how to incorporate the vertices of $\mathcal{F}^2_6$ into a packing of "*light*" $\mathbf{P}_4$. The construction of these paths is formalized in the following property and illustrated in Figure 18.17.

PROPERTY 18.9.– Given a path $P \in \mathcal{F}^2_6$ and two edges $[x, y] \neq [x', y'] \in M^1_{\mathcal{P}'}$, there exists a $\mathbf{P}_4$ partition $\mathcal{F} = \{P_1, P_2\}$ of $\big(V(P) \cup \{x, y, x', y'\}\big)$ that is of total weight at most 8. Furthermore, if $[x, y]$ and $[x', y']$ both belong to $M_{\mathcal{P}',1}$, then we can decrease this weight to (at most) 7.

Proof. Consider $P = \{\alpha, \beta, \gamma, \delta\} \in \mathcal{F}^2_6$ and $[x, y] \neq [x', y'] \in M^1_{\mathcal{P}'}$. We set $P_1 = \{\alpha, x, x', \delta\}$ and $P_2 = \{\beta, y, y', \gamma\}$. We know from Property 18.6 that $w(x, x') = w(y, y') = 1$. Thus, if every edge from $\{\alpha, \beta, \gamma, \delta\} \times \{x, x', y, y'\}$ is of weight 1, then $P_1 \cup P_2$ has a total weight 6. Conversely, if there exists a 2-weight edge that links a vertex from $\{\alpha, \beta, \gamma, \delta\}$ to a vertex from $\{x, x', y, y'\}$, we may assume that $[\beta, y]$ is such an edge; we then deduce from Property 18.8 that $w(\delta, x') = w(\gamma, y') = 1$ and hence, that $P_1 \cup P_2$ is of total weight at most 8. Finally, if $w(x, y) = 1$, then $w(\alpha, x) = 1$ from Property 18.7 and thus, $w(P_1) + w(P_2) = 7$. ∎

We are now able to calculate an approximate worst solution that provides an efficient upper bound for $\mathrm{wor}_{\textsc{Max}\mathbf{P}_4\textsc{P}_{1,2}}(I)$.

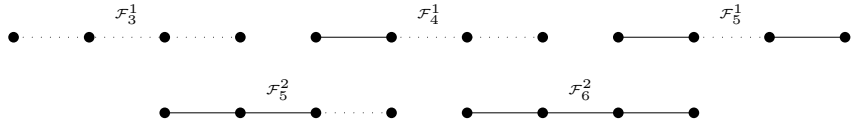


Figure 18.18. *A partition of $\mathcal{P}'$*

LEMMA 18.4.– *Let $I = (K_{4n}, w)$ be an instance of* MIN$\mathbf{P}_4\mathrm{P}_{1,2}$ *and let $\mathcal{P}'$ be the solution provided by Hassin and Rubinstein algorithm on I. We can calculate on I a solution $\mathcal{P}_*$ that verifies:*

$$p_* + q_* \leqslant q + (|\mathcal{F}_6^2| - \lfloor p_1^1/2 \rfloor)^+ + (|\mathcal{F}_6^2| - n + q)^+$$

where p_*, q_* and p_1^1 are defined as $p_* = |M_{\mathcal{P}_*,2}|$, $q_* = |R_{\mathcal{P}_*,2}|$ and $p_1^1 = |M_{\mathcal{P}'}^1 \cap M_{\mathcal{P}',1}|$ (and expression X^+ is equivalent to $\max\{X, 0\}$).

Proof. The proof is algorithmic, based on algorithm `Approximate Worst` $\mathtt{P}_4\mathtt{P}$. Note that, even though this has no impact on the rightness of the proof, the calculation of $\mathcal{P}_*$ has a polynomial runtime.

This means that the good properties of the approximate solution $\mathcal{P}'$ make it possible to really exhibit an approximate worst solution (and not only to provide an evaluation of such a solution, as it is often the case while stating differential approximation results).

In order to estimate the value of the approximate worst solution $\mathcal{P}_*$, we have to count the number $p_* + q_*$ of 2-weight edges it contains. Let p_i^1 refer to $|M_{\mathcal{P}'}^1 \cap M_{\mathcal{P}',i}|$ for $i = 1, 2$ (the cardinality p_1^1 enables the expression of the number of iterations during step 1). Steps 1, 2 and 3 respectively put into $\mathcal{P}_*$ at most one, two and three 2-weight edges per iteration.

Any path from $\mathcal{F}_6^2$ is treated by one of the three steps 1, 2 and 3. If $2|\mathcal{F}_6^2| \geqslant p_1^1$, only $|\mathcal{F}_6^2| - \lfloor p_1^1/2 \rfloor$ paths from $\mathcal{F}_6^2$ are treated by one of the steps 2 and 3. Finally, if $|\mathcal{F}_6^2| \geqslant |\mathcal{F}^1|$, only $|\mathcal{F}_6^2| - |\mathcal{F}^1|$ paths from $\mathcal{F}_6^2$ are treated during step 3. Furthermore, step 4 puts at most $|\mathcal{F}_5^2|$ 2-weight edges into $\mathcal{P}_*$ (at most one per iteration), while steps 0 and 5 do not incorporate any 2-weight edges within $\mathcal{P}_*$.

Thus, considering $q = |\mathcal{F}_5^2| + |\mathcal{F}_6^2|$ and $|\mathcal{F}^1| = n - q$, we obtain the announced result. ∎

Let us introduce some more notation. Analogously to $\mathcal{F}^2 = \mathcal{F}_5^2 \cup \mathcal{F}_6^2$, we define a partition of $\mathcal{F}^1$ into three subsets $\mathcal{F}_3^1$, $\mathcal{F}_4^1$ and $\mathcal{F}_5^1$ according to the path total weight.

Note that, since the subsets $\mathcal{F}_j^1$ define a partition of $\mathcal{P}'$, we have $n = |\mathcal{F}_3^1| + |\mathcal{F}_4^1| + |\mathcal{F}_5^1| + |\mathcal{F}_5^2| + |\mathcal{F}_6^2|$ (see Figure 18.18 for an illustration of this partition; the edges of

weight 2 are drawn in continuous lines whereas the edges of weight 1 are drawn in dotted lines).

`Approximate Worst` $\mathtt{P}_4\mathtt{P}$

0 Set $\mathcal{P} = \mathcal{P}'$, $\mathcal{P}_* = \{\,\}$;

1 While $\exists\, \{P, e_1, e_2\} \subseteq \mathcal{P}$ *s.t.* $(P, e_1, e_2) \in \mathcal{F}_6^2 \times M^1_{\mathcal{P}',1} \times M^1_{\mathcal{P}',1}$
  1.1 Calculate $\mathcal{F} = \{P_1, P_2\}$ on $V(P) \cup V(e_1) \cup V(e_2)$ with $w(\mathcal{F}) \leqslant 7$ according to Property 18.9;
  1.2 $\mathcal{P} \longleftarrow \mathcal{P} \setminus \{P, e_1, e_2\}$, $\mathcal{P}_* \longleftarrow \mathcal{P}_* \cup \{P_1, P_2\}$;

2 While $\exists\, \{P, e_1, e_2\} \subseteq \mathcal{P}$ *s.t.* $(P, e_1, e_2) \in \mathcal{F}_6^2 \times M^1_{\mathcal{P}'} \times M^1_{\mathcal{P}'}$
  2.1 Calculate $\mathcal{F} = \{P_1, P_2\}$ on $V(P) \cup V(e_1) \cup V(e_2)$ with $w(\mathcal{F}) \leqslant 8$ according to Property 18.9;
  2.2 $\mathcal{P} \longleftarrow \mathcal{P} \setminus \{P, e_1, e_2\}$, $\mathcal{P}_* \longleftarrow \mathcal{P}_* \cup \{P_1, P_2\}$;

3 While $\exists P \subseteq \mathcal{P}$ *s.t.* $P \in \mathcal{F}_6^2$
  3.1 $\mathcal{P} \longleftarrow \mathcal{P} \setminus P$, $\mathcal{P}_* \longleftarrow \mathcal{P}_* \cup \{P\}$;

4 While $\exists P \subseteq \mathcal{P}$ *s.t.* $P \in \mathcal{F}_5^2$
  4.1 Calculate $\mathcal{F} = \{P_1\}$ on $V(P)$ with $w(\mathcal{F}) \leqslant 4$;
  4.2 $\mathcal{P} \longleftarrow \mathcal{P} \setminus P$, $\mathcal{P}_* \longleftarrow \mathcal{P}_* \cup \{P_1\}$;

5 While $\exists\, \{e_1, e_2\} \subseteq \mathcal{P}$ *s.t.* $(e_1, e_2) \in M^1_{\mathcal{P}'} \times M^1_{\mathcal{P}'}$
  5.1 Calculate $\mathcal{F} = \{P_1\}$ on $V(e_1) \cup V(e_2)$ with $w(\mathcal{F}) = 3$;
  5.2 $\mathcal{P} \longleftarrow \mathcal{P} \setminus e_1, e_2$, $\mathcal{P}_* \longleftarrow \mathcal{P}_* \cup \{P_1\}$;

6 Output $\mathcal{P}_*$.

The following lemma states three relations between the couples of quantities (p, q), (p^*, q^*) and (p_*, q_*) that determine the value of the approximate solution, the considered optimal solution and the approximate worst solution, respectively.

LEMMA 18.5.–

$$p \geqslant q^* + (|\mathcal{F}_6^2| - \lfloor p_1^1/2 \rfloor)^+ \quad [18.21]$$

$$2q \geqslant q^* + (|\mathcal{F}_6^2| + q - n)^+ \quad [18.22]$$

$$q \geqslant p_* + q_* - (|\mathcal{F}_6^2| - \lfloor p_1^1/2 \rfloor)^+ - (|\mathcal{F}_6^2| + q - n)^+ \quad [18.23]$$

Proof. Inequality [18.21]: obvious if $|\mathcal{F}_6^2| \leqslant \lfloor p_1^1/2 \rfloor$, since $p \geqslant q^*$ (inequality [18.17]). Otherwise, we can write p as the sum $p = n + |\mathcal{F}_6^2| + |\mathcal{F}_5^1| - |\mathcal{F}_3^1|$. Then observe that $|\mathcal{F}_5^1| - |\mathcal{F}_3^1|$ is precisely the half of the difference between the number of 2-weight and of 1-weight edges in $M^1_{\mathcal{P}'}$: indeed, $p_2^1 = |\mathcal{F}_4^1| + 2|\mathcal{F}_5^1|$ and $p_1^1 = |\mathcal{F}_4^1| + 2|\mathcal{F}_3^1|$ and

thus $p_2^1 - p_1^1 = 2(|\mathcal{F}_5^1| - |\mathcal{F}_3^1|)$. From this latter equality, we deduce that p_1^1 and p_2^1 have the same parity, or, equivalently, that $(1/2)(p_2^1 - p_1^1) = \lfloor p_2^1/2 \rfloor - \lfloor p_1^1/2 \rfloor$. We deduce: $p = n + |\mathcal{F}_6^2| + \lfloor p_2^1/2 \rfloor - \lfloor p_1^1/2 \rfloor \geqslant n + |\mathcal{F}_6^2| - \lfloor p_1^1/2 \rfloor$. Just observe that $n \geqslant q^*$ (inequality [18.18]) in order to conclude.

Inequality [18.22]: obvious if $|\mathcal{F}_6^2| \leqslant n - q$, from inequality [18.16]. Otherwise, consider that $q \geqslant |\mathcal{F}_6^2|$ (by definition of q and $\mathcal{F}_6^2$) and $n \geqslant q^*$ (inequality [18.18]); therefore, $q \geqslant |\mathcal{F}_6^2| \geqslant |\mathcal{F}_6^2| + (q^* - n)$.

Inequality [18.23]: immediate from Lemma 18.5. ∎

THEOREM 18.12.– *The solution $\mathcal{P}'$ provided by the algorithm achieves a 2/3-differential approximation for $\mathbf{P}_4\mathrm{P}_{a,b}$ and this ratio is tight.*

Proof. By combining inequalities [18.21] to [18.23], together with $2p \geqslant 2p^*$, we obtain the expected result:

$$\begin{aligned} 3\mathrm{apx}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}}(I) &= 3(3n + p + q) \\ &\geqslant 2(3n + p^* + q^*) + (3n + p_* + q_*) \\ &= 2\mathrm{opt}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}_{1,2}}(I) + \mathrm{wor}_{\mathrm{MAX}\mathbf{P}_4\mathrm{P}_{1,2}}(I) \end{aligned}$$

The tightness is provided by the instance $I = (K_8, w)$ that is shown on Figure 18.13; since this instance contains some vertex v such that any edge from v is of weight 2, the result follows. ∎

18.5. Conclusion

Whereas both the complexity and the approximation status of bounded-size path packing problems in bipartite graphs with maximum degree 3 have been decided here, there remain some open questions: notably, the complexity of (INDUCED) $\mathbf{P}_k$PARTITION for $k \geqslant 4$ and the **APX**-hardness of MAX(INDUCED)$\mathbf{P}_k$PACKING for $k \geqslant 4$ in planar bipartite graphs with maximum degree 3. Those questions matter because, by drawing the precise frontier between "easy" and "hard" instances of those problems, they participate in a better understanding of what makes the problems tractable or intractable. However, it is also important to obtain better approximation bounds; in particular, concerning MAXW$\mathbf{P}_k$PACKING and MINk-PATHPARTITION: as we have already mentioned, there are no specific approximation results that exploit the specific structure of these problems. Even the results we propose here are obtained by means of quite naive algorithms; thus, we could expect better bounds using more sophisticated algorithms. Finally, an important question concerns the approximation of $\mathbf{P}_k$P, and may be more specifically that of MINMETRIC$\mathbf{P}_k$P, because of its relations to the minimum vehicle routing problem. We were here interested in the analysis of a given algorithm, but not really in the improvement of the approximation bounds for $\mathbf{P}_k$P.

However, we could expect better and, moreover, the following question remains open: does the problem admit a **PTAS**?

18.6. Bibliography

[ARK 98] ARKIN E., HASSIN R., "On local search for weighted packing problems", *Mathematics of Operations Research*, vol. 22, p. 640-648, 1998.

[ARK 06] ARKIN E. M., HASSIN R., LEVIN A., "Approximations for minimum and min-max vehicle routing problems", *Journal of Algorithms*, vol. 59, num. 1, p. 1-18, 2006.

[AUS 80] AUSIELLO G., D'ATRI A., PROTASI M., "Structure preserving reductions among convex optimization problems", *Journal of Computational System Sciences*, vol. 21, p. 136-153, 1980.

[AUS 99] AUSIELLO G., CRESCENZI P., GAMBOSI G., KANN V., MARCHETTI-SPACCAMELA A., PROTASI M., *Complexity and Approximation (Combinatorial Optimization Problems and Their Approximability Properties)*, Springer, Berlin, 1999.

[BEL 95] BELLARE M., ROGAWAY P., "The complexity of approximating a nonlinear program", *Mathematical Programming*, vol. 69, p. 429-441, 1995.

[BER 06] BERMAN P., KARPINSKI M., "$8/7$-Approximation Algorithm for $1,2$-TSP", *SODA'06*, Miami, Florida, United States, p. 641-648, January 2006.

[BLA 05] BLÄSER M., RAM L. S., "An Improved Approximation Algorithm for TSP with Distances One and Two", *FCT'05*, Lübeck, Germany, p. LNCS 3623 504-515, August 2005.

[CHA 00] CHAZELLE B., "A minimum spanning tree algorithm with Inverse-Ackermann type complexity", *Journal of the ACM*, vol. 47, p. 1028-1047, 2000.

[CSA 02] CSABA B., KARPINSKI M., KRYSTA P., "Approximability of dense and sparse instances of minimum 2-connectivity, TSP and path problems", *SODA*, p. 74-83, 2002.

[DEM 96] DEMANGE M., PASCHOS V. T., "On an approximation measure founded on the links between optimization and polynomial approximation theory", *Theoretical Computer Science*, vol. 158, p. 117-141, 1996.

[DYE 86] DYER M., FRIEZE A., "Planar 3DM is NP-complete", *J. Algorithms*, vol. 7, p. 174-184, 1986.

[FRA 76] FRANK A., "Some Polynomial Time Algorithms for Certain Graphs and Hypergraphs", *Proceedings of the 5th British Combinatorial Conference, Congressus Numerantium XV, Utilitas Mathematicae*, Winnipeg, p. 211-226, 1976.

[FRE 78] FREDERICKSON G. N., HECHT M. S., KIM C. E., "Approximation algorithms for some routing problems", *SIAM J. on Computing*, vol. 7, p. 178-193, 1978.

[GAR 79] GAREY M. R., JOHNSON D. S., *Computers and intractability. A guide to the theory of NP-completeness*, CA, Freeman, 1979.

[HAS 97] HASSIN R., RUBINSTEIN S., "An Approximation Algorithm for Maximum Packing of 3-Edge Paths", *Information Processing Letters*, vol. 63, p. 63-67, 1997.

[HAS 01] HASSIN R., KHULLER S., "z-approximations", *Journal of Algorithms*, vol. 41, p. 429-442, 2001.

[HAS 06] HASSIN R., RUBINSTEIN S., "An Approximation Algorithm for Maximum Triangle Packing", *Discrete Applied Mathematics*, vol. 154, num. 6, p. 971-979, 2006.

[KAN 03] KANEKO A., "A necessary and sufficient condition for the existence of a path factor every component of which is a path of length at least two", *Journal of Combinatorial Theory, Series B*, vol. 88, p. 195-218, 2003.

[KAR 06] KARPINSKI M., "Personal communication", 2006.

[KIR 78] KIRKPATRICK D., HELL P., "On the Completeness of a Generalized Matching Problem", *STOC'78*, p. 240-245, 1978.

[LOV 86] LOVASZ L., PLUMMER M., *Matching Theory*, North-Holland, Amsterdam, 1986.

[MON 02a] MONNOT J., "Differential approximation of **NP**-hard problems with constant size feasible solutions", *RAIRO/operations research*, vol. 36, num. 4, p. 279-297, 2002.

[MON 02b] MONNOT J., "Differential approximation results for the traveling salesman and related problems", *Information Processing Letters*, vol. 82, p. 229-235, 2002.

[MON 07a] MONNOT J., TOULOUSE S., Approximation results for the weighted P_4 partition problem, to appear in J. of Discrete Algorithms, 2007.

[MON 07b] MONNOT J., TOULOUSE S., Complexity of the P_k packing problem and related problems in bipartite graphs, to appear in Operations Research Letters, 2007.

[PET 94] PETRANK E., "The Hardness of Approximation: Gap Location", *Computational Complexity*, vol. 4, p. 133-157, 1994.

[STE 00] STEINER G., "On the k-Path partition problem in cographs", *Congressus Numerantium*, vol. 147, p. 89-96, 2000.

[STE 03] STEINER G., "On the k-path partition of graphs", *Theoretical Computer Science*, vol. 290, p. 2147-2155, 2003.

[VIS 92] VISHWANATHAN S., "An Approximation Algorithm for the Asymmetric Travelling Salesman Problem with Distances One and Two", *Information Processing Letter*, vol. 44, num. 6, p. 297-302, 1992.

[WAN 94] WANG H., "Path factors of bipartite graphs", *Journal of Graph Theory*, vol. 18, p. 161-167, 1994.

[YAN 97] YAN J.-H., CHANG G., HEDETNIEMI S., HEDETNIEMI S., "On the k-path partition of graphs", *Discrete Applied Mathematics*, vol. 78, p. 227-233, 1997.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Chapter 19

An Upper Bound for the Integer Quadratic Multi-knapsack Problem

19.1. Introduction

This chapter presents a method to calculate a good upper bound for the separable quadratic multi-knapsack problem $(QMKP)$, which is derived from the solution method developed by Djerdjour *et al.* [DJE 88]. The problem we examine is a generalization of the integer quadratic knapsack problem (QKP) which consists of maximizing a concave separable quadratic integer function subject to a single linear capacity constraint. Although there is a paucity of solution methods to (QKP), significant contributions may be found in other works. Among these, Mathur *et al.* [MAT 83] solve (QKP) to optimality by applying a piecewise linearization to the objective function to obtain an equivalent 0-1 linear problem. Bretthauer and Shetty suggest several effective methods, such as pegging algorithms [BRE 02b] and projection methods [BRE 96, BRE 02a] to solve the LP-relaxation of (QKP) so as to calculate an upper bound of the optimal value in a fast CPU time. The main application of (QKP) is in finance [MAT 83], [BRE 97] for the portfolio management problem can be formulated as a mathematical program with a quadratic objective function under a knapsack constraint [MAR 52]. The quadratic function measures both the expected return and the risk and the single knapsack constraint represents the budget constraint. The assumption of a single knapsack constraint does not allow the possibility of investing into assets of different risk levels. This can be formulated by means of several knapsack constraints, each representing a budget allocated to assets of a given risk level. We therefore face an integer quadratic multi-knapsack problem $(QMKP)$ which is a

Chapter written by Dominique QUADRI, Eric SOUTIF and Pierre TOLLA.

generalization of (QKP). This capital budgeting model is discussed in [DJE 88] and [FAA 74].

Formally, the integer (non-pure binary) quadratic multi-dimensional knapsack problem $(QMKP)$ can be written as:

$$(QMKP) \left\{ \begin{array}{l|l} \max & f(x) = \sum_{j=1}^{n} c_j x_j - d_j x_j^2 = \sum_{j=1}^{n} f_j(x_j) \\ s.t. & \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i, i = 1, ..., m \\ & 0 \leqslant x_j \leqslant u_j, j = 1, ..., n \\ & x_j \; integer \; j = 1, ..., n \end{array} \right.$$

where the coefficients c_j, d_j, a_{ij}, b_i are non-negative. The bounds u_j of variables x_j are integers such that $u_j \leqslant \frac{c_j}{2d_j}$; indeed, the separable objective function is concave which implies that for all f_j, $x_j^* \leqslant \frac{c_j}{2d_j}$, where x_j^* is the optimal solution of the program $\max_{0 \leqslant x_j \leqslant u_j} f_j(x_j)$.

The integer quadratic multi-dimensional knapsack problem $(QMKP)$ has received less attention in other works than (QKP). To the best of our knowledge, Djerdjour *et al.* [DJE 88] are the only authors to propose a specific solution method to solve $(QMKP)$. As such, their method is more effective than more general techniques that have been primarily developed to solve general integer quadratic programs (see [COO 81] and [KOR 85, KOR 90]). The method of Djerdjour *et al.* first consists of a piecewise linearization of the objective function which consequently converts $(QMKP)$ into an equivalent 0-1 multi-dimensional knapsack problem, (MKP), for which a wide range of methods exists (see for instance [FRÉ 86] and [CHU 98]). These methods are presented and analyzed in the recent survey of Fréville and Hanafi [FRÉ 05]. Djerdjour *et al.* then apply a surrogate relaxation to the m constraints of (MKP) in order to calculate an upper bound of the objective function of $(QMKP)$.

In this chapter we propose an upper bound that improves the surrogate relaxation originally proposed by Djerdjour *et al.* [DJE 88]. The bound is improved from both a qualitative and a calculational standpoint. We also develop a heuristic method to get a feasible solution to $(QMKP)$. As no numerical evaluation of the quality of the bounds for $(QMKP)$ is available in other works, we provide a theoretical and experimental comparison of the different bounds described in this chapter. We will compare the LP relaxation, a linearization, the surrogate relaxation [DJE 88] as well as the upper and lower bounds we propose. The objective of the calculational study we conduct in this chapter is to determine which bound is finally the most appropriate to be used in a branch-and-bound procedure to efficiently solve the problem. To this purpose we consider instances up to 2,000 variables and 2,000 constraints. Simulation results show that our method provides an upper bound of good quality in most

cases and is always better than the surrogate relaxation [DJE 88], while requiring a significantly less calculation time.

The chapter is organized as follows. Section 19.2 summarizes the algorithm proposed in [DJE 88] to calculate an upper bound of $(QMKP)$. In section 19.3, we present two improvements of this algorithm. The first improvement is meant to speed up the calculation of the bound and the second one increases the quality of the bound. A feasible solution is proposed in section 19.4. The calculational results are reported in section 19.5. In section 19.6 we summarize the main results of this chapter and we point out some directions for further research.

In the remainder of this chapter, we adopt the following notations: letting (P) be an integer or a 0-1 program, we will denote by $(\overline{P})$ the continuous relaxation problem of (P). We let $Z[P]$ be the optimal value of the problem (P) and $Z[\overline{P}]$ the optimal value of $(\overline{P})$.

19.2. The algorithm of Djerdjour *et al.*

The method proposed by Djerdjour *et al.* [DJE 88] is an exact method to solve $(QMKP)$. At each node of the search tree, an upper bound is calculated by solving a polynomial problem derived from $(QMKP)$. First, an equivalent formulation of $(QMKP)$ is obtained by using a direct expansion of the integer variables x_j as originally proposed by Glover [GLO 75] and by applying a piecewise linear interpolation to the initial objective function as discussed by Mathur *et al.* [MAT 83]. Consequently, $(QMKP)$ is equivalent to the 0-1 piecewise linear program (MKP):

$$(MKP)\ \begin{cases} \max \sum_{j=1}^{n}(\sum_{k=1}^{u_j} s_{jk}y_{jk}) = l(y) \\ s.t. \left| \begin{array}{l} \sum_{j=1}^{n}(a_{ij}\sum_{k=1}^{u_j} y_{jk}) \leqslant b_i,\ i = 1, ..., m \\ y_{jk} \in \{0, 1\} \end{array} \right. \end{cases}$$

where $y = (y_{jk})$ ($j = 1, ..., n$ and $k = 1, ..., u_j$), $\sum_{k=1}^{u_j} y_{jk} = x_j$, $s_{jk} = f_{jk} - f_{j,k-1}$ and $f_{jk} = c_j k - d_j k^2$.

In the second step of the algorithm, a surrogate relaxation is applied to the LP-relaxation of (MKP). The surrogate relaxation initially introduced by Glover [GLO 65] consists of aggregating the m initial linear constraints into a single constraint, namely a surrogate constraint, by replacing the set of constraints $Ay \leqslant b$ by a unique constraint $wAy \leqslant wb$, where A stands for the matrix of constraints of (MKP). The vector $w = (w_1, ..., w_i, ..., w_m)$ is non-negative and is called the surrogate multiplier. The resultant formulation (KP, w) is the surrogate relaxation of

(MKP) and is written as:

$$(KP,w)\ \left\{\begin{array}{l|l}\max & \sum_{j=1}^{n}(\sum_{k=1}^{u_j} s_{jk}y_{jk}) \\ s.t. & \sum_{j=1}^{n}[\sum_{i=1}^{m} w_i a_{ij}]\sum_{k=1}^{u_j} y_{jk} \leqslant \sum_{i=1}^{m} w_i b_i \\ & y_{jk}\in\{0,1\}\end{array}\right.$$

The above problem (KP,w) is a knapsack problem whose LP relaxation may efficiently be solved in $O(n^{'}log2(n^{'}))$ operations, where $n^{'}$ stands for the number of variables of (KP,w). The knapsack problem (KP,w) is one of the most common problems examined in the operations research works (see [MAR 90]). As proved by Glover [GLO 65], (KP,w) is a relaxation of (MKP). The proof relies on the fact that an optimal solution of (MKP) is feasible for (KP,w). For any value of $w \geqslant 0$ the optimal value $Z[\overline{KP,w}]$ of $(\overline{KP,w})$ is an upper bound of the optimal value $Z[\overline{MKP}]$ of $(\overline{MKP})$. Solving the dual surrogate problem: $\min_{w\geqslant 0} Z[\overline{KP,w}]$, denoted by (SD), leads to find the best upper bound $Z[\overline{KP,w^*}]$. Since the objective function of (SD) is quasi-convex the authors use a local descent method that provides a global minimum w^*.

For each value of $w \geqslant 0$ the authors solve the problem $(\overline{KP,w})$ using a quicksort algorithm whose complexity is $O(n^{'}log(n^{'}))$, where $n^{'}$ stands for the number of variables of (KP,w). However, a more efficient method exists whose complexity is $O(n^{'})$. This method was proposed by Fayard and Plateau [FAY 79] and is the one we finally adopted in the experiments instead of the quicksort algorithm.

19.3. Improving the upper bound

We present two improvements of the method in section 19.2 to calculate an upper bound for $(QMKP)$. We chose to keep the surrogate relaxation of (MKP) initially used by the authors although a Lagrangean relaxation could have been implemented. The rationale for using a surrogate relaxation rather than a Lagrangean relaxation stems from theoretical results which show the superiority of the former over the latter [FRÉ 05].

First, the local search descent method originally used by Djerdjour *et al.* to calculate the optimal surrogate multiplier is abandoned for a global method which is proved to be faster as evidenced by the calculational results presented in section 19.5. The second improvement proceeds from an additional stage in which we solve $(KP,w*)$ in 0-1 variables rather than in continuous variables. We finally establish an order relation between all the upper bounds included in the experiment for the sake of comparison.

The first improvement is derived from the following proposition.

PROPOSITION 19.1.– *If $w^* \geqslant 0$ is the dual optimal solution of $(\overline{MKP})$ then the optimal value of $(\overline{MKP})$ is equal to the optimal value of $(\overline{KP, w^*})$ i.e.:*

$$Z[\overline{MKP}] = Z[\overline{KP, w^*}] \qquad [19.1]$$

and w^* is an optimal surrogate multiplier for $(SD) = \min_{w \geqslant 0} Z[\overline{KP, w}]$.

The proof of Proposition 19.1 is given in Appendix A and relies on the fact that the optimal value of the dual problem of $(\overline{MKP})$ is an upper bound for $Z[\overline{KP, w^*}]$. In this special case there is therefore no (surrogate) duality gap between $Z[\overline{MKP}]$ and $Z[\overline{KP, w}]$.

From Proposition 19.1 an optimal vector w^* can be obtained by solving the dual of $(\overline{MKP})$ instead of using the local descent method suggested by Djerdjour *et al.* The numerical results presented in section 19.5 assess the calculational efficiency of this alternative way for calculating w^*.

To improve the upper bound $Z[\overline{KP, w^*}]$ we propose an additional stage in which we use w^* calculated as previously described. This stage consists of solving (KP, w^*) in 0-1 variables rather than in continuous variables. In other words we calculate the optimal value $Z[KP, w^*]$ instead of $Z[\overline{KP, w^*}]$. The surrogate multiplier w^* we calculate is not the optimal surrogate multiplier for the integer problem (KP, w^*). We could have searched for this optimal vector using the method of Fréville and Plateau [FRÉ 93] but the resultant additional quality would not have been offset by the time required.

REMARK 19.1.– If the optimal solution of $(KP, w*)$ is feasible for $(QMKP)$ then the value $Z[KP, w^*]$ is the optimal value of $(QMKP)$.

It follows from Remark 19.1 that the value of the bound will actually be the optimal value of several instances in our experiments. Traditionally the optimal value $Z[\overline{QMKP}]$ of the continuous relaxation of $(QMKP)$ is used as an upper bound for $(QMKP)$. This value $Z[\overline{QMKP}]$ can easily be calculated by using a commercial software, since it is a concave problem (the quadratic and separable objective function is positive semi-definite and the feasible set is convex). The following proposition shows that the upper bound of Djerdjour *et al.* [DJE 88] and our improved upper bound are always better than $Z[\overline{QMKP}]$.

PROPOSITION 19.2.– *The optimal value of the continuous relaxation of (MKP) is never worse than the optimal value of the continuous relaxation of (QMKP), i.e.:* $Z[\overline{MKP}] \leqslant Z[\overline{QMKP}]$

A proof of Proposition 19.2 is straightforwardly obtained by noting that the optimal solution of the continuous relaxation of (MKP) is feasible for the continuous relaxation of $(QMKP)$.

19.4. An efficient heuristic to calculate a feasible solution

The lower bound we propose is based on the fact that we know from Remark 19.1 that $Z[\overline{MKP}]$ is tighter than $Z[\overline{QMKP}]$. Consequently, we suggest to use as a starting point the optimal solution of $(\overline{MKP})$ to construct a feasible solution for $(QMKP)$.

The main idea of the proposed heuristic is the following. We first consider the optimal solution y^* of $(\overline{MKP})$. Letting $\sum_k y^*_{jk} = \alpha_j$, for each variable x_j of $(QMKP)$, we add to $(QMKP)$ the constraint $\lfloor \alpha_j \rfloor \leqslant x_j \leqslant \lfloor \alpha_j \rfloor + 1$, where $\lfloor \alpha_j \rfloor$ denotes the greatest integer smaller than or equal to α_j. Thus, each variable becomes bivalent and, since the objective function is separable, it can straightforwardly be shown that the resulting problem is a 0-1 linear multi-dimensional knapsack problem. Obviously, solving this knapsack problem yields a feasible solution for $(QMKP)$ which is not necessarily optimal for $(QMKP)$.

19.5. Computational results

In this section we report the calculational results of comparing the performance of each upper bounds of $(QMKP)$ described in this chapter to that of the lower bound proposed in section 19.4. Since no benchmark for $(QMKP)$ is available nowadays, we consider three types of randomly generated instances endowing each a particular structure: squared problems (n=m), rectangular problems (m=0.05n) and correlated problems $c_j = \sum_{i=1}^{m} a_{ij}$ and $d_j = c_{min}/2$ where c_{min} is the minimum of all c_j values. The rationale for using correlated problems stems from the fact that they are difficult to solve in practice for 0-1 linear multi-dimensional knapsack problems (MKP), which are a special case of $(QMKP)$.

As in [DJE 88] integer coefficients , a_{ij}, c_j and d_j were uniformly drawn at random in the range $\{1, ..., 100\}$. Coefficients b_i and u_j are integers uniformly distributed such that $b_i \in [50, \sum_{i=1}^{m} a_{ij} u_j \; u_j \in [1, \lceil c_j/2d_j \rceil]$, where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. For the correlated problems, c_j and d_j are derived from a_{ij}, whereas they are randomly generated according to a uniform law in the range $\{1, ..., 100\}$ for squared and rectangular problems.

To assess the quality of the four upper bounds we used our lower bound to calculate the relative gap (Gap = (upper bound - lower bound)/(lower bound)) since this lower bound was near optimal in most of the instances considered by Djerdjour *et al.* [DJE 88]. Our lower bound is actually the value of the best known feasible solution. The simulation results show that our feasible solution was better than the 3 feasible solutions provided by Djerdjour *et al.* [DJE 88] in 66% of the instances, equal in 33% of them and worse in 1%. Our lower bound is on average 3% higher than the best of the three feasible solutions which are already closed to the optimum (see [DJE 88] for

Instances	$\mathbf{Z}[\overline{\mathbf{QMKP}}]$ **Cplex9.0**	$\mathbf{Z}[\overline{\mathbf{MKP}}] = \mathbf{Z}[\overline{\mathbf{KP}, \mathbf{w}^*}]$ **Cplex9.0 or [DMS88]**	$[\mathbf{KP}, \mathbf{w}^*]$ **Our approach**
nb var,nb const	**Gap** $(\%)$	**Gap** $(\%)$	**Gap** $(\%)$
Rectangular			
100,5	2.6	0.5	0.3
500,25	2.8	0.4	0.3
1000,50	3.8	0.6	0.5
1500,75	5.5	1.6	1.5
2000,100	5.3	1.0	0.9
Correlated			
100,100	56.3	14.8	11.2
500,500	81.5	20.9	17.3
1000,1000	80.2	19.0	14.9
1500,1500	111.3	31.9	23.1
2000,2000	128.8	37.6	33.4
Squared			
100,100	16.9	9.5	8.2
500,500	12.9	7.9	7.5
1000,1000	32.2	23.0	21.7
1500,1500	37.8	24.6	23.9
2000,2000	53.0	36.9	36.2

Table 19.1. *Comparison of the quality of the upper bounds*

more details). Our lower and upper bounds as well as the upper bound of Djerdjour *et al.* were coded in C language. The two other upper bounds $Z[\overline{MKP}]$ and $Z[\overline{QMKP}]$ were obtained using the commercial software ILOG-Cplex9.0. Simulations were run on a bi-xeon 3.4 Ghz with 4Go of main memory.

Table 19.1 displays the average deviation of each upper bound to the feasible solution over 10 replications of each type of instances. For example, $Z[\overline{QMKP}]$ is on average 56.3% higher than our feasible solution over 10 replications of correlated problems with 100 variables and 100 constraints. The last column provides the percentage of instances for which our upper bound corresponds to the optimum value (see Remark 19.1). It appears that our bound behaves quite well for the rectangular problems for which the overall gap is less than 1.6%.

The quality of the upper bound is lower for squared and correlated problems with a gap ranging from 7% to 33%. However, our upper bound significantly outperforms the continuous relaxation of $(QMKP)$ in all cases. Our upper bound is also better than the upper bound $Z[\overline{KP, w^*}]$ with a maximum improvement of 4.2% for the squared problems (1,000,1,000).

Instances nb var,nb const	$Z[\overline{QMKP}]$ Cplex9.0	$Z[\overline{MKP}]$ Cplex9.0	$Z[\overline{KP}, w^*]$ [DMS88]	$[KP, w^*]$ Our approach
Rectangular				
100,5	0.0	0.0	0.3	0.0
500,25	7.5	0.1	10.1	0.1
1000,50	55.3	0.3	41.7	0.4
1500,75	193.1	0.8	100.3	0.8
2000,100	437.9	1.6	183.3	1.8
Correlated				
100,100	0.0	0.0	0.0	0.0
500,500	0.0	0.0	0.5	0.0
1000,1000	0.2	0.0	1.5	0.1
1500,1500	0.8	0.1	3.7	0.2
2000,2000	2.2	0.2	7.6	0.4
Squared				
100,100	0.0	0.0	0.3	0.0
500,500	7.3	0.1	9.0	0.2
1000,1000	58.2	0.5	37.9	0.5
1500,1500	184.5	1.5	86.6	1.6
2000,2000	421.3	3.4	157.8	3.6

Table 19.2. *Comparison of the CPU times (s) required for each upper bound*

The lowest gap is obtained for rectangular problems which suggests that our method performs best for this type of instances. We strongly believe that this good performance is more the result of an increased quality of our upper bound than that of an improvement of the quality of the lower bound. Indeed, as previously mentioned, simulations results showed that the lower bound is always near optimal for any of the three types of instances. Moreover, fewer constraints are aggregated in this type of instances so less information is lost. This could explain an improvement of the quality of our upper bound in this context.

Table 19.2 displays the CPU time in seconds required to calculate the four upper bounds. The most time-consuming bound is the continuous relaxation with a maximum of about 8 minutes to solve one of the largest correlated problems. The fastest bound is $Z[\overline{MKP}]$ with almost instantaneous results for rectangular problems and an average of 3.4 seconds for the largest squared problems.

The time to calculate our upper bound deviates at most 0.2 seconds from the time to obtain $Z[\overline{MKP}]$. Our method can therefore be considered as fast as the previous one. The advantage of calculating w* by solving the dual of $(\overline{MKP})$ rather than using a descent local method as suggested by Djerdjour *et al.* [DJE 88] strikingly appears: CPU times are sometimes divided by 100.

19.6. Conclusion

In this chapter we have designed a method to calculate a good upper bound for $(QMKP)$ and we have compared this bound to three other bounds over a large number of instances. The numerical results clearly show that our method provides the best upper bound in a very competitive calculation time compared to the linearization which is the quickest method. The proposed upper bound could therefore be used in an exact solution method. The calculational study also evidenced the good quality of our feasible solution which could consequently be used as an initial solution in a branch-and-bound method. It is worth mentioning that the continuous relaxation of $(QMKP)$, although widely used in practice, is not an efficient method from either a qualitatively or a calculational standpoint. A possible way to get a further improvement of the upper bound would be to use a composite relaxation including both a Lagrangean and a surrogate relaxation of the initial problem as suggested by Fréville and Hanafi [FRÉ 05], who present several methods to solve the 0-1 multi-dimensional knapsack problem.

Appendix A: proof of Proposition 19.1

Let c and A respectively denote the cost vector and the constraints matrix of $(\overline{MKP})$. Let us also respectively denote by e and I the unit vector (${}^t e = (1, \ldots, 1)$) and the identity matrix. $(\overline{MKP})$ and its dual problem $(\overline{DMKP})$ can be written as:

$$(\overline{MKP}) \begin{cases} \max\ cy \\ s.t. \left| \begin{array}{lcll} Ay & \leqslant & b & (\rightarrow \text{dual var. } u) \\ y & \leqslant & e & (\rightarrow \text{dual var. } v) \\ y \geqslant 0 & & & \end{array} \right. \end{cases}$$

$$(\overline{DMKP}) \begin{cases} \min\ g(u,v) = ub + ve \\ s.t. \left| \begin{array}{lcl} uA + Iv & \geqslant & c \\ u \geqslant 0,\ v \geqslant 0 & & \end{array} \right. \end{cases}$$

Let us recall the dual surrogate problem $(SD) : \min_{w \geqslant 0} Z(\overline{KP, w})$ where $(\overline{KP, w})$ is the following knapsack problem:

$$(\overline{KP, w}) \begin{cases} \max\ h(y) = cy \\ s.t. \left| \begin{array}{l} wAy \leqslant wb \\ 0 \leqslant y \leqslant e \\ \text{with} \end{array} \right. \end{cases}$$

We first prove the following statement: $\forall (u, v)$ feasible for $(\overline{DMKP}),\ g(u, v) \geqslant Z(\overline{KP, u})$.

Let (u, v) be a feasible solution for $(\overline{DMKP})$ and y_u the optimal solution for $(\overline{KP, u})$:

$$\begin{array}{rcl} g(u,v) = ub + ve & \geqslant & uAy_u + ve \\ & & (y_u \text{ feasible for } (\overline{KP,u}) \Rightarrow ub \geqslant uAy_u) \\ & \geqslant & cy_u - vy_u + ve \\ & & ((u,v) \text{ feasible for } (\overline{DMKP}) \Rightarrow uA \geqslant c - v \text{ and } y_u \geqslant 0) \\ & \geqslant & cy_u + v(e - y_u) \\ & \geqslant & cy_u = Z(\overline{KP,u}) \\ & & (0 \leqslant y_u \leqslant e) \end{array}$$

Now let (u^*, v^*) and w^* respectively be the optimal solution for $(\overline{DMKP})$ and for (SD). By duality in linear programming, we know: $Z(\overline{DMKP}) = Z(\overline{MKP})$. So the following inequality holds :

$$\begin{array}{rcl} Z(\overline{MKP}) & = & Z(\overline{DMKP}) = g(u^*, v^*) \geqslant Z(\overline{KP, u^*}) \\ & \geqslant & \min\limits_{w \geqslant 0} Z(\overline{KP, w}) = Z(\overline{KP, w^*}) \end{array}$$

Since $(\overline{KP, w^*})$ is a (surrogate) relaxation of $(\overline{MKP})$, we know that $Z(\overline{MKP}) \leqslant Z(\overline{KP, w^*})$. This yields the following result: $Z(\overline{MKP}) = Z(\overline{DMKP}) = Z(\overline{KP, w^*})$. Moreover: $Z(\overline{DMKP}) = g(u^*, v^*) \geqslant Z(\overline{KP, u^*}) \geqslant Z(\overline{KP, w^*})$. Since we know that $Z(\overline{DMKP}) = Z(\overline{KP, w^*})$, it follows that u^* is an optimal multiplier for the surrogate dual problem (SD).

19.7. Bibliography

[BRE 96] BRETTHAUER K., SHETTY B., "A projection method for the integer quadratic knapsack problem", *Journal of Operational Research Society*, vol. 47 (3), p. 457-463, 1996.

[BRE 97] BRETTHAUER K., SHETTY B., "Quadratic resource allocation with generalized upper bounds", *Operation Research Letters*, vol. 20 (2), p. 51-57, 1997.

[BRE 02a] BRETTHAUER K., SHETTY B., "The nonlinear knapsack problem - algorithms and applications", *European Journal of Operational Research*, vol. 138(3), p. 459-472, 2002.

[BRE 02b] BRETTHAUER K., SHETTY B., "A pegging algorithm for the nonlinear resource allocation problem", *Computers and Operations Research*, vol. 29 (5), p. 505-527, 2002.

[CHU 98] CHU P., BEASLEY J., "A genetic algorithm for the multi-dimensional knapsack problem", *Journal of Heuristics*, vol. 4, p. 63-86, 1998.

[COO 81] COOPER M., "A survey of methods for pure nonlinear integer programming", *Management Science*, vol. 3, p. 353-361, 1981.

[DJE 88] DJERDJOUR M., MATHUR K., SALKIN H., "A surrogate-based algorithm for the general quadratic multi-dimensional knapsack", *Operation Research Letters*, vol. 7 (5), p. 253-257, 1988.

[FAA 74] FAALAND B., "An integer programming algorithm for portfolio selection", *Management Science*, vol. 20 (10), p. 1376-1384, 1974.

[FAY 79] FAYARD D., PLATEAU G., "Reduction algorithms for single and multiple constraints 0-1 linear programming problems", *International conference Methods of Mathematical Programming proceedings*, 1979.

[FRÉ 86] FRÉVILLE A., PLATEAU G., "Heuristics and Reduction Methods for Multiple Constraints 0-1 Linear Programming Problems", *European Journal of Operational Research*, vol. 24, p. 206-215, 1986.

[FRÉ 93] FRÉVILLE A., PLATEAU G., "An exact search for the solution of the surrogate dual of the 0-1 bidimensional knapsack problem", *European Journal of Operational Research*, vol. 68, p. 413-421, 1993.

[FRÉ 05] FRÉVILLE A., HANAFI S., "The multi-dimensional 0-1 Knapsack Problem-Bounds and Computational Aspects", *Annals of Operations Research*, vol. 139, p. 195-227, 2005.

[GLO 65] GLOVER F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem", *Operations Research*, vol. 13, p. 879-919, 1965.

[GLO 75] GLOVER F., "Improved linear programming formulations of nonlinear integer problems", *Management Science*, vol. 22(4), p. 455-460, 1975.

[KOR 85] KORNER F., "Integer quadratic programming", *European Journal of Operational Research*, vol. 19(2), p. 268-273, 1985.

[KOR 90] KORNER F., "On the numerical realization of the exact penalty method for quadratic programming algorithms", *European Journal of Operational Research*, vol. 46(3), p. 404-408, 1990.

[MAR 52] MARKOWITZ H., "Portfolio Selection", *Journal of Finance*, vol. 7(1), p. 77-91, 1952.

[MAR 90] MARTELLO S., TOTH P., *Knapsack Problems: algorithms and computer implementations*, John Wiley & Sons, New York, USA, 1990.

[MAT 83] MATHUR K., SALKIN H., MORITO S., "A branch and search algorithm for a class of nonlinear knapsack problems", *Operations Research Letters*, vol. 2(4), p. 155-160, 1983.

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

List of Authors

Mohamed Ali ALOULOU
LAMSADE, Université Paris-Dauphine, France

Eric ANGEL
IBISC, Université d'Evry, France

Giorgio AUSIELLO
Dipartimento di Informatica e Sistemistica, Universitá di Roma "La Sapienza", Italy

Evripidis BAMPIS
IBISC, Université d'Evry, France

Beatrice BÉRARD
LAMSADE, Université Paris-Dauphine, France

Giuliana CARELLO
D.E.I., Politecnico di Milano, Italy

Franck CASSEZ
IRCCyN, Ecole Centrale de Nantes, France

Yann CHEVALEYRE
LAMSADE, Université Paris-Dauphine, France

Federico DELLA CROCE
D.A.I, Politecnico di Torino, Italy

Marc DEMANGE
SID, ESSEC, France

Gabriele DI STEFANO
D.I.E.I, Università dell'Aquila, Italy

Ulle ENDRISS
ILCC, University of Amsterdam, The Netherlands

Bruno ESCOFFIER
LAMSADE, Université Paris-Dauphine, France

Virginie GABREL
LAMSADE, Université Paris-Dauphine, France

Aristotelis GIANNAKOS
LAMSADE, Université Paris-Dauphine, France

Laurent GOURVÈS
LAMSADE, Université Paris-Dauphine, France

Andrea GROSSO
D.I., Università degli Studi di Torino, Italy

Serge HADDAD
LAMSADE, Université Paris-Dauphine, France

Marcin KAMIŃSKI
Algorithms Research Group, Université Libre de Bruxelles

Michel KOSKAS
LAMFA, Université de Picardie, France

Benjamin LEROY-BEAULIEU
ROSE, Ecole Polytechnique Fédérale de Lausanne

Didier LIME
IRCCyN, Ecole Centrale de Nantes, France

Marco LOCATELLI
D.I., Università degli Studi di Torino, Italy

Giorgio LUCARELLI
Department of Informatics, Athens University of Economics and Business, Greece

Nicolas MAUDET
LAMSADE, Université Paris-Dauphine, France

Martin MILANIČ
RUTCOR, Rutgers University, USA

Ioannis MILIS
Department of Informatics, Athens University of Economics and Business, Greece

Jérôme MONNOT
LAMSADE, Université Paris-Dauphine, France

Cécile MURAT
LAMSADE, Université Paris-Dauphine, France

Vangelis Th. PASCHOS
LAMSADE, Université Paris-Dauphine, France

Olivier POTTIÉ
LAMSADE, Université Paris-Dauphine, France

Dominique QUADRI
LIA, Université d'Avignon, France

Olivier-Henri ROUX
IRCCyN, Ecole Centrale de Nantes, France

Eric SOUTIF
CEDRIC, CNAM, France

Pierre TOLLA
LAMSADE, Université Paris-Dauphine, France

Sophie TOULOUSE
LIPN, Université Paris XIII, France

Dominique DE WERRA
ROSE, Ecole Polytechnique Fédérale de Lausanne

Combinatorial Optimization and Theoretical Computer Science: Interfaces and Perspectives
Edited by Vangelis Th. Paschos
Copyright © 2008, ISTE Ltd.

Index

2-CSP, 205
ε-approximation, 436
algorithm, 203
  exact, 203
  exponential, 203
  Hassin and Rubinstein, 478
approximate
  differential approximate, 458
approximation level
  APX, **DAPX**, 459
  constant approximable, 459
  polynomial-time approximation scheme, 459
  PTAS, **DPTAS**, 459
approximation ratio, 71, 76, 78, 89, 260
  differential ratio, 458
  standard ratio, 458
approximation theory
  standard approximate, 457, 458
APX, 435
  MIN3-PATHPARTITION, 474
  MAX$\mathbf{P}_3$PACKING, 473
  MAXW$\mathbf{P}_3$PACKING, 467, 471
  MAX$\mathbf{P}_4$P, 482
  MINMETRIC$\mathbf{P}_4$P, 480
  MAX$\mathbf{P}_4\mathrm{P}_{1,2}$, 484
  MIN$\mathbf{P}_4\mathrm{P}_{1,2}$, 484
APX-complete, 436, 438, 446
APX-hard
  MAXINDUCED$\mathbf{P}_k$PACKING, 465
  MAX$\mathbf{P}_k$PACKING, 465
Balanced biclique problem, 400
Balanced satisfiability problem, 436
  MAX BALANCED 2-SAT, 436
  MIN BALANCED 2-SAT, 436
Bipartite, 435
  2-regular bipartite, 435, 436
  3-regular bipartite, 435, 444
  Complete bipartite, 435, 446
branch-and-bound, 211
budget, 73
  model, 75
budget saving, 73, 75, 88, 89
Characteristic graph, 441
Chromatic index, 260
Chromatic number ($\chi(G)$), 260
Circular Arc Coloring, 286
Clique, 400
Clique (maximum, $\omega(G)$), 263
Clique tree, 410
Clique-width, 395, 422, 424
Clique-width expression, 422
coalition, 341, 342
color, 433
coloring, 259
  Brook's theorem, 210
  k-colorable graphs, 266
competitive ratio, 72, 73, 76–78, 80, 82, 86, 87
complementarity,substitutability, 182
complexity, 203, 211
  exponential-time, 203
  polynomial-space, 203, 211
  time, 203

concave function, 345
congestion, 355, 360–364, 368
congestion model, 355, 360–364, 367
continuous relaxation, 208
convergence, 180
  (maximal classes guaranteeing), 191
  associated decision problems, 188
  guaranteed, 181, 184, 185
convex function, 345
convex set, 345
counting problem, 344
cut, 204
  pseudo-boolean function, 206
  weight, 204
cyclic strategies, 165
DAPX
  MAX$\mathbf{P}_k$P, 477
  $\mathbf{P}_4$P, 482
  $\mathbf{P}_4\mathrm{P}_{a,b}$, 491
decidability, 343
degree, 204
  average, 204
  maximum, 204
direct expansion, 497
dominance, 39, 211–214, 228
dominating set, 91
Dynamic programming algorithm, 405, 407–409, 413, 415, 417, 423
equilibrium
  $(L)\cdot$, 342
  $\mathcal{K}$, 343
  $\mathcal{L}(K)$, 342
  in the strong sense, 343
  mixed Nash, 351
  Nash, 343, 345–349, 351, 357, 363, 364, 368
  Nash ε-approximate, 348
  pure Nash, 357, 363, 364, 368
Exact arborescence problem, 393
Exact cut problem, 393
Exact matching problem, 394, 428
Exact perfect matching problem, 393–395, 399, 423, 424
exact perfect matching problem, 394
Exact spanning tree problem, 393
Exact weighted independent set problem (EWIS), 394, 395, 397, 399, 406, 425, 428
Exact weighted maximum independent set problem (EWIS$_\alpha$), 394, 395, 397, 400, 404
EXPP, 343
finite improvement property, 357
fixpoint, 347
FNP, 344
Fully polynomial time approximation scheme (FPTAS), 261
game, 339, 340
  concave, 345, 347
  congestion, 360–363, 367
  cooperative, 339, 345
  coordination, 359, 361, 363
  defined on a graph, 350
  exact potential, 356, 359, 360, 363, 368
  fictitious, 359, 362, 363
  isomorphism, 360–363
  mixed extension, 347, 348
  network congestion, 363, 364
  non-cooperative, 340–342, 345
  ordinal potential, 356, 357
  potential, 356, 359
  quasi-concave, 345, 346
  symmetrical two-player, 351, 354, 355
  transformation, 350
  transformation of, 347
  two-player, zero-sum, 351
  weighted potential, 356
graph theory
  2-colorable, bipartite, 459
  chordal, chordless, 459
  interval graph, 459
  planar, 459
Graphs, 396
  AT-free, 395, 413
  Bipartite, 261, 273, 393–395, 399, 400
  (*bull*, *fork*)-free, 428
  Chordal, 281, 395, 399, 410
  Circle, 395, 409, 415
  $(claw, net)$-free, 395, 415
  (*co-P*, *fork*)-free, 428
  Co-comparability, 413
  Cographs (P_4-free), 261, 284

Distance-hereditary, 395, 415
Forests, 399
Interval, 261, 285, 395, 407
k-thin, 395, 408
k-tree, 261, 270
Line graph, 261, 394, 395, 399
Line graphs of bipartite, 396, 399, 424, 429
mK_2-free, 395, 407
(P_5, *co-P*)-free, 428
P_k-free bipartite, 261, 273
(P_5, *double-gem*, *co-domino*)-free, 428
Perfect, 403, 413
Permutation, 413
(P_5, *fork*)-free, 428
Planar, 261, 270
prime, 425
Split, 261, 280
Triangle free (K_3-free), 261, 270
Greedy algorithm, 435, 449
independence or irrelevant choices, 365
independent set, 211, 394
maximum independent set, 466
maximum independent set ($\alpha(G)$), 394, 402
maximum weighted independent set, 466
MAXWIS, 466
Turan's theorem, 211
individual rationality, 179
Induced subgraph, 396, 403, 415
influence graph, 349
integer linear program
continuous relaxation, 208
integer quadratic multidimensional knapsack, 496
Kleene's star, 343
knapsack problem, 498
Label, 433
Label squaring instance, 439
Label squaring operation, 439
Labeled perfect matching problem, 435
LABELED $Max\ PM$, 435
LABELED $Min\ PM$, 435
LABELED $Min\ PM_r$, 435
LABELED $Min\ PPM$, 439
Labeled problem (LABELED Π), 433
Max problem (LABELED Max Π), 433
Min problem (LABELED Min Π), 433
language, 343
linear program, 208
linear space, 345
linearization, 497
linearization technique, 208
List Coloring, 268
Local optimum, 44
Local search, 44
Master-slave, 264
matching, 394, 435
maximum *k*-dimensional matching problem, 460
3-dimensional matching, 464
PLANAR 3DM, 464
k-dimensional matching, 460, 464
k-dimensional matching problem, 460
*k*DM, 460
MAX*k*DM, 460
perfect matching, 394, 435
matrix
determinant, 208
totally unimodular, 208
MAX-2-SAT, 205
MAX-CUT, 204
unweighted, 204
weighted, 204
MAX-CUT-3, 211, 214
MAX COLORING, 259
maximum cut, 204
maximum independent set, 204
MAX**P**$_3$PACKING
`P3Packing`, 473
MAXW**P**$_3$PACKING
`Tree-P3PackingCover`, 468
`Weighted P3Packing`, 471
metric instance
bi-valuated instance, 456
MIN 3-COLORING, 227
MIN 3-DOMINATING SET, 204, 226
MIN 3-SET COVER, 227
MIN3-PATHPARTITION
`Minimum 3Path Partition`, 475
minimum dominating set, 204
minimum graph coloring, 204

minimum vertex covering, 204
MIN WEIGHTED NODE COLORING, 259
Modular decomposition, 424, 425
Multi-criteria optimization, 38
multiagent strategy, 163
Nearest neighbor, 48, 60
negotiation protocol, 185
  Contract-Net, 181
  deals, 178
  monetary transfers, 179
Neighborhood, 44
NEXP, 343, 354
node idleness, 163
Non-approximability, 42, 56
NP, 203, 343, 351, 354
NP-complete, 351
  INDUCED$\mathbf{P}_k$PARTITION, 463
  INDUCED$\mathbf{P}_3$PARTITION, 464
  $\mathbf{P}_k$PARTITION, 463
  $\mathbf{P}_3$PARTITION, 464
NP-hard, 71, 203, 354, 442
  MAX$\mathbf{P}_k$PACKING, 463
  MAX$\mathbf{P}_3$PACKING, 464
  MINk-PATHPARTITION, 463
  MIN3-PATHPARTITION, 464
  problem, 203
offline, 71, 77
  algorithm, 72, 75, 82, 89, 91
online, 71
  algorithm, 72, 74, 77, 79, 82
  calculation, 71
  model, 72, 73, 75, 77, 79, 82, 89
  set-covering, 72
optimality, 341, 342, 345, 347
  $\mathcal{L}(K)$, 342
  Pareto $\mathcal{K}$, 342
ordinal potential, 356
$\sharp\mathbf{P}$, 344, 355
P, 203, 343
  MAX$\mathbf{P}_k$PACKING, 466
packing
  maximum $\mathbf{P}_k$ packing problem, 456
  MAXINDUCED$\mathbf{P}_k$PACKING, 456
  MAX$\mathbf{P}_k$PACKING, 456
  MAXWINDUCED$\mathbf{P}_k$PACKING, 456
  MAXW$\mathbf{P}_k$PACKING, 456
  $\mathbf{P}_k$ packing, 456
  weighted $\mathbf{P}_k$ packing, 456
  weighted $\mathbf{P}_k$ packing, 456
    maximum weighted $\mathbf{P}_k$ packing problem, 456
Pareto curve, 39
  ε-approximate, 40, 42, 62
Pareto optimality, 39, 342, 343
Pareto stability, 342, 354
partition
  INDUCED$\mathbf{P}_k$PARTITION, 455
  k-path partition, 456
  k-path partition, 456
  minimum k-path partition problem, 456
  MINk-PATHPARTITION, 456
  $\rho_{k-1}(G)$, 456
  MIN2-PATHPARTITION, 456
  MINPATHPARTITION, 456, 474
  $\rho(G)$, 456, 474
  partition into isomorphic subgraphs, 455
  path partition, 456, 474
  weighted $\mathbf{P}_k$ partition problem, 456
  MINMETRIC$\mathbf{P}_k$P, 456
  $\mathbf{P}_k$PARTITION, 455
  $\mathbf{P}_k$ partition, 455
    $\mathbf{P}_k$ partitioning problem, 455
  weighted $\mathbf{P}_k$ partition, 456
partition-based strategies, 169
patrolling problem, 164
payoff function, 341, 346–349, 351–354, 356, 360, 364, 367
$\mathbf{P}_k$P
  deleting and turning around, 475
PLS, 364
polyLog-APX, 435, 445
polynomial space, 210
Polynomial time approximation scheme (PTAS), 260
polynomially balanced, 343
potential
  ω, 356
  generalized ordinal, 358
PPAD, 344, 350
PREXT NODE COLORING, 273
preference, 340, 341, 368
  strict, 341
Preference relation, 44

prisoner's dilemma, 343
problem
  maximum cut, 204
  maximum independent set, 204
  minimum dominating set, 204
  minimum graph coloring, 204
  minimum vertex covering, 204
  satisfiability, 204
pruning, 226
Pruning sequence, 415
pseudo-boolean function, 207
Pseudo-polynomial time, 393, 395, 398, 406–408, 410, 414, 415, 418, 422, 424, 428
quadratic program, 207
quasi-concave function, 345, 346
quasi-convex function, 345
rational agents, 176
Reduction, 438, 440
  L-reduction, 436, 446
reoptimization, 76
resource allocation, 175
  combinatorial auctions, 175, 198
  distributed, 175
resource allocation function, 360–363, 367
3-SAT PLANAR, 270
satisfiability, 204
Satisfiability problem, 438
  MAXSAT, 438
  MINSAT, 438
search tree, 204, 211, 213, 226, 227
  algorithm, 213
set cover, 71, 72, 441
  3-set cover, 473
  minimum 3-set cover problem, 473
situation, 341, 345
  $\mathcal{L}(K)$-optimal, 342
  attainable, 341
social welfare, 176, 179, 353, 354
specific instance
  metric instance, 456
    bi-valuated instance, 456
  planar graph, 459
Stable set, 259, 394
strategy, 341, 346–349, 355, 359, 361, 363, 364, 368
  mixed, 347, 348, 351, 352, 354, 355
  pure, 357, 361
Strong NP-completeness, 395, 400, 404
surrogate relaxation, 497
TF, 344, 350
TFNP, 344, 355
total problem, 344
total unimodularity, 208
traveling salesman problem, 39
  Hamiltonian cycle, 475
  MAXTSP, 475
  multi-criteria, 39
Tree-decomposition, 418
Treewidth, 395, 400, 418, 422
Turing machine, 343
utility (theory of), 341
utility functions, 176
  k-additivity, 189
  (classes of), 184
  additivity, 184
  bidding language, 189
  externalities, 178
  modularity, 182
  profiles, 187
  representation language, 187, 189
Vertex cover problem (MINVC), 443
Weighted independent set problem, 394, 397, 406
worst idleness, 163
worst-case complexity, 203