Computational and Constructive Design Theory

Mathematics and Its Applications

Managing Editor:

M. HAZEWINKEL

Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

Volume 368

Computational and Constructive Design Theory

edited by

W. D. Wallis
Southern Illinois University,
Carbondale, Illinois, U.S.A.



SPRINGER-SCIENCE+BUSINESS MEDIA, B.V.

A C.I.P. Catalogue record for this book is available from the Library of Congress

ISBN 978-1-4757-2499-8 ISBN 978-1-4757-2497-4 (eBook)
DOI 10.1007/978-1-4757-2497-4

Printed on acid-free paper

All Rights Reserved
© 1996 Springer Science+Business Media Dordrecht
Originally published by Kluwer Academic Publishers in 1996
Softcover reprint of the hardcover 1st edition 1996

No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission from the copyright owner.

CONTENTS

PREFACE ix

1 A BRIDGE CLUB DESIGN
W. D. Wallis 1
1 Introduction 1
2 The Problem 2
3 A Complete Search 3
4 Specifying the Group 4
5 Greedy Search 5
6 Steepest Descent 7
7 Hillclimbing and Simulated Annealing 7
8 Genetic Algorithms 9
9 The Solutions 10

2 THE FIRST TIME
Steven Furino 13
1 Introduction 13
2 A Plan 15
3 Near Resolvable Designs in Z_{tk+1} 17
4 Near Resolvable Designs in $Z_{tk} \cup \{\infty\}$ 24
5 Conclusion 28

3 COMPUTATIONAL METHODS IN DESIGN THEORY
Rudolf Mathon 29
1 Introduction 29
2 tk-matrices 31

3 Enumeration of 2-(6,3,2q) Designs 32
4 Tactical Decompositions 41
5 Randomized Search 43

4 FINDING DESIGNS WITH GENETIC ALGORITHMS
Daniel Ashlock 49
1 Genetic Algorithms 49
2 Direct Decomposition of K_n into Copies of K_m 51
3 Decomposing K_n into K_k with Difference Sets 56
4 Seating People on Successive Nights 59
5 Some Options in Designing a Genetic Algorithm 60
6 Conclusions and Suggestions 63

5 MAKING THE MOLS TABLE
Charles J. Colbourn and Jeffrey H. Dinitz 67
1 Basic Definitions 67
2 Filling, Truncation and Inflation 69
3 PBD, GDD and the Bose–Shrikhande–Parker Theorem 72
4 Steiner Systems, Symmetric Designs and Difference Sets 82
5 Wilson's Theorem 91
6 Direct Constructions 103
7 A Knowledge Base for *ITD* 107
8 Architecture of the System 109
9 Implementing the System 117
10 Experience with the System 124

6 ON WRITING ISOMORPHISM PROGRAMS
William Kocay 135
1 Introduction 135
2 Partition Refinement 137
3 Stabilising Vertices 145
4 The Schreier-Sims Algorithm 151
5 Graph Isomorphism 159
6 Afterword 173

7 THE NONEXISTENCE OF 4–(12,6,6) DESIGNS
Brendan D. McKay and Stanisław P. Radziszowski 177
1 Introduction 177
2 Three Towers of Designs 179
3 Algorithms 180
4 Constructive Results 182
5 Computations 184
6 The Next Challenge 187

8 BIBDS WITH $k = 6$ AND $\lambda = 1$
W. H. Mills 189
1 Introduction 189
2 Automorphism Groups and Base Blocks 191
3 Transversal Designs and Incomplete Transversal Designs 200
4 Pairwise Balanced Designs 204
5 Constructions 205
6 Proofs 222
7 Short lists 223

9 POLYHEDRAL METHODS IN DESIGN THEORY
Lucia Moura 227
1 Introduction 227
2 Polyhedral Theory and Algorithms 229
3 Integer Programming Formulations for Designs 236
4 Set Partitioning, Set Packing and Set Covering Polyhedra: Facets for Designs with $\lambda = 1$ 240
5 Designs: Polytopes and Algorithms 247

10 ANOTHER LOOK AT LARGE SETS OF STEINER TRIPLE SYSTEMS
Martin J. Sharry and Anne Penfold Street 255
1 Introduction 255
2 Direct Constructions from Quasigroups 258
3 Recursive Constructions 261
4 What Else Do We Need? 267

5 Proving the Theorem 270
6 Auxiliary Designs and Large Sets of Steiner Triple Systems 272

11 $(22, 33, 12, 8, 4)$-BIBD, AN UPDATE
G. H. J. van Rees 337
1 Introduction 337
2 Counting 338
3 Coding Theory 343
4 Group Methods 349
5 New Work 353
6 Acknowledgements 356

PREFACE

Over the last several years, there has been a significant increase in computational combinatorics. The most widely reported results were, of course, the proof of the Four Color Theorem and the proof that there is no projective plane of parameter 10.

Although the computer was essential in both proofs, the only reason for this was the fact that life is short. The computations involved were not different in kind from those which have been done by human brains without electronic assistance; they were just longer.

Another important fact to notice is that both problems were theoretical, pure-mathematical ones. The pursuit of the Four-Color Theorem has led to the development of whole branches of graph theory. The plane of parameter 10 is not an isolated case; its nonexistence is the first (and so far, the only) counterexample to the conjecture that the Bruck-Chowla-Ryser conditions were necessary and sufficient for the existence of a symmetric balanced incomplete block design; the study of this problem has also led to a number of theoretical advances, including investigation of the relationship between codes and designs.

Computer constructions have also been useful in the study of combinatorial spectrum problems: if a class of design has certain parameters, what is the set of values which the parameters can realize? One of the simplest cases, and one of the most difficult, is: what are the possible parameters of finite projective planes? The set of parameters is a set of positive integers; all the prime powers are members, but are there any others? We know of none, but infinitely many unsolved cases remain. In many cases, there is a recursive construction, so that the existence of a small number of "starter" designs leads to the construction of infinite classes of designs. Computers have proven very useful in finding "starter" designs.

There is very little mathematical literature on computers in design theory. This book is a step toward remedying this deficiency. I have aimed at two audiences,

the established researcher and the graduate student in combinatorics. It is hoped that both will gain new insights into these fascinating techniques.

My original intention was to write a long introduction which would outline the various computational techniques used in constructive design theory, with special reference to the ways in which various techniques normally associated with numerical analysis and continuous optimization, where all the variables are continuous, have been used in this essentially discrete field. However, I found that the topic was growing out of hand. So I tied my ideas to one particular class of designs where we had a great deal of success with hillclimbing methods, and where other numerical techniques have been applied and analyzed. The result appears as a tutorial paper. Steve Furino has supplied another tutorial paper, which was written to guide a graduate student just starting in computational design theory. These two papers are followed by a survey paper written by Rudi Mathon, a revised version of the material he presented at the British Combinatorial Conference in 1991.

The remaining papers are presented in alphabetical order, by the first author's surname. The grouping by topics in this introduction is rather arbitrary.

Dan Ashlock has written a fascinating introduction to genetic algorithms. At my request, he includes a genetic attack on the problem I discuss in my tutorial paper. Bill Kocay discusses isomorphism algorithms, with emphasis on graph isomorphism. (Not only do graphs arise as underlying structures in design theory, but also the methods of discussing graph isomorphism apply to design isomorphism in general.) Lucia Moura writes on a new area, polyhedral methods in design theory.

The construction of orthogonal Latin squares is a problem of ongoing combinatorial interest. Charlie Colbourn and Jeff Dinitz have recently constructed a table which lists the best known orders of sets of pairwise orthogonal Latin squares of sides up to 100,000, for inclusion in the Handbook of Combinatorial Designs. The construction of this table is essentially computational, essentially combinatorial, and of great interest to readers of this volume (but not the faint-hearted).

Brendan McKay and Stanisław Radziszowski were the first authors to respond to my call for papers. Their reward has been an inordinate delay in publication of an excellent paper in design theory, wherein they show that a set of parameters – in a natural sense the smallest undecided parameters for t-designs – cannot be realized.

I am very happy to present extensive discussions of two important open problems in design theory. Bill Mills surveys the existence of balanced incomplete block designs with block size six. Up until the 1970s there had been found no example of a balanced incomplete block design with $k = 6$, $\lambda = 1$ which did not have replication number r congruent to 0 or 1 (modulo 6). Many design theorists thought such a design was impossible. Mills startled us with a series of computer-aided constructions of the "impossible" designs. John van Rees writes about the (22, 8, 4)-balanced incomplete block design – the smallest parameters such that a 2-design is unknown but its existence is not excluded by the Bruck-Ryser-Chowla Theorem. This design was pursued diligently by the late Marshall Hall for a great part of his life.

Martin Sharry and Anne Street have supplied a description of the construction of large sets of triple systems. This important problem was solved only recently. The original proof by Lu was written in a series of seven papers; only six appeared, and the seventh was not submitted before his death. Although a copy of the final paper existed (and a translation by Shen Hao was circulated privately), the general mathematical public did not see a proof until Teirlinck presented his version; and both Lu and Tierlinck wrote in a concise style. This paper will make the construction clear to a wider audience.

Compiling this book has been a frustrating but rewarding experience. I would like to thank John Martindale, without whose encouragement this book would certainly not exist, and all the contributors.

CONTRIBUTORS

Daniel Ashlock
Department of Mathematics
Iowa State University
Ames, Iowa 50010
USA

Charles J. Colbourn
Department of Combinatorics
and Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

Jeffrey H. Dinitz
Department of Mathematics
and Statistics
University of Vermont
Burlington, Vermont 05405
USA

Steven Furino
Department of Combinatorics
and Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

William Kocay
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba R3T 2N2
CANADA

Rudolf Mathon
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
CANADA

Brendan D. McKay
Department of Computer Science
Australian National University
Canberra, ACT 0200
AUSTRALIA

W. H. Mills
Institute for Defense Analyses
Princeton, New Jersey 08540
USA

Lucia Moura
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
CANADA

Stanisław P. Radziszowski
Department of Computer Science
Rochester Institute of Technology
Rochester, New York 14623
USA

Martin J. Sharry
Centre for Combinatorics
Department of Mathematics
The University of Queensland
Queensland 4072
AUSTRALIA

Anne Penfold Street
Centre for Combinatorics
Department of Mathematics
The University of Queensland
Queensland 4072
AUSTRALIA

G. H. J. van Rees
Department of Computer Science
University of Manitoba
Winnipeg, Manitoba R3T 2N2
CANADA

W. D. Wallis
Department of Mathematics
Southern Illinois University
Carbondale, Illinois 62901-4408
USA

1

A BRIDGE CLUB DESIGN

W. D. Wallis

Department of Mathematics,
Southern Illinois University,
Carbondale, IL 62901
wdwallis@math.siu.edu

ABSTRACT

In this tutorial paper we consider a particular design problem which arose in scheduling a social bridge club consisting of twelve couples. At each meeting the club was divided into three groups of four couples, and each couple competed against each other couple in its group. The club met eight times, and it was required that each couple plays against each other couple two or three times. We have used this problem as a vehicle for exploring several techniques of design construction, some of which involve the application of methods more commonly used in continuous optimization problems.

1 INTRODUCTION

In many design construction problems, it is clear that the existence or otherwise of a suitable design could be determined by a complete search, but such a search would be infeasible. So various approximate methods have been proposed, many of them based on optimization methods. A solution is not guaranteed, even when one exists, but sometimes one will be found in a reasonable time. Elenbogen and Maxim [4] introduced such a design problem and analyzed some methods to tackle it. They thought their problem was infeasible, but in fact a complete search was possible provided the mathematical properties of the design were exploited.

W. D. Wallis (ed.), Computational and Constructive Design Theory, 1–12.
© 1996 *Kluwer Academic Publishers.*

In this paper we discuss their design problem. We show how to conduct a complete search, and describe some of the incomplete search methods outlined in [4] and some others which have been applied to the problem.

2 THE PROBLEM

In 1992, Elenbogen and Maxim [4] studied a design problem which arose in a social bridge club consisting of twelve couples. At each meeting the club is divided into three groups of four couples, and each couple competes against the other couple in its group. The club meets eight times, and it is required that each couple plays against every other couple two or three times.

The design they required was in fact a resolvable regular graph design. A regular graph design $RGD(v;k;r)$ is a collection of blocks of size k on a v-set such that every element occurs in r blocks and any pair of objects occur together in either λ_1 or λ_2 blocks, where

$$bk = vr,$$

and that $\lambda_1 = \lfloor\lambda\rfloor$, where

$$\lambda = r(k-1)/(v-1).$$

A design is resolvable if its blocks can be partitioned into r parallel classes such that each element occurs in exactly one block into class. So what Elenbogen and Maxim actually required is a resolvable $RGD(12,4:8)$, or $RRGD(12,4:8)$.

We shall use the term "bridge design" to mean a resolvable block design in which twelve elements meet in eight rounds, with three blocks of size four per round. We define the imbalance $C(D)$ of a design D as

$$C(D) = \sum_{x<y} \lambda_{xy} - \lfloor\lambda\rfloor$$

(as was done by Brown [2] and essentially in [4]). Then an $RRGD(12,4:8)$ is precisely a bridge design with imbalance 12, the theoretical minimum. Elenbogen and Maxim [4] did not succeed in producing such a design, but produced examples with imbalance 14. A design with imbalance 12 – an RRGD – was subsequently constructed in [10].

3 A COMPLETE SEARCH

There are 5775 possible parallel classes of twelve objects. One may as well relabel the first class as 0123 4567 89TE. There are then $(5775)^7/7!$ ways to choose eight classes. At first sight constructing and testing all these designs would take an unfeasible length of time. (The estimate in [4] is more than 10^13 years.) However, it is possible to restrict the range of possibilities which must be considered in an exhaustive search.

Suppose an $RRGD(12, 4 : 8)$ exists. Let us refer to two elements which occur together in three blocks as a special pair. Each element belongs to two special pairs, so the design contains twelve special pairs in total. So there are 36 occurrences of special pairs.

Consider a block a, b, c, d in the first parallel class of the design. Clearly it has intersection size at least 2 with at least one block in every subsequent parallel class. So there are at least 13 occurrences of two elements from a, b, c, d in a block of the design – one in block abcd, and at least one in each later parallel class. So a, b, c, d contains at least one special pair. So every block contains at least one special pair. We shall refer to a block with exactly one special pair as *good*. At the other extreme, if a block contained 5 or 6 special pairs, some element would belong to three of them, which is impossible. So every block contains 1, 2, 3 or 4 special pairs. Write a_i for the number of blocks containing i special pairs. We have

$$a_1 + a_2 + a_3 + a_4 = 24,$$

$$a_1 + 2a_2 + 3a_3 + 4a_4 = 36,$$

so $a_2 + 2a_3 + 3a_4 = 12$, whence $a_2 + a_3 + a_4 \leq 12$, and $a_1 \geq 12$. So the design must contain at least two parallel classes each of which contains at least two good blocks. Without loss of generality we may assume that the first two parallel classes each contain at least two good blocks.

Suppose the first parallel class is 0123 4567 89TE, where 0123 and 4567 are good blocks. In every subsequent parallel class there must be precisely one block with one pair from 0123 and precisely one block with one pair from 4567, or else there will be more than 13 occurrences of pairs from these blocks. So there are only two possible patterns:

$xxyy$ $xyzz$ $xyzz$ or $xxyz$ $xyyz$ $xyzz$

where x means "one of $0, 1, 2, 3$" , y means "one of $4, 5, 6, 7$" and z means "one of $8, 9, T, E$". We shall say that classes of these kinds meet the first class *in type I* and *in type II* respectively.

Class 2 also contains two good blocks. There are two cases: if classes 1 and 2 meet in type I, class 2 may be taken as 0145 2689 $36TE$; if they meet in type II, one may assume it is 0148 2569 $37TE$.

Based on these ideas, two searches were carried out by computer (see [10, 12]), one for each choice of class 2.

First sets S_1 and $S2$ were constructed, where S_i is the set of all possible parallel classes which meet class i either in type I or type II; S is defined as $S_1 \cup S_2$. S was partitioned into orbits under the action of the set of permutations which stabilize classes 1 and 2; the orbits were arbitrarily labeled T_0, T_1, ... and it was assumed:

if i is the smallest integer such that the RRGD contains a member of T_i, then the orbit leader of T_i is included.

So for each i the search found all possible RRGDs containing the first two parallel classes, the orbit leader of T_i, and five other classes chosen from $T_i \cup T_{i+1} \cup \cdots$.

No further isomorph rejection was carried out, although clearly the computation could have been further reduced by more theoretical work. But the exhaustive computation was now feasible, and 66 designs were found. After testing for isomorphism it is found that there are two isomorphism classes. They are listed below in Section 9.

4 SPECIFYING THE GROUP

If a complete search is infeasible, one may still attempt to construct a desired design by limiting the search. Of course, nonexistence cannot be proven in this way, and there is no guarantee of finding the design even when one exists.

One technique which has often been successful is to specify automorphisms of the design. Suppose , for example, there is a solution D to our problem which has automorphism group G. (G need not be the whole automorphism group.)

If R is one of the rounds of D, and $g \in G$, then Rg must also be a round of D. Kramer and Mesner [7] observed that the existence of a $t-$design with certain parameters and a certain group was equivalent to the existence of a (0,1)-solution to a certain matrix equation. Kreher and Radziszowski [9] used this observation and the technique of basis reduction to construct $t-$designs; Kreher has implemented it together with other algorithms in a suite of programs called the *Design Theory Toolchest.* An example of its use is [8].

A modification of these methods was applied to bridge club designs in [10]. In fact, the technique was applied to a series of designs, as we constructed an $RRGD(12,4;r)$ for $r = 9$ and $13 \leq r \leq 19$. (These values, together with a recursive construction and a known block design, enable us to construct $RRGD(12,4;r)$s for all $r > 8$.) First, all regular graph designs which had a certain automorphism were found, by applying the Kreher-Radziszowski technique to a modification of the equation (this is all incorporated into the Toolchest). The designs were tested for isomorphism, and exemplars of the isomorphism classes were tested for resolvability. It should be noted that the smaller the group, the larger the computation. Solutions were found relatively quickly. In all cases but 9 and 18, a solution was found with automorphism $(0123456789TE)$, although an automorphism of order 6 had to be used for $r = 9$ and a slightly more complicated ad hoc approach was used for $r = 18$ using some of the classes from the $r = 9$ case. But no solution for $r = 8$ was found. The reason for failure is clear when the two non-isomorphic solutions in Section 9 are examined: both have automorphism group of size 3. Several presentations of groups of size 4 were tested (unsuccessfully, of course), but the computation with group size 3 was too large.

5 GREEDY SEARCH

In an optimization algorithm, the aim is to minimize some objective function (cost function). Suppose a problem is solved by carrying out k steps in sequence, and the objective function is $c(S_1) + c(S_2) + \cdots + c(S_k)$, where $c(S_i)$ depends only on the implementation S_i chosen for step i. A *greedy algorithm* is one in which the only criterion for the choice of S_i is that $c(S_i)$ be minimized.

There is no guarantee that a greedy algorithm is optimal. Possibly the cheapest overall strategy involves making some early choices which are not locally optimal. However, there are many optimal greedy algorithms, including the simplex algorithm for linear programming and Kruskal's and Prim's algorithms

for minimal-cost spanning trees. A famous problem in which greedy algorithms do not fare well is the Traveling Salesman Problem. Greedy algorithms are discussed, for example, in [3].

A greedy algorithm for the bridge club problem is proposed in [4]. The cost of a round is defined to be the number of pairs which meet in round i and which had already met in an earlier round. The program kept a list of all possible rounds; it calculated the cost of each and chose one with the minimum cost. So the greedy algorithm minimized the total number of repeated pairings, up to and including the current round. The authors implemented this algorithm, which took about 3 hours of PC time, but the solution achieved had imbalance 22.

The objective function chosen by Elenbogen and Maxim does not seem to be appropriate. When a tournament is conducted, eighteen pairs meet in each round, so the total number of meetings is 132. If k of the 66 possible pairs do not ever meet then the number of repeat meetings will be $144-(66-k) = 78+k$, which attains its minimum value of 78 whenever every pair meets at least once. In particular, the schedule of imbalance 22 found in [4] satisfies this condition.

A more appropriate approach is to minimize the number of meetings between pairs which have already met twice. If this total is reduced to 12, an RRGD is achieved. Such an approach is doomed to failure however. Whatever ordering of the possible rounds is used, the first choice of a second round will be a repeat of round 1. But if even one block, let alone a complete round, is repeated then there will be elements which belong to three special pairs, as was pointed out in the preceding section.

A more refined search method is a multiple greedy algorithm in which two or more cost functions are used. Such methods are often indicated when the greedy algorithm gives many equally "good" possibilities, especially in the early stages of the solution. We shall describe one example. When the first $i-1$ rounds have been selected, all possible candidates S_i for round i are examined. The rounds which minimize $c(S_i)$, the total number of pairs which meet in S_i and have already met at least three times, are determined. Then among those we find the rounds for which $d(S_i)$, the number of pairs which meet in S_i and have already met at least twice,is smallest. Finally we choose a round S_i for which $e(S_i)$, the number of pairs which meet in S_i and have already met at least once is minimized. As a further refinement, after S_{i-1} is chosen, any round with a common block with are deleted from the list of possible rounds. Such an approach was attempted. It gives a better solution, with imbalance

18 and no pairs meeting four times, but was not optimal. Again, the solution obtained is in Section 9.

6 STEEPEST DESCENT

Several solution techniques involve the idea of two solutions being neighbors. A candidate solution (certainly not optimal, perhaps not very good) is chosen. Then another solution is selected from among the neighbors of the candidate. A neighbor of this new candidate is chosen next, and so on.

We shall define two bridge designs to be neighbors if they differ in only one round, and if moreover one can transform one design to the other by exchanging two elements, one each from two blocks in the round where they differ. Then each design has $384 = 8.3.4.4$ neighbors.

In the method of steepest descent, one starts by choosing a candidate design D_1 at random. After D_i has from been constructed, D_{i+1} is chosen from among the neighbors of D_i whose imbalance is smallest. The process stops when it is impossible to choose a D_{i+1} with smaller imbalance than D_i.

Elenbogen and Maxim [4] experimented with this approach. They added a modification – they defined the neighborhood of D to consist of those designs which differed from D by the exchange of two elements both of which meet some elements too often or too infrequently in D. This is an appealing idea (one is surely more likely to reduce the imbalance by tampering with two elements which contributed to it), and gave similar results to searching the whole neighborhood, but reduced the search time by a factor of over 20, requiring an average of about 15 evaluations at each stage rather than 384. They found that a typical run terminated after about 30 steps. They ran their program 25 times, and obtained one solution of imbalance 14, although the average was rather more (about 18). The solution of imbalance 14 is shown in Section 9.

7 HILLCLIMBING AND SIMULATED ANNEALING

Hillclimbing and simulated annealing are techniques which sample the neighborhood rather than examining all of it. If D_i is the candidate design at stage

i, then one examines one member of the neighborhood. In hillclimbing, if the cost of the new design is equal to or less than the cost of D_i, then the new design is chosen as D_{i+1}; otherwise $D_{i+1} = D_i$.

Simulated annealing (see [11]) is a little more liberal, in that occasional small fluctuations are allowed. (This models the cooling process when a liquid crystallizes; occasionally the temperature of the medium will rise temporarily, although there is a cooling trend.) In practise a small constant ("the annealing factor") is chosen; if the new design has a higher cost than the old one, a random number is selected and the new design becomes the candidate only when the result of dividing the absolute difference in cost by the annealing factor is less than the random number. Of course the selection of the annealing factor and the range of the random number is entirely empirical so this method may be difficult to apply at first; its applicability would appear to be greatest when one has to solve several related problems, or when it is possible to get some idea of the appropriate ranges by considering small examples.

Both hillclimbing and simulated annealing are attempts to get around the problem of local minima. To understand this, consider an inertialess smooth object on a smooth surface, acted on by gravity. The object will stop at any locally flat or upward-convex point. It may be that, if moved slightly, the object would recommence descending, and possibly reach the lowest point of the surface. In this analogy, hillclimbing is like pushing the object along a flat part of the surface to see if it starts moving again; simulated annealing adds the possibility of occasionally pushing it out of a small hollow.

A simulated annealing approach was tried in [4]. The authors made 25 trials of 2000 iterations each, with various initial schedules. Their best imbalance was 14, the same as in the steepest descent method, but they report that the method was significantly faster and overall gave better results than steepest descent.

Hillclimbing was attempted [10, 12] and was the first method to yield an optimal solution. The method was very fast. It was decided to terminate a run after 20000 iterations, and this took only seven seconds of cpu time on a SPARC ELC (in 1992). A solution was found after 50 runs.

8 GENETIC ALGORITHMS

Genetic algorithms were introduced by Holland [5], and are described in detail in another chapter of this volume [1]. They work by considering a set of candidates (the population) rather than one candidate; the population may sometimes contain duplicates. The aim is to maximize a function called *fitness*.

The notations and ideas are borrowed from genetics and evolution. The candidates are called *chromosomes*, and represented as sequences $(x_1, x_2, \cdots, x_k)$, where each variable (or *gene*) x_i has a small range of possible values. At each stage a new population (the *next generation*) is constructed from the previous one using three operations.

The first operation is *selection*, a random process in which some chromosomes are duplicated and others deleted; the probabilities of retention and duplication are larger when the fitness is larger. Of course the probability of duplication can be set at zero, in which case the population may steadily decrease.

The second operation is *crossover*, analogous to chromosome crossing in meiosis. The population is randomly partitioned into pairs; given the pair

$$(x_1, x_2, \cdots, x_k), (y_1, y_2, \cdots, y_k),$$

an index i is chosen at random, and with probability p the pair is replaced by

$$(x_1, x_2, \cdots, x_i, y_i + 1, y_i + 2, \cdots, y_k), (y_1, y_2, \cdots, y_i, x_i + 1, x_i + 2, \cdots, x_k).$$

The final operation is *mutation*. A probability q is specified, and any gene in any chromosome has probability q of being changed in value. If the genes have a range of more than two values, a probability distribution is applied in selecting the new value; often this is done so that the change will usually be small.

These basic ideas are modified significantly for specific problems. For example, one might change selection so that the best few chromosomes always survive to the next generation (*elitism*). Crossover might be allowed only at certain points. And so on. Moreover, the probabilities p and q, and the selection criterion, must be chosen; this is usually done by trial and error, using small examples.

Applying a genetic algorithm to the bridge design problem, the chromosomes would be designs. The genes might be the competitors (in which a design would

be a sequence of 96 variables, each with 12 possible values, and allowance would be made for the division into blocks and rounds), or possible blocks, or possible rounds. Crossover can only occur between rounds. The fitness of a chromosome is the negative of its imbalance.

Simpson [13] used such an algorithm with the players as genes. The first round was always 0123 4567 89TE, and crossover was allowed only between later rounds, with probability 0.1. Mutation was the transformation between neighbors described in Section 6, with probability 0.7 of a change in a round. The probability of selection of a chromosome (design) D was

$$\frac{b + C_{min} - C(D)}{n(b + C_{min} - C_{ave})}$$

where n, C_{min} and C_{ave} are the size of the previous generation and the minimum and average imbalances respectively of its members, and b is an arbitrary parameter.

After experimentation this algorithm was run 20 times with population size 150 and 250 generations. An optimal solution was found twice; imbalances varied from 12 to 20.

9 THE SOLUTIONS

This paper is about search methods, and the actual result is in a sense irrelevant. However the reader would surely be frustrated if the designs were not included! There are two isomorphism classes of solutions; representatives of the classes are given below. In each case the design has automorphism group of order 3, generated by (048)(159)(26T)(37E).

0123	4567	89TE	0123	4567	89TE
017T	245E	3689	025E	178T	3469
0359	148T	267E	0149	268E	357T
0258	1479	36TE	036T	1458	279E
0469	158E	237T	0589	136E	247T
0678	134E	259T	017E	2348	569T
04TE	1269	3578	0678	129T	345E
079E	156T	2348	04TE	1256	3789
First RRGD(12,4;8)			Second RRGD(12,4;8)		

0123	4567	89TE
016T	245E	3789
079E	1568	234T
059T	1478	236E
048E	1359	267T
0258	17TE	3459
068E	1249	357T
048T	135E	2679

Greedy solution from [4]

0123	4567	89TE
0148	2569	37TE
024T	1579	357T
0358	167T	249E
016E	258T	3479
069T	1345	278E
057E	1239	468T
0789	15TE	2346

Our greedy solution

02TE	1348	5679
034E	1578	269T
0479	136T	258E
0238	169E	457T
0678	124T	359E
015T	2379	468E
0356	127E	489T
048T	135E	2679

Steepest descent [4]

013T	279E	4568
0147	28TE	3569
0578	139E	246T
0235	146E	789T
0249	158E	367T
0689	1257	34TE
05TE	1269	3478
067E	1238	459T

Simulated annealing [4]

REFERENCES

[1] D Ashlock, Finding designs with genetic algorithms, this volume.

[2] R B Brown, Nonexistence of a regular graph design with $v = 17$ and $k = 8$, *Discrete Math* 68, 1988, 315–318.

[3] G Chartrand and O R Oellermann, *Applied and Algorithmic Graph Theory*. McGraw-Hill, 1993.

[4] B S Elenbogen and B R Maxim, Scheduling a bridge club (a case study in discrete optimization) , *Math Mag* 65 (1992), 18–26.

[5] J Holland, *Adaption in Natural and Artificial Systems*. University of Michigan Press, 1975.

[6] J A John and T J Mitchell, Optimal incomplete block designs, *J. Royal Statist. Soc.* 39B (1977), 39-43.

[7] E S Kramer and D M Mesner, t-designs on hypergraphs, *Discrete Math.* 15 (1976), 293-296.

[8] D L Kreher, Y M Chee, D de Caen, C J Colbourn and E S Kramer, Some new simple t-designs, *J. Combin. Math. Combin. Comput.* 7 (1990), 53–90.

[9] D L Kreher and S P Radziszowski, Finding t-designs by using basis reduction, *Congressus Num.* 55 (1986), 235–244.

[10] D L Kreher, G F Royle and W D Wallis, A family of resolvable regular graph designs, *Discrete Math.* (to appear).

[11] P J M van Laarhoven and E H L Aarts, *Simulated Annealing: Theory and Applications.* Reidel, 1987.

[12] G F Royle and W D Wallis, Constructing bridge club designs, *Bull. Inst. Combin. Appl* 11 (1994), 122–125.

[13] R J Simpson, Scheduling a bridge club using a genetic algorithm (submitted).

2

THE FIRST TIME

Steven Furino

Department of Combinatorics and Optimization
Faculty of Mathematics
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

ABSTRACT

Searching for a new combinatorial design can be an intimidating task. This paper is a tutorial in the art of discovering designs.

1 INTRODUCTION

This essay is directed primarily towards graduate students in design theory. It is a record of one design theorist's attempts to construct some new designs. I hope that an exposition of the process of searching for new designs may be of use to those who are engaged in the same task, perhaps for the first time. I encourage the reader to examine the process much more closely than the mathematics.

I assume that the reader knows the rudiments of design theory, in particular, that the reader knows what difference families are. I would recommend Anderson's *Design Theory* [1] as a good reference.

In my work on resolvable designs it has become clear that a particular class of designs, called near resolvable designs, are quite useful. A *near resolvable design* is a $BIBD(v, k, k-1)$ with the additional property that its blocks can be partitioned into classes that contain every point of the design but one and every point is missed by exactly one class. We denote such a design by $NRB(v, k)$ and give an example below.

W. D. Wallis (ed.), Computational and Constructive Design Theory, 13–28.
© 1996 *Kluwer Academic Publishers.*

1.1. Example. An $NRB(7,3)$.

points	{0}	{1}	{2}	{3}	{4}	{5}	{6}
block	{1,2,4}	{2,3,5}	{3,4,6}	{4,5,0}	{5,6,1}	{6,0,2}	{0,1,3}
set	{3,5,6}	{4,6,0}	{5,0,1}	{6,1,2}	{0,2,3}	{1,3,4}	{2,4,5}

Here is our first problem.

1.2. Problem. Does there exist an $NRB(22,7)$?

The basics of problem solving have not changed much through the ages. I think of them as follows.

1. Understand the problem.
2. Make a plan.
3. Execute the plan.
4. Check your solution.

We should not operate under the mistaken impression that important problems are solved in four distinct sequential steps. This is an iterative process which sometimes converges to a solution and sometimes, well, sometimes we are simply unable to solve the problem.

Let us make sure we understand what we need to find. At the risk of being excruciatingly explicit, we need to find:

(i) a $BIBD(22,7,6)$,
(ii) a partitioning of the blocks of this $BIBD$ into into 22 classes such that:
  (a) every class contains all but one point of the design and
  (b) each point is missing from exactly one class.

As part of understanding the problem, can we restate these conditions more simply? Condition (ii)(a), for example, is equivalent to requiring three mutually disjoint blocks in each class. Condition (ii)(b) is not actually necessary. The replication number for a $BIBD(22,7,6)$ is 21 and so there must be at least one class in which a given point, x, does not occur. But if blocks are mutually disjoint within a class, x must occur in 21 classes, and so x is absent from at most one class. Hence, x is missing from exactly one class. Thus, we really need to find:

(i) a $BIBD(22,7,6)$,

(ii) a partitioning of the blocks of this $BIBD$ into 22 classes each containing three mutually disjoint blocks.

Verifying that a set of blocks does satisfy these requirements is tedious but straightforward. Now we need a plan to find an $NRB(22,7)$.

2 A PLAN

First, we should check that the parameters of the design we seek satisfy the usual necessary conditions

$$\lambda(v-1) \equiv 0 \bmod (k-1)$$

$$\lambda v(v-1) \equiv 0 \bmod k(k-1)$$

The parameters (22, 7, 6) do indeed satisfy these conditions. Careful observation suggests that, in the case of near resolvable designs, the necessary conditions can be stated more simply. With $\lambda = k-1$ the first of these conditions will always be satisfied. With one point absent from each class it must be the case that $v \equiv 1 \bmod k$. But then the second condition is also satisfied. Thus we have proved the following lemma.

2.1. Lemma. *If an* $NRB(v,k)$ *exists, then* $v \equiv 1 \bmod k$.

Returning to the problem at hand, the first plan in design theory is: *Look it up.* Has someone already found the design we want? In my case I went to the tables of Mathon and Rosa [2] and happily discovered that there are three known $BIBD(22,7,6)$s. Checking the reference I unhappily discovered that none of these designs is near resolvable.

2.2. Exercise. Prove that the $BIBD(22,7,6)$s referred to in [2] are not near resolvable $BIBD$s.

The failure of our first plan illustrates a useful lesson. Failure is common. Most plans do not work. Persistence is essential.

Let us continue by analyzing the most straightforward search technique, exhaustive search. In this instance, we could generate all subsets of size 7 from a set of size 22, choose 66 blocks, test if they form a $BIBD(22,7,6)$, and, if so, try to arrange the blocks into appropriate classes. There are $\binom{22}{7} = 170544$

subsets of size 7 and

362967865068373365361692099815948733720920226526805217
422345530443153785412188557026961776881027990704827794
941884373414222748948749199540319254830313536709582808
601892611471546403874213449862948337550881776288178585
082202934973904022637707286812191264O

choices of 66 blocks. Investigating all of these possibilities would be prohibitively time consuming.

We need a new plan. How have other people looked for designs? Difference families have certainly been one of the most popular techniques. Unfortuntately, their purpose is to find $BIBD$s, not NRBs. Can we adapt the technique to suit our purposes? Possibly. Let's try to understand the technique well enough to alter it.

How many base blocks are required to generate a $BIBD(22,7,6)$ in Z_{22}? There are 21 non-zero elements each of which must appear 6 times in the list of differences generated by a difference family. Hence, 126 differences are required all together. Each block of size 7 produces $7 \times 6 = 42$ differences and so three blocks are needed. But this is exactly the number of disjoint blocks in a near resolution class of an $NRB(22,7)$. If we choose three mutually disjoint blocks as our difference family we will generate 22 classes of disjoint blocks, exactly what we want!

What plan can we use to find three disjoint base blocks? Exhaustive search might be a candidate in this smaller search space. Of the $\binom{22}{7} = 170544$ subsets of size 7 in Z_{22} we need to choose 3. Though $\binom{170544}{3} = 826704772153744$ is roughly 238 orders of magnitude smaller than naive exhaustive search, it is still too large. Here lies another lesson: A very small fraction of a very large search space can still be a very large search space.

Perhaps, though, a more intelligent exhaustive search would sufficiently reduce the search space. For example, if one point must miss the blocks of any class why don't we choose a point, say 0, that will not appear in any of the base blocks. And, since 1 must appear in one of the base blocks, we can agree that 1 will appear in the first block. Moreover, once we have chosen the points in one block, these points cannot occur in any other block. With this approach there are are most $\binom{20}{6} \times \binom{14}{7} \times \binom{7}{7} = 133024320$ cases to consider. Though this is large we are quickly approaching a reasonable number.

Before we continue, it is worth our while to decide whether we want to take a "general" approach to finding an $NRB(22,7)$ or a more "specific" approach. I am inclined to be as general as possible for as long as possible for three reasons:

1. We are able to test our theory on much smaller and workable examples.
2. If there is a problem with our approach, working in a broader context may allow us to determine, in advance, classes of situations in which the approach is either likely to succeed or doomed to fail.
3. Most importantly, if the approach works on this problem, it may work on other similar problems.

What kind of "general" approach do I have in mind? By Lemma 2.1, $v = tk+1$. Hence, we need to find t base blocks, all of which are mutually disjoint and none of which contain zero, that produce every nonzero element of Z_{tk+1} exactly $k-1$ times as a difference. Can we test out the reasonableness of this plan on a small but nontrivial example? If we look at the $NRB(7,3)$ in Example 1.1, the blocks $\{1,2,4\}$ and $\{3,5,6\}$ associated with the missing point 0 are exactly the kind of blocks we want. Now we must execute this plan.

3 NEAR RESOLVABLE DESIGNS IN Z_{tk+1}

We will attempt to find t base blocks which satisfy the following conditions:

(i) the base blocks are mutually disjoint,
(ii) no base block contains zero,
(iii) the list of differences generated by the base blocks contain every non-zero element of Z_{tk+1} exactly $k-1$ times.

Since examining all combinations of t blocks can be prohibitively time consuming, we will narrow the search by being much more selective about the blocks we generate. In particular, we attempt to determine the internal "structure" of each base block.

We assume that $tk+1$ is composite (we leave $tk+1$ a prime as an exercise). Let p be a factor of $tk+1$ and so

$$tk+1 = ps.$$

Since Z_{ps} is cyclic, it can be partitioned into cosets of a subgroup G of size s. Thus

$$G = \{0, p, 2p, \ldots, (s-1)p\}.$$

We denote the jth coset by $G + j$. That is,

$$G + j = \{j, p + j, 2p + j, \ldots, (s - 1)p + j\}.$$

Label the t base blocks as

$$B_0, B_1, B_2, \ldots, B_{t-1}.$$

We will attempt to determine the distribution of coset elements in these base blocks. To that end, let x_{ij} denote the number of elements in block B_i that belong to coset $G + j$.

Since every point in the subgroup G, other than the element 0, must appear in the base blocks, we have

$$\sum_{i=0}^{t-1} x_{i0} = s - 1 \tag{1}$$

Since every point of the coset $G + j, j = 1, 2, \ldots, p - 1$, must appear in the base blocks we have

$$\sum_{i=0}^{t-1} x_{ij} = s, j = 1, 2, 3, \ldots, p - 1 \tag{2}$$

Furthermore, every base block must contain exactly k elements. Hence

$$\sum_{j=0}^{p-1} x_{ij} = k, i = 1, 2, 3, \ldots, t - 1 \tag{3}$$

We can also count differences. Recall the following lemma from group theory.

3.1. Lemma. *If $x, y \in G + j$ then $x - y \in G$ and $y - x \in G$.*

Count the number of nonzero differences which lie in G and are generated by the base blocks.

$$\sum_{i=0}^{t-1} \sum_{j=0}^{p-1} x_{ij}(x_{ij} - 1) = (k - 1)(s - 1) \tag{4}$$

We also need the following lemma.

3.2. Lemma. *If $x \in G + d + i$ and $y \in G + i$ then $x - y \in G + d$.*

Now count the number of differences generated by the set of base blocks that must lie in $G + d$.

$$\sum_{i=0}^{t-1} \sum_{j=0}^{p-1} x_{ij}(x_{ij} - 1) = (k-1)(s-1) \tag{5}$$

Finally we require that

$$0 \leq x_{ij} \leq \min\{k, s\} \tag{6}$$

To see how all of these equations can help us, consider the set of linear equations (1)–(3). Diagrammatically they look like:

x_{i0}	x_{i1}	x_{i2}	$\cdots$	$x_{i,p-1}$		from
$11\cdots1$					$= s-1$	(1)
	$11\cdots1$				$= s$	(2)
		$11\cdots1$			$= s$	(2)
			.		.	.
			.		.	.
			.		.	.
				$11\cdots1$	$= s$	(2)
1	1	1	$\cdots$	1	$= k$	(3)
1	1	1	$\cdots$	1	$= k$	(3)
.	.	.	$\cdots$	.	.	.
.	.	.	$\cdots$	.	.	.
.	.	.	$\cdots$	.	.	.
1	1	1	$\cdots$	1	$= k$	(3)

Since the sum of the equations associated with systems (1) and (2) equals the sum of the equations associated with system (3), there is some redundancy. If we delete the first of the equations associated with system (3) we are left with $p + t - 1$ linearly independent equations in pt unknowns. Since (6) restricts the values x_{ij} can take, it is possible to actually enumerate all the solutions to the above system of linear equations. We only need to observe: from (6),

$$x_{ij} \in \{0, 1, 2, \ldots, \min\{k, s\}\}, i = 1, 2, \ldots, t-1, j = 1, 2, \ldots, t-1, \tag{7}$$

from (2),

$$x_{0j} = s - \sum_{i=1}^{t-1} x_{ij}, j = 1, 2, 3, \ldots, p-1, \tag{8}$$

from (3),

$$x_{i0} = k - \sum_{j=1}^{p-1} x_{ij}, i = 1, 2, 3, \ldots, t-1, \tag{9}$$

and from (1)

$$x_{00} = (s-1) - \sum_{i=1}^{t-1} x_{i0} \tag{10}$$

We can use this information to develop the following plan to search for an $NRB(tk+1, k)$.

3.3. Algorithm.

```
while (search space not exhausted) and (NRB not found) do
    use (7) to generate values for the independent variables
    use (8)-(10) to find values of the dependent variables
    test this solution against systems (4)-(6)
     if this solution is OK then
         attempt to choose t base blocks
     end if
end while
```

3.4. Exercise. Implement this algorithm.

Let us return to the problem of finding an $NRB(22,7)$. In this case $t = 3$, $p = 2$ and $s = 11$. We quickly learn that there are no solutions to the equations (1)–(6) for these parameters. After all this effort, we have failed.

Perhaps not entirely. Consider the problem of finding an $NRB(21,5)$ with $t = 4$, $p = 3$ and $s = 7$. In this case there are several solutions to the equations (1)–(6), one of which is given below.

3.5. Example.

		coset $G + j$		
	x_{ij}	0	1	2
block i	0	3	2	0
	1	2	1	2
	2	1	2	2
	3	0	2	3

How large is the search space associated with this solution to (1)–(6)? The computation is easy. For the first block there are $\binom{6}{3} \times \binom{7}{2} \times \binom{7}{0} = 420$ possibilities. Given the first block, there are $\binom{3}{2} \times \binom{5}{1} \times \binom{7}{2} = 315$ choices for the second block. Given the first two blocks there are $\binom{1}{31} \times \binom{4}{2} \times \binom{5}{2} = 60$ choices for the third block. The fourth block is completely determined by the preceding three blocks. All together this leaves 7938000 possibilities, accessible with a computer. In fact, an $NRB(21,5)$ was found by this algorithm.

3.6. Lemma. *An $NRB(21,5)$ exists.*

Proof. Developing the base blocks
{1, 3, 6, 12, 13}
{2, 4, 8, 9, 15}
{5, 7, 10, 14, 18}
{11, 16, 17, 19 20}
produces an $NRB(21,5)$.

3.7. Exercise. Verify that the blocks given in Lemma 3.6 will generate an $NRB(21,5)$.

This algorithm also produced previously unknown BIBDs. For example, we have the following lemma.

3.8. Lemma. *An $NRB(28,9)$ exists.*

Proof. Developing the base blocks
{1, 3, 4, 9, 11, 12, 15, 16, 21}
{5, 6, 7, 8, 13, 17, 19, 22, 26}
{2, 10, 14, 18, 20, 23, 24, 25, 27}
produces an $NRB(28,9)$.

3.9. Exercise. What near resolvable designs can your implementation of the algorithm generate?

Even when a solution to (1)–(6) exists, there are still problems. First, once a solution has been found, the number of blocks generated that correspond to this solution could be very large making the attempt to find the t base blocks very time consuming. For example, in searching for an $NRB(33,8)$ one of the solutions to the systems (1)–(6) would require constructing 15127843500000 blocks. Second, there may be too many solutions to (7)–(10) to completely enumerate.

In fact, systems (7)–(10) generate $(\min\{k,s\}+1)^{(t-1)(p-1)}$ solutions, almost all of which will not satisfy (4)–(6). Thirdly, even though solutions of (1)–(6) may be found that do not generate large numbers of blocks, it may be too time consuming to examine the search spaces associated with all of these solutions. Let's consider each of these concerns in more detail.

In the case of an $NRB(33,8)$, we return to the plan of reducing the number of blocks we wish to generate. One approach would be to observe that Z_{33} is isomorphic to $Z_{11} \times Z_3$. We could require that each block simultaneously satisfy solutions to (1)–(6) with $p = 3, s = 11$ and $p = 11, s = 3$. Intuitively, this should work since any near resolvable design constructed from base blocks will satisfy (1)–(6) for any p.

An example may clarify this. We have the following solution to (1)–(6) for $v = 21$, with $p = 7$ and $s = 3$.

3.10. Example.

x_{ij}	0	1	2	3	4	5	6
0	0	1	0	1	0	1	2
1	0	2	2	0	1	0	0
2	2	0	0	1	1	1	0
3	0	0	1	1	1	1	1

This particular solution would already make the search space small enough to examine by computer but let's see what happens when we require a difference family to satisfy both this set of constraints and the constraints of Example 3.5. Consider the following diagram.

3.11. Example. Candidates for the first block

x_{ij}	0	1	0	1	0	1	2
3	0	3	6	9	12	15	18
2	1	4	7	10	13	16	19
0	2	5	8	11	14	17	20

We must choose five points from the 21 elements in Z_{21}. The 21 group elements are arranged as the direct product of subgroups of size 3 and size 7. The first column contains the values x_{1j} of Example 3.5 and the first row contains the

values x_{1j} of Example 3.10. We can deduce a fair amount from this information. First, the first block contains no elements from rows or columns headed by a zero. This immediately eliminates 13 elements. The elements 18 and 19 must be in the first block since the column heading $x_{16} = 2$. Then any choice for the second element in row 2 forces the remaining choices. This implies that there are only three candidates for the first block!

3.12. Exercise. Construct the entire search space that satisfies a simultaneous solution to the values given in Examples 3.5 and 3.10.

As promising as this approach may be, it immediately runs into a problem. There may simply be too many solutions to (7)–(10) to completely enumerate. By tracking the sources of failure, it is evident that the step

> "use (7) to generate values for the independent variables"

forces a frequent violation of condition (6) for the dependent variables. It seems then, that we need an alternative to a complete enumeration. One possibility would be to generate only values of x_{ij} which partition k into parts all of size less than or equal to $\min\{k, s\}$. There is a discussion of this kind of integer partitioning in Reingold's Combinatorial Algorithms [4]. The trade off is that in generating only those values of x_{ij} that satisfy (6) we may be introducing violations of (1) and (2).

3.13. Exercise. Use integer partitioning to find solutions to (1)–(6).

It is not at all clear to me how to proceed when there are too many solutions to (1)–(6). So far it has not been a problem.

Returning to our starting point, what is the next plan to find an $NRB(22,7)$? There are several options. For example, we could use a different group or we could use a cyclic group of order 21 and one additional fixed point. Since cyclic groups are easiest to work with, let's try the second option.

4 NEAR RESOLVABLE DESIGNS IN $Z_{tk} \cup \{\infty\}$

In this case, we will assume the existence of one particular near parallel class associated with $\{\infty\}$ and then attempt to construct another near parallel class associated with $\{0\}$ that can be developed modulo tk.

Since Z_{tk} is cyclic, it can be partitioned into cosets of a subgroup H of size k. Thus

$$H = \{0, t, 2t, \ldots, (k-1)t\}.$$

We denote the jth coset of H by $H + j$. That is,

$$H + j = \{j, t + j, 2t + j, \ldots, (k-1)t + j\}.$$

Associate with the point $\{\infty\}$ the near parallel class

$$\{H, H + 1, H + 2, \ldots, H + (t-1)\}.$$

Now construct a near parallel class associated with 0 that can be developed modulo tk. We will attempt to find t base blocks which satisfy the following conditions:

(i) the base blocks are mutually disjoint,
(ii) no base block contains zero,
(iii) exactly one base block contains $\{\infty\}$,
(iv) the differences generated by the base blocks contain every element of Z_{tk} not in H exactly $k-1$ times and every nonzero element of H exactly $k-2$ times.

We will proceed exactly as we did in our previous plan.

Label the t base blocks we wish to find as

$$B_0, B_1, B_2, \ldots, B_{t-1}.$$

We may assume that $\infty \in B_0$. We will attempt to determine the distribution of coset elements in these base blocks. To that end, let x_{ij} denote the number of elements in block B_i that belong to coset $H + j$.

Since every point in the subgroup H, other than the element 0, must appear in the base blocks, we have

$$\sum_{i=0}^{t-1} x_{i0} = k - 1 \tag{11}$$

Since every point of the coset $H + j, j = 1, 2, 3, \ldots, t-1$ must appear in the base blocks we have

$$\sum_{i=0}^{t-1} x_{ij} = k, j = 1, 2, 3, \ldots, t-1 \tag{12}$$

Furthermore, the first base block, B_0, contains $k-1$ points of Z_{tk}. Hence

$$\sum_{j=0}^{t-1} x_{0j} = k - 1 \tag{13}$$

Also, every other base block must contain exactly k elements. Hence

$$\sum_{j=0}^{t-1} x_{ij} = k, i = 1, 2, 3, \ldots, t-1 \tag{14}$$

We can also count differences.Using Lemma 3.1, the number of nonzero differences in H generated by the set of base blocks is:

$$\sum_{i=0}^{t-1} \sum_{j=0}^{t-1} x_{ij}(x_{ij} - 1) = (k-2)(k-1) \tag{15}$$

Now count the number of differences generated by the base blocks that must lie in $H + d$.

$$\sum_{i=0}^{t-1} \sum_{j=0}^{t-1} x_{i,j+d} x_{ij} = (k-1)k \tag{16}$$

Finally, we require that

$$0 \leq x_{ij} \leq k \tag{17}$$

To see how all of these equations can help us, consider the set of linear equations (11)–(14). Diagrammatically they look like :

x_{i0}	x_{i1}	x_{i2}	$\cdots$	$x_{i,p-1}$		from
$11\cdots1$					$= k-1$	(11)
	$11\cdots1$				$= k$	(12)
		$11\cdots1$			$= k$	(12)
			.		.	.
			.		.	.
			.		.	.
				$11\cdots1$	$= k$	(12)
1	1	1	$\cdots$	1	$= k-1$	(13)
1	1	1	$\cdots$	1	$= k$	(14)
.	.	.	$\cdots$	.	.	.
.	.	.	$\cdots$	.	.	.
.	.	.	$\cdots$	.	.	.
1	1	1	$\cdots$	1	$= k$	(14)

Since the sum of the equations associated with systems (11) and (12) equals the sum of the equations associated with systems (13) and (14), there is some redundancy. If we delete the equation associated with (13) we are left with $2t-1$ linearly independent equations in t^2 unknowns. Since (17) restricts the values x_{ij} can take, we can again enumerate all the solutions to the above system of linear equations. We only need to observe: from (17)

$$x_{ij} \in \{0, 1, 2, \ldots, ki = 1, 2, \ldots, t-1, j = 1, 2, \ldots, t-1\}, \tag{18}$$

from (12),

$$x_{0j} = k - \sum_{i=1}^{t-1} x_{ij}, j = 1, 2, \ldots, t-1, \tag{19}$$

from (13),

$$x_{i0} = k - \sum_{j=1}^{t-1} x_{ij}, i = 1, 2, \ldots, t-1, \tag{20}$$

and from (11)

$$x_{00} = (k-1) - \sum_{i=1}^{t-1} x_{i0} \tag{21}$$

We can use this information to develop the following plan to search for an $NRB(tk+1, k)$.

4.1. Algorithm.

```
while (search space not exhausted) and (NRB not found) do
    use (18) to generate values for the independent variables
    use (19)-(21) to find values of the dependent variables
    test this solution against systems (14)-(17)
     if this solution is OK then
         generate all blocks corresponding to this solution
         attempt to choose t base blocks
     end if
end while
```

4.2. Exercise. Implement this algorithm.

This time we succeed!

4.3. Lemma. An NRB(22,7) exists.

Proof. Developing the base blocks
$\{\infty, 1, 5, 6, 7, 15, 18\}$
$\{9, 11, 12, 14, 16, 19, 20\}$
$\{2, 3, 4, 8, 10, 13, 17\}$
and appending the class
$\{0, 3, 6, 9, 12, 15, 18\}$
$\{1, 4, 7, 10, 13, 16, 19\}$
$\{2, 5, 8, 11, 14, 17, 20\}$
produces an NRB(22,7).

The next part of the plan is to *Check the solution.* Certainly verifying that the base blocks above satisfy the stated requirements would be a minimal expectation, but there are more opportunities here than simple confirmation. Thinking back to the original reason for investigating near resolvable designs, that is, finding tools for the construction of resolvable designs, we might ask ourselves if these techniques could be applied directly to the search for resolvable designs. They most certainly can.

4.4. Exercise. Adapt the approach outlined in this section to search for resolvable designs with index $k - 1$. An example of such a design is given below.

4.5. Example. An $RB(24,6,5)$ exists.

Proof. Developing the base blocks
$\{\infty, 0, 1, 7, 15, 20\}$
$\{2, 3, 4, 5, 10, 14\}$
$\{6, 11, 13, 17, 19, 22\}$
$\{8, 9, 12, 16, 18, 21\}$
produces an $RB(24,6,5)$.

5 CONCLUSION

The preceding discussion may have left the reader unsatisfied, bothered by that uneasy feeling associated with the question *Just how do I come up with a good idea*? Here I must draw the reader's attention to Polya's excellent book *How To Solve It* [3]. It is certainly worth reading. I do not wish to imply that one will immediately be able to produce good ideas after reading Polya's book but rather one may ask better questions that will lead to good ideas.

REFERENCES

[1] I. Anderson, *Combinatorial Designs: Construction Methods* (Ellis Horwood, Chichester, England, 1990).

[2] R. Mathon and A. Rosa, "Tables of Parameters of BIBDs with $r \leq 41$ including existence, enumeration and resolvability results: an update," *Ars Combinatoria* 30 (1990), 65–96.

[3] G. Polya, *How to Solve It* (Princeton University Press, Princeton N. J., 1945).

[4] E. M. Reingold, J. Nievergelt, N. Deo, *Combinatorial Algorithms* (Prentice Hall, Englewood Cliffs, N. J., 1977).

3

COMPUTATIONAL METHODS IN DESIGN THEORY

Rudolf Mathon

Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S1A4

1 INTRODUCTION

Enumeration theory, which aims to count the number of distinct (non-equivalent) elements in a given class of combinatorial objects, constitutes a significant area in combinatorial analysis. The object of constructive enumeration consists of creating a complete list of configurations with given properties [5], [8]. There are several reasons which stimulate research in constructive enumeration. Classical methods are not applicable to many interesting classes of objects such as strongly regular graphs, combinatorial designs, error correcting codes, etc. At present, the only available way to count them is by using algorithmic techniques for fixed values of parameters. Lists of objects are important for generating and testing various hypotheses about invariants, characterization, etc. Moreover, examples of designs with given properties are needed in many areas of applied combinatorics such as coding and experiment planning theories, network reliability and cryptography. Algorithms for constructive enumeration frequently require searching in high dimensional spaces and employ sophisticated techniques to identify partial (final) solutions. Such methods may be of independent interest in artificial intelligence, computer vision, neural networks and combinatorial optimization.

There are several common algorithmic approaches which are used to search for combinatorial configurations with particular properties. These can be divided into two broad classes depending on whether or not they search for all possible solutions in a systematic manner. Among the exhaustive techniques

Research of the author is supported by the Natural Sciences and Engineering Research Council of Canada under grant number A8651.
An earlier version of this paper appeared as [18].

W. D. Wallis (ed.), Computational and Constructive Design Theory, 29–48.
© 1996 *Kluwer Academic Publishers.*

backtracking plays a prominent role. A backtrack algorithm attempts to find a solution vector by recursively building up partial solutions one element at a time. The vectors are examined in lexicographical order until all possible candidates for a component have been exhausted after which one backtracks. The search tree is pruned by rejecting partial solutions which are not admissible or have been generated before (so called isomorph rejection). One of the first systematic searches was carried out in 1919 by Cole, Cummings and White [4] to classify Steiner triple systems of order 15. Since then many researchers have used backtrack search to enumerate designs with small parameters [3], [6], [8]. However, simple exhaustive techniques quickly run out of steam for larger size problems due to exponential growth in search complexity. One way of delaying the combinatorial explosion is to look only for some restricted classes of "nice" configurations such as cyclic or transitive designs. By assuming certain automorphisms it has been possible to either construct or prove nonexistence of classes of graphs, designs and projective planes. The use of group actions has also found its way into algorithmic techniques for generating designs of larger size. Most current construction methods employ a tk-matrix or an incidence matrix as a design representation. A group action induces a fusion of a tk-matrix or a tactical decomposition of an incidence matrix which results in a significant reduction of computational effort. Methods based on tk-matrices and tactical decompositions will be discussed in Sections 2 and 4. In Section 3 we will use tk-matrices to enumerate completely 2-$(6,3,2q)$ designs for arbitrary values of q.

The other class of methods is nonexhaustive in nature and frequently employs some probabilistic techniques. The search is usually represented in the form of an optimization problem in which the "goodness" of an approximate solution is measured in terms of an objective or cost function. Several randomized search procedures have been applied in the construction of combinatorial objects. These include hill-climbing, local search and simulated annealing [2], [9], [10], [21]. In hill-climbing we keep randomly extending a partial solution until either a final solution is found or a set of partial solutions is reached which can not be extended, in which case we repeat the search from another initial guess. In local search one defines a set of configurations, a simple transformation and a cost function on these configurations. The goal is to find a configuration of minimum cost by applying randomized transformations. A new configuration is accepted if its cost does not exceed the cost of its pre-image. Simulated annealing is a local search in which poor local minima are avoided by allowing an occasional uphill move. This is accomplished in a randomized fashion by using "temperature" as a control parameter [9], [10]. Randomized search techniques will be discussed in Section 5.

2 tk-MATRICES

A t-(v,k,λ) design, $0 < t < k < v$, is a pair (X,B) where B is a collection of k-subsets (called blocks) from a v-set X such that each t-subset of X is contained in exactly λ blocks of B. A design is called *simple* if it has no repeated blocks and *trivial* if it is a multiple of the set of all k-subsets of X. For $0 < s < v$ denote by X_s the set of all s-subsets of X, $\mid X_s \mid = \binom{v}{s}$. Let $A_{tk} = (a_{ij})$ be a $\mid X_t \mid$ by $\mid X_k \mid$ matrix with rows and columns indexed by t- and k-subsets of X, respectively, $a_{ij} = 1$ if the i-th t-set is contained in the j-th k-set and $a_{ij} = 0$ otherwise. Hence, A_{tk} is a t- versus k- subset incidence matrix, where incidence is set inclusion. It is easy to see that the existence of a t-(v,k,λ) design is equivalent to the existence of a solution $\mathbf{x}$ to the matrix equation $A_{tk}\mathbf{x} = \lambda\mathbf{j}$, where $\mathbf{x}$ is a $\mid X_k \mid$-vector with non-negative integer entries and $\mathbf{j}$ is the all one $\mid X_t \mid$-vector. This formulation converts the problem of finding a design to an integer programming problem. Unfortunately, for most designs of interest the tk-matrix A_{tk} is prohibitively large. So, for example, the smallest parameter set 2-(22,8,4) for which a design is not known to exist gives rise to a 231 by 319770 tk-matrix. The size of A_{tk} can be reduced by assuming a group action on the set X. A group G acting on X induces an action on both X_t and X_k. Let ρ_t and ρ_k be the number of orbits of G on X_t and X_k, respectively. Denote by $A_{tk}(G)$ a ρ_t by ρ_k matrix with integer entries a_{ij} counting the number of k-sets in the j-th orbit of X_k containing a representative t-set in the i-th orbit of X_t. Hence $A_{tk}(G)$ is obtained from A_{tk} by adding the columns in each k-orbit and keeping one representative row from each t-orbit. The new integer system $A_{tk}(G)\mathbf{x} = \lambda\mathbf{j}$ requires finding a non-negative integer ρ_k-vector $\mathbf{x}$, where $\mathbf{j}$ is a ρ_t-vector of all ones. If a solution is found then the corresponding design has G as a subgroup of its automorphism group. Tk-matrices under group action were introduced by Kramer and Mesner [10] and led to a series of significant discoveries in the theory of t-designs. Magliveras and Leavitt [16] used projective linear groups to find the first examples of simple 6-designs. Later, Kramer, Leavitt and Magliveras [12] generated many simple 6-designs on 19 and 20 points based on the groups PSL_2 and PGL_2. To solve the linear system $A_{tk}(G)\mathbf{x} = \lambda\mathbf{j}$ they employed an ingenious exhaustive algorithm due to Leavitt in which new equations are added to the system to restrict the search space. Leavitt's algorithm can handle general tk-matrices with typically 30-50 rows and up to a few hundred columns.

An integer programing approach for computing (0,1)-vectors $\mathbf{x}$ satisfying

$$A_{tk}(G)\mathbf{x} = \lambda\mathbf{j}$$

was proposed by Kreher and Radziszowski [15]. They observed that if $\mathbf{x}$ is an integer solution then $[\mathbf{x},\mathbf{0}]^T$ is a vector in the integer lattice L spanned by the

columns (integer linear combinations) of the matrix

$$\begin{bmatrix} I & \mathbf{0} \\ A_{tk}(G) & -\mathbf{j} \end{bmatrix}$$

They also noticed that integer solutions $\mathbf{x}$ satisfying $A_{tk}(G)\mathbf{x} = \lambda\mathbf{j}$ are often short vectors in L and used a modified basis reduction algorithm for finding short vectors in some reduced lattice. These heuristics proved successful in the construction of many nice t-designs with specified automorphism groups, most notably a simple 6-(14,7,4) design [15].

3 ENUMERATION OF 2-(6,3,2q) DESIGNS

In order to illustrate the tk-matrix approach we will completely enumerate the one parameter family of 2-(6,3,2q) designs . A 2- (6,3,2q) design (X, B) has $10q$ blocks of size 3, each element is contained in half of them and each unordered pair of elements appears in $2q$ blocks. The tk-matrix A_{23} has 15 rows and 20 columns indexed by the unordered pairs and triples from the element set $X = \{1, 2, 3, 4, 5, 6\}$ (see Table 1).

Then a 2-(6,3,2q) design exists if and only if $A_{23}\mathbf{x} = 2q\mathbf{j}$ for an integer 20-vector of block multiplicities $\mathbf{x} \geq 0$. Since $A_{23}{A_{23}}^T = 4I + B$, where B is a symmetric adjacency matrix of a strongly regular graph (15,8,4,4) with eigenvalues 8,2,-2 of multiplicity 1,5,9 respectively, we see that $\text{rank}(A_{23}{A_{23}}^T) = 15$. Hence we can write $\mathbf{x} = A_{23}^T\mathbf{y} + Z^T\mathbf{z}$, where ${A_{23}}^T = 0$ and $\text{rank}(ZZ^T) = 5$, $\mathbf{y}$ and $\mathbf{z}$ are real vectors of length 15 and 5 respectively. Since $A_{23}\mathbf{x} = A_{23}{A_{23}}^T\mathbf{y} = 2q\mathbf{j}$ is constant, we see that $\mathbf{y} = \frac{q}{6}\mathbf{j}$ and so $\mathbf{x} = Z^T\mathbf{z} + \frac{q}{2}\mathbf{j}$ for any real 5-vector $\mathbf{z}$. The rows $10+i$ and $11-i$ of Z add up to 0 which implies that $x_{10+i} = q - x_{11-i}$, $1 \leq i \leq 10$, and so $\mathbf{0} \leq \mathbf{x} \leq q\mathbf{j}$.

It is possible to express $\mathbf{z}$ in terms of a special subset of the components of $\mathbf{x}$ and derive a special relation between them:

$$\begin{bmatrix} z_1 \\ z_3 \\ z_4 \\ z_2 \\ z_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_9 \\ x_{10} \\ x_6 \end{bmatrix} - \frac{q}{2} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

A_{23}	1 2 3	1 2 4	1 2 5	1 2 6	1 3 4	1 3 5	1 3 6	1 4 5	1 4 6	1 5 6	2 3 4	2 3 5	2 3 6	2 4 5	2 4 6	2 5 6	3 4 5	3 4 6	3 5 6	4 5 6
12	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
16	0	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
24	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0
25	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0
26	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0
34	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0
35	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	1	0
36	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1	0
45	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
46	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1
56	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1

Table 1 A_{23} matrix for the set $X = \{1, 2, 3, 4, 5, 6\}$

$$\begin{bmatrix} x_8 \\ x_7 \\ x_3 \\ x_5 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & -1 \\ -1 & 0 & 1 & 0 & -1 \\ -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_9 \\ x_{10} \\ x_6 \end{bmatrix} + \begin{bmatrix} q \\ q \\ q \\ q \\ q \end{bmatrix}$$

where $x_{10+i} = q - x_{11-i}, 1 \leq i \leq 10$. This can be used to find all distinct designs for a particular value of q. We count the number of distinct selections of $x_1, x_2, x_6, x_9, x_{10}$ between 0 and q such that x_3, x_4, x_5, x_7, x_8 have values also between 0 and q. If $x_1 = q - k$, $x_6 = q - i$ and $x_9 = q - j$ then the admissible values of x_2 and x_{10} are given in Figure 1. The sides which are not parallel to the coordinate axes have slope -1. The number of distinct designs is the sum of integer points in the hatched areas over admissible values of i, j and k. Let $\delta = q - i + j$ and denote by $V(k, j, \delta, q)$ the hatched area in the left-hand

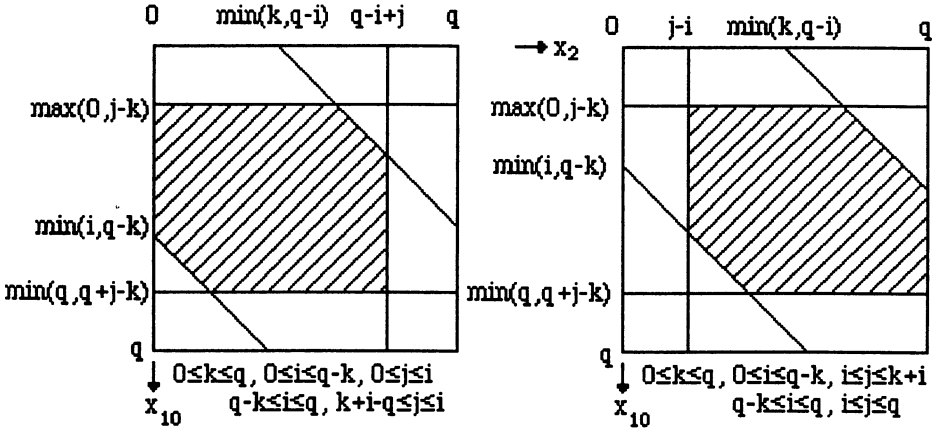


Figure 1 Admissible values of x_2 and x_{10}.

diagram for $0 \le k, j \le \delta, 0 \le \delta \le q$. If

$$W(\delta, q) = \sum_{k=0}^{\delta} \sum_{j=0}^{\delta} V(k, j, \delta, q),$$

then

$$\begin{aligned} W(2s+1,q) &= 4\sum_{k=0}^{s}\sum_{j=0}^{k}[(2s+2)(q-k+j+1) - 2\binom{2s+2-k}{2}] \\ &\quad +4\sum_{k=0}^{s-1}\sum_{j=k+1}^{s}[(2s+2)(q+k-j+1) - 2\binom{2s+2-j}{2}] \\ &= 24(2q+7)\binom{s+2}{3} - 240\binom{s+3}{4} - 16\binom{s+1}{2} + 8q(s+1), \\ W(2s,q) &= 4\sum_{k=0}^{s-1}\sum_{j=0}^{k}[(2s+1)(q-k+j+1) - 2\binom{2s+1-k}{2}] \\ &= 4\sum_{k=0}^{s-2}\sum_{j=k+1}^{s-1}[(2s+1)(q+k-j+1) - 2\binom{2s+1-j}{2}] \end{aligned}$$

$$
\begin{aligned}
&+4\sum_{k=0}^{s-1}[(2s+1)(q+k-s+1)-2\binom{2s+1}{2}]\\
&+(2s+1)(q+1)-2\binom{s+1}{2}]\\
=\ & 48(q+1)\binom{s+2}{3}-240\binom{s+2}{4}-(24q+70)\binom{s+1}{2}\\
&+(q+1)(2s+1).
\end{aligned}
$$

Noting that the second diagram can be obtained from the first by adding $j-i$ to x_2, we obtain a formula for the number $D(q)$ of distinct 2-(6,3,2q) designs:

$$
\begin{aligned}
D(2t+1) &= 2\sum_{s=0}^{t} W(2s+1,2t+1)+W(2s,2t+1)-W(2t+1,2t+1)\\
&= 30\binom{2t}{5}+120\binom{2t}{4}+190\binom{2t}{3}+150\binom{2t}{2}+122t+12,\\
D(2t) &= 2\sum_{s=0}^{t} W(2s+1,2t)+W(2s,2t)-W(2t,2t)\\
&= 30\binom{2t-1}{5}+120\binom{2t-1}{4}+190\binom{2t-1}{3}\\
&\quad +150\binom{2t-1}{2}+61(2t-1)+12.
\end{aligned}
$$

The last two expressions can be combined into a final formula

$$D(q)=(q+1)(3q^4+12q^3+23q^2+22q+12)/12.$$

In order to count nonisomorphic designs it is necessary to investigate prime order automorphisms. The primes dividing $|S_6|$ are 5, 3 and 2.

(α). *Automorphisms of order 5.*

The permutation $\sigma=(23456)$ generates 4 block-orbits and 3 orbits of pairs. Solving $A_{23}(\sigma)\mathbf{x}=2q\mathbf{j}$ yields $x_2=x_3=q-x_1, x_4=x_1$ in terms of x_1. If $x_2=x_3$ then $|\,G\,|=60$. If $q=2t+1$ then the number of distinct designs $D_{60}(2t+1)=2t+2$. If $q=2t$ then the case $x_1=x_2=x_3=x_4=t$ must be excluded (a t-multiple of a complete design) and so $D_{60}(2t)=2t$. Dividing by the index $r_{60}=2$ of the stabilizer in the block-orbit group counts the nonisomorphic designs $N_{60}(2t+1)=N_{60}(2t+2)=t+1$. Finally, the t-multiple of the complete design is unique, $N_{720}(2t+1)=0, N_{720}(2t+2)=1$.

(β). *Automorphisms of order 3.*

Both permutations $\sigma_1 = (123)(456)$ and $\sigma_2 = (123)$ fix 2 blocks and partition the remaining blocks into 6 orbits of size 3. For σ_1 we obtain the solution $x_1 = 2q - x_2 - x_3 - x_4$, and $x_{i+4} = q - x_i, 1 \leq i \leq 4$, where all x_i's are between 0 and q inclusive. The total number of solutions can easily be counted with the aid of a diagram in which the admissible values for x_3 and x_4 are plotted for a given value of $x_2 = i$. Summation over all $i, 0 < i < q$ yields $T(q) = (q+1)(2q^2 + 4q + 3)/3$. Designs with $|\ G\ | = 3s, s > 1$ correspond to either $x_k = x_l$, or to $x_k = q - x_l$, for some $k < l$. Summation over all values $i, 0 \leq i \leq q$ and $k, l \in 2, 3, 4, k < l$, and using inclusion-exclusion, yields $P(2t+1) = 18t^2+24t+6-2Q(2t+1), P(2t+2) = 18t^2+36t+21-2Q(2t+2)$, where $Q(s) = |\ \{x \mid s \leq 3x \leq 2s\}\ |$ is the number of solutions with $x_2 = x_3 = x_4$. The number of distinct designs with $|\ G\ | = 3$ is $D_3(q) = T(q) - P(q)$. Dividing by the index r_3=12 yields the number of nonisomorphic designs

$$\begin{aligned} N_3(2t+1) &= (t(t+1)(8t+11) + 3Q(2t+1))/18, \\ N_3(2t+2) &= (8t^3 + 9t^2 + t - 3 + 3Q(2t+2))/18. \end{aligned}$$

If $x_2 = x_3 = x_4 \neq q/2$ then $|\ G\ | = 36$ and $r_{36} = 2$. Since $D_{36}(2t+1) = Q(2t+1), D_{36}(2t+2) = Q(2t+2) - 1$, where $Q(3t+i) = t+i-1$, we see that $N_{36} = D_{36}/2$. If $x_k = x_l \neq q/2$ for some $k < l$ and $x_m \neq x_k, x_m \neq q - x_k$ then $|\ G\ | = 6$ and $r_3 = 6$. The number of distinct designs is easily counted. We have $D_6(2t+1) = 6t(t+1) - 3Q(2t+1), D_6(2t+2) = 6t^2 + 9t + 6 - 3Q(2t+2)$ and so $N_6(2t+i) = t(t+1) - N_{36}(2t+i), i = 1, 2$. If $x_k = q - x_l, x_k \neq x_l$ and $x_m \neq x_k, x_m \neq x_l, x_m \neq q/2$ for some distinct $k, l, m \in \{2, 3, 4\}$ then $|\ G\ | = 12$ and $r_{12} = 12$. Counting distinct designs yields $D_{12}(2t+i) = 12t(t+1)$ and so $N_{12}(2t+i) = t(t+1), i = 1, 2$. If $q = 2t$ is even and either $x_k = x_l = t$ and $x_m \neq t$, or $x_k = q - x_l, x_k \neq x_l$ and $x_m = t$ for some distinct $k, l, m \in 2, 3, 4$ then $|\ G\ | = 24$ and $r_{24} = 6$. Then $D_{24}(2t+2) = 12(t+1)$ and so $N_{24}(2t+1) = 0, N_{24}(2t+2) = t+1$. Finally, if $x_k = x_l = q - x_m \neq q/2$ then $|\ G\ | = 60$ and if $x_2 = x_3 = x_4 = t$ and q is even then $|\ G\ | = 720$. These cases have been considered before. For σ_2 we obtain the solutions $x_1 = 3x_8 - q, x_5 = q - x_1, x_2 = x_3 = x_4 = q - x_8, x_6 = x_7 = x_8$ which are contained in those for σ_1.

(γ). *Automorphisms of order 2.*

There are 3 types of permutations, $\sigma_1 = (12)(34)(56), \sigma_2 = (12)(34)$ and $\sigma_3 = (12)$, fixing 0, 2 and 4 elements, respectively. The solutions of the tk-

matrix equations for σ_1 are contained in those for σ_2. We will treat each of the remaining cases separately.

Case 1 (2 fixed points). There are 4 fixed blocks and 8 orbits of size 2. Solving the tk-matrix equations yields $x_1 = x_9 + x_{11} + x_{12} - q, x_2 = q + x_9 - x_{11} - x_{12}, x_3 = x_{10} = q - x_9, x_4 = x_9, x_5 = q - x_{11}, x_6 = q - x_{12}, x_7 = q - x_1$ and $x_8 = q - x_2$, where $0 \leq x_i \leq q$ for all i. Using a diagram to sum the numbers of admissible values of x_{11} and x_{12} for a fixed $x_9 = i$ and then summing over all $i, 0 \leq i \leq q$ yields $T(2t+1) = 2(t+1)(5t^2 + 10t + 6)/3, T(2t+2) = (10t^3 + 45t^2 + 71t - 39)/3$. Designs with $|G| = 2s, s > 1$ satisfy at least one of the following conditions: $x_9 = x_{11}, x_9 = x_{12}, x_9 = q - x_{11}, x_9 = q - x_{12}, x_{11} = x_{12}, x_{11} = q - x_{12}, x_9 = q/2$. Using inclusion-exclusion we can sum over this subset of values to obtain $P(4t+1) = 52t^2 + 36t + 4 - 8Q(1), P(4t+2) = 64t^2 + 54t + 13 - 8Q(2), P(4t+3) = 52t^2 + 88t + 36 - 8Q(3), P(4t+4) = 64t^2 + 118t + 55 - 8Q(4)$, where $Q(i) = N_{36}(4t+i)$. The number of distinct designs with $|G| = 2$ is $D_2^1(q) = T(Q) - P(Q)$. Since the index is 8, we have $N_2^1(q) = D_2^1(q)/8$. If $x_{11} = x_{12}$ and $x_9 \neq q/2$, or if $x_{11} \neq x_{12}, x_9 = q/2$ and $x_{11}, x_{12} \neq q/2, x_{11} + x_{12} \neq q, x_{11} + 2x_{12} \neq 3q/2, 2x_{11} + x_{12} \neq 3q/2$ then $|G| = 4$ and the index is 4 in the first case and 12 in the second case. Using inclusion-exclusion we can sum over this subset of values to obtain the number of distinct designs $D_4(4t+1) = 4t(t+1) - 4Q(1), D_4(4t+3) = 4(t+1)^2 - 4Q(3), D_4(4t+2) = 4t^2 - 4Q(2) + [12t^2], D_4(4t+4) = 4t(t+1) - 4Q(3) - [12t(t+1)]$, where $Q(i) = N_{36}(4t+i)$. The number of nonisomorphic designs can be obtained from D_4 by dividing the terms in the square brackets by 12 and all other terms by 4. Finally, if $q = 2t, x_9 = t, x_{11} \neq t$ and either $x_{11} = x_{12}$, or $x_{11} + 2x_{12} = 3t$, or $2x_{11} + x_{12} = 3t$, then $|G| = 8$ and the index is 8. An easy count yields $D_8(4t) = D_8(4t+2) = 6t$ and so $N_8(4t+2) = t, N_8(4t+4) = t+1$. All other values of x_9, x_{11} and x_{12} have been treated before.

Case 2 (4 fixed points). There are 8 fixed blocks and 6 orbits of size 2. Solving the tk-matrix equations yields $x_1 = x_{12} + x_{13} + x_{14} - q, x_2 = q + x_{12} - x_{13} - x_{14}, x_3 = q - x_{12} + x_{13} - x_{14}, x_4 = q - x_{12} - x_{13} + x_{14}, x_{4+i} = q - x_{11+i}, 1 \leq i \leq 3, x_{7+j} = q - x_j, 1 \leq j \leq 4$, where $0 \leq x_i \leq q$ for all i. Using a diagram to sum the numbers of admissible values of x_{13} and x_{14} for a fixed $x_{12} = i$ and then summing over all $i, 0 \leq i \leq q$ yields $T(2t+1) = 4t(t+1)(t+2)/3, T(2t+2) = (4t^3 + 18t^2 + 32t + 21)/3$. Designs with $|G| = 2s, s > 1$ satisfy at least one of the following conditions: $x_k = x_l, x_k = q - x_l$, for some $k < l, k, l \in 12, 13, 14$. Summation over all $i, 0 \leq i \leq q$ yields $P(4t+1) = 24t(t+1) - 16Q(1), P(4t+3) = 24(t+1)^2 - 16Q(3), P(4t+2) = 24t^2 + 24t + 7 - 16Q(2), P(4t+4) = 24t^2 + 48t + 25 - 16Q(4)$, where again $Q(i) = N_{36}(4t+i)$. The number of distinct designs is $D_2^2(q) = T(Q) - P(Q)$. Since the index is 24, we have $N_2^2(q) = D_2^2(q)/24$. All other cases have been treated before.

The number $N_1(q)$ of nonisomorphic designs with a trivial automorphism group is then given by

$$N_1(q) = \frac{D(q) - 720 \sum_g N_g(q)/g}{720},$$

where g=2,3,4,6,8,12,24,36,60 and 720. The total number $N(q)$ of nonisomorphic 2-(6,3,2q) designs is the sum of all the $N_g(q)$'s and can be expressed as a polynomial of degree 5 (see Tables 2 and 3)

$$N(12t+i) = \sum_{s=0}^{5} a_k(i) t^k,$$

with coefficients $a_k(i)$ depending on the residue class i modulo 12, $1 \leq i \leq 12$.

i	$10a_5$	$10a_4$	$10a_3$	$10a_2$	$10a_1$	$10a_0$
1	864	720	1400	620	136	10
2	864	1080	1700	1005	321	40
3	864	1440	2120	1480	476	60
4	864	1800	2660	2075	821	130
5	864	2160	3320	2820	1176	190
6	864	2520	4100	3745	1771	340
7	864	2880	5000	4880	2436	480
8	864	3240	6020	6255	3411	760
9	864	3600	7160	7900	4536	1050
10	864	3960	8420	9845	6061	1530
11	864	4320	9800	12120	7836	2060
12	864	4680	11300	14755	10121	2860

Table 2 Coefficients of N(12t+i) enumerating nonisomorphic 2-(6,3,2q) designs.

In Table 4 we list the numbers of nonisomorphic 2-(6,3,2q) designs for all admissible orders of the automorphism group and values of q ranging from 1 to 20. It is worth noting that, while N_{720} is constant, $N_{60}, N_{36}, N_{24}, N_8$ are linear, N_{12}, N_6, N_4 are quadratic, N_3, N_2 are cubic polynomials. N_1, N and $D/720$ are fifth degree polynomials with the same leading coefficients. Consequently, $N(q)$ can be approximated from below by $D(q)/720$.

g	720	60		36		24		12			8		6			4		
i	a_0	a_1	a_0	a_1	a_0	a_1	a_0	a_2	a_1	a_0	a_1	a_0	a_2	a_1	a_0	a_2	a_1	a_0
1	0	6	1	2	0	0	0	36	6	0	0	0	36	4	0	9	1	0
2	1	6	1	2	0	12	2	36	6	0	3	0	36	4	0	18	-2	0
3	0	6	2	2	1	0	0	36	18	2	0	0	36	16	1	9	4	0
4	1	6	2	2	0	12	4	36	18	2	3	1	36	16	2	18	4	0
5	0	6	3	2	1	0	0	36	30	6	0	0	36	28	5	9	7	1
6	1	6	3	2	1	12	6	36	30	6	3	1	36	28	5	18	10	1
7	0	6	4	2	1	0	0	36	42	12	0	0	36	40	11	9	10	3
8	1	6	4	2	1	12	8	36	42	12	3	2	36	40	11	18	16	3
9	0	6	5	2	2	0	0	36	54	20	0	0	36	52	18	9	13	4
10	1	6	5	2	1	12	10	36	54	20	3	2	36	52	19	18	22	7
11	0	6	6	2	2	0	0	36	66	30	0	0	36	64	28	9	16	7
12	1	6	6	2	2	12	12	36	66	30	3	3	36	64	28	18	28	10

g	3				2				1					
i	a_3	a_2	a_1	a_0	$2a_3$	$2a_2$	$2a_1$	$2a_0$	$10a_5$	$10a_4$	$10a_3$	$10a_2$	$10a_1$	$10a_0$
1	96	-6	-3	0	204	-33	-9	0	864	720	-580	35	21	0
2	96	18	1	0	204	-9	1	0	864	1080	-280	-30	-4	0
3	96	42	3	0	204	69	-3	0	864	1440	140	-95	1	0
4	96	66	15	1	204	93	15	0	864	1800	680	50	-14	0
5	96	90	25	2	204	171	37	2	864	2160	1340	255	11	0
6	96	114	45	6	204	195	63	8	864	2520	2120	730	96	0
7	96	138	63	9	204	273	111	12	864	2880	3020	1325	251	20
8	96	162	91	17	204	297	145	24	864	3240	4040	2250	566	50
9	96	186	117	24	204	375	219	42	864	3600	5180	3355	1001	110
10	96	210	153	37	204	399	261	56	864	3960	6440	4850	1716	230
11	96	234	187	49	204	477	361	88	864	4320	7820	6585	2621	400
12	96	258	231	69	204	501	411	114	864	4680	9320	8770	3946	680

Table 3 Coefficients of $N_g(12t + i)$ enumerating nonisomorphic 2- (6,3,2q) designs with automorphism group of order g.

q	N_{720}	N_{60}	N_{36}	N_{24}	N_{12}	N_8	N_6	N_4	N_3	N_2	N_1	N
1	0	1	0	0	0	0	0	0	0	0	0	1
2	1	1	0	2	0	0	0	0	0	0	0	4
3	0	2	1	0	2	0	1	0	0	0	0	6
4	1	2	0	4	2	1	2	0	1	0	0	13
5	0	3	1	0	6	0	5	1	2	1	0	19
6	1	3	1	6	6	1	5	1	6	4	0	34
7	0	4	1	0	12	0	11	3	9	6	2.	48
8	1	4	1	8	12	2	11	3	17	12	5	76
9	0	5	2	0	20	0	18	4	24	21	11	105
10	1	5	1	10	20	2	19	7	37	28	23	153
11	0	6	2	0	30	0	28	7	49	44	40	206
12	1	6	2	12	30	3	28	10	69	57	68	286
13	0	7	2	0	42	0	40	10	87	81	106	375
14	1	7	2	14	42	3	40	16	115	98	163	501
15	0	8	3	0	56	0	53	13	141	135	235	644
16	1	8	2	16	56	4	54	22	178	156	338	835
17	0	9	3	0	72	0	69	17	213	207	463	1053
18	1	9	3	18	72	4	69	29	261	235	633	1334
19	0	10	3	0	90	0	87	22	306	300	836	1654
20	1	10	3	20	90	5	87	37	366	335	1101	2055

Table 4 Number N_g of nonisomorphic 2-(6,3,2q) designs with group of order g.

The counting formulae for 2-(6,3,2q) designs have been known to the author since 1984 and have been used to obtain the values listed in [9], [10]. They are published here for the first time. Since then Engel and Gronau [7] independently derived the formula for the number of distinct designs $D(q)$ by interpolating the known numbers for small values of q.

4 TACTICAL DECOMPOSITIONS

A majority of algorithms for the construction of 2-(v,k,λ) designs are based on incidence matrices. An element-block incidence matrix A of a 2-design satisfies the matrix equation $AA^T = (r-\lambda)I - \lambda J$, where $r = \lambda(v-1)/(k-1)$ is the repetition number and I, $J$are the unit and all one matrices of order v, respectively. A backtrack algorithm can be employed on a block by block or element by element basis. In either case its efficiency depends critically on the ability to recognize suitable partial solutions.The following inexpensive but discriminating inequality has been utilized in most of our recent searches.

Lemma. *Let $D(X,B)$ be a 2-(v,k,λ) design with b blocks and repetition number r. For $Y \in X$ and $C \in B$ let E be the corresponding induced configuration with w elements and c blocks. Denote by s the sum of all blocksizes in E and by p the sum of pairs they cover. Then*

$$(wr-s)d + p \leq \binom{w}{2}\lambda + (b-c)\binom{d+1}{2}, \qquad d = \lfloor \frac{wr-s}{b-c} \rfloor.$$

Proof. The sum of blocksizes of the remaining $b-c$ blocks $B\backslash C$ of D induced by Y is $wr-s$. To derive a lower bound on the total number of pairs they might cover we note, that from all distributions of n non-negative integers k_i with a fixed sum s, the one having the smallest sum of squares has the k_i's as close to each other as possible or, equivalently, has the smallest maximum difference $\mid k_i - k_j \mid$ over all i and j. In our case, this corresponds to $l = wr - s - d(b-c)$ blocks of size $d = \lfloor \frac{wr-s}{b-c} \rfloor$ and $b-c-l$ blocks of size $d+1$ covering $ld(d+1)/2 + (b-c-l)d(d-1)/2$ pairs. The inequality then follows from the fact that this lower bound cannot exceed the actual number of pairs $\lambda w(w+1)/2 - p$ of Y which must be covered in $B\backslash C$.

Another ingredient of a successful backtrack is an efficient isomorph rejection. Ideally, we would like to prune the search tree by eliminating any partial configuration which has been encountered before. One variation of backtracking called an orderly search generates all configurations in canonical form, thus eliminating the need of checking for duplicates [2], [8]. Unfortunately, for many design problems it appears difficult to use canonical forms in a recursive way. A popular technique employs only a partial isomorph rejection by utilizing selected automorphisms of the configuration under construction. For example, in the enumeration of all 2-(9,3,3) designs [19] we have used neighborhood graphs to reduce significantly the production of isomorphic solutions. For the 22521 nonisomorphic designs we have generated a total of about 50000 solutions.

If G is a group of automorphisms of a 2-(v,k,λ) design $D(X,B)$, let $X_1, X_2, \ldots, X_m$ be the point orbits and $B_1, B_2, \ldots, B_n$ be the block orbits of G. A tactical decomposition of D with respect to G is an m by n matrix $T(G) = (t_{ij})$ where t_{ij} is the number of elements from X_i contained in a block from B_j. In most cases G is a single automorphism of D of prime order generating orbits of size p and 1. From the matrix equation governing A we obtain relations for the entries of $T(G)$

$$\sum_{q=1}^{n} t_{iq}s_q = rr_i, \qquad \sum_{q=1}^{n} t_{iq}t_{jq}s_q = \lambda r_i r_j + (r-\lambda)r_i\delta_{ij},$$

where $r_i = \mid X_i \mid, s_j = \mid B_j \mid$, and $1 \leq i,j \leq m$. The search for 2-(v,k,λ) designs proceeds in two stages, each of which requires solving a smaller problem than the original. In the first stage we backtrack for all tactical decompositions $T(G)$ of the matrix A. For large designs we might want to impose a group action on T itself, thus inserting another stage. In any case, we end up with a set of feasible decompositions from which we keep only those which are nonisomorphic, and determine their automorphism groups. In the second stage we search for the actual circulants with valencies t_{ij} in the T-decomposition of A using the group of T for partial isomorph rejection. As in the case of tk-matrices, each of the resulting 2- (v,k,λ) designs has G as a subgroup of its automorphism group. We have used an efficient implementation of both of these stages to enumerate all 2-(25,4,1) and 2-(31,10,3) designs with nontrivial automorphism groups [11], [17] as well as many other families of designs possessing a given automorphism [20].

As an illustration of a tactical decomposition search we will enumerate all symmetric 2-(45,12,3) designs with an automorphism of order 11. A simple analysis shows that both X and B decompose into 4 orbits of length 11 and a fixed point. After completing the first stage we end up with 2 nonisomorphic T matrices

$$\begin{bmatrix} 1 & 4 & 4 & 3 \\ 4 & 1 & 4 & 3 \\ 4 & 4 & 1 & 3 \\ 3 & 3 & 3 & 2 \end{bmatrix} \qquad \begin{bmatrix} 5 & 2 & 2 & 3 \\ 2 & 5 & 2 & 3 \\ 2 & 2 & 5 & 3 \\ 3 & 3 & 3 & 2 \end{bmatrix}$$

(omitting the row and column corresponding to the fixed point and block). The first matrix yields a single self-dual design with group of order 11 and base blocks

$(0_0\ 0_1\ 1_1\ 4_1\ 5_1\ 0_2\ 1_2\ 3_2\ 9_2\ 0_3\ 2_3\ 7_3)$
$(0_0\ 1_0\ 3_0\ 4_0\ 10_1\ 0_2\ 4_2\ 6_2\ 10_2\ 4_3\ 7_3\ 9_3)$
$(0_0\ 1_0\ 5_0\ 7_0\ 3_1\ 5_1\ 6_1\ 8_1\ 9_2\ 6_3\ 9_3\ 10_3)$
$(0_0\ 2_0\ 5_0\ 2_1\ 4_1\ 8_1\ 6_2\ 7_2\ 10_2\ 1_3\ 2_3\ \infty)$
$(0_3\ 1_3\ 2_3\ 3_3\ 4_3\ 5_3\ 6_3\ 7_3\ 8_3\ 9_3\ 10_3\ \infty)$ mod (11)

The second matrix does not generate any solutions. The total CPU time for both stages and the analysis has not exceeded 5 minutes on an Apple MacII.

5 RANDOMIZED SEARCH

The methods discussed in this section are all based on the concept of *local optimization*. A combinatorial optimization problem can be specified as a set S of solutions (or states) together with a cost function c that assigns a value to each solution. An optimal solution corresponds to a global minimum of the cost function over the set Σ (this might not be unique). Given an arbitrary solution $S \in \Sigma$ we define a set T_S of transformations (moves), each of which can be used to change S to another solution S'. The set of solutions that can be reached from S by applying T_S is called the *neighborhood* of S and denoted by $N(S)$. In local optimization we repeatedly search through the neighborhood of S for a solution S' of lower cost $c(S') < c(S)$. Starting from some initial guess this procedure ends up in a locally optimal solution in the sense that none of its neighbors has lower cost. The hope is that some local minimum will be good enough. There are many ways of selecting the next candidate in the neighborhood. If $N(S)$ is small in size we can choose S' of minimum cost in $N(S)$ which results in a steepest descent type algorithm. If $N(S)$ is large we can keep selecting new candidates at random until one is found which has lower cost. It is sometimes desirable to accept a new S' if $c(S') \leq c(S)$, to increase the chance of reaching a global minimum, i.e. we accept not only "downhill" moves but also "sideway" moves. A basic property which a set Σ with the associated neighborhood and cost functions should have is that from any $S \in \Sigma$ it should be possible to reach a global minimum by a finite sequence of moves (some of which might be "uphill"). A computational requirement on $\Sigma(T, c)$ is that it should be easy to perturb solutions and to evaluate their cost. We are now in a position to explain a particular instance of local search for combinatorial designs which has enjoyed success in some cases.

A *hill-climbing* search for a t-(v, k, λ) design (X, B) is a local optimization which uses partial designs as the solution set Σ. A partial t-(v, k, λ) design (X, B') is

a set B' of blocks of size k from X such that any t-subset of X is contained in at most λ blocks. The cost associated with a partial design (X, B') is the difference $|B| - |B'|$, where the final number of blocks $|B|$ can be expressed in terms of the parameters [2]. From a partial design (X, B'), we move to another one by adding a new block β to B' and possibly deleting some other blocks which violate the definition of a partial design. To make this efficient we should avoid creating too many conflicts by choosing only those new blocks which contain a large number of unused t-subsets. A successful hill-climbing algorithm was used by Stinson [21] for Steiner triple systems (which are $2-(v,3,1)$ designs). A new triple (x,y,z) is formed by selecting at random two unused pairs (x,y) and (x,z). If the third pair (y,z) already appears in the partial design we delete the unique block containing it. After a move the cost goes down by one or remains the same. Implementations of this algorithm exist in which the computational complexity of a move is a constant independent of v. Stinson's algorithm works well for all triple systems and related configurations such as Latin squares, etc. Hill-climbing does not work well if $k > 3$ or $t > 2$. One possible explanation of this is that almost all partial designs created during the search contain a large number of subconfigurations which are "forbidden" in the final design. This causes the algorithm to get stuck by failing to make any progress. In order to find a successful and generally applicable hill-climbing search we must learn how to efficiently recognize and avoid bad partial solutions.

In *simulated annealing* we employ local search augmented with a mechanism to avoid entrapment in poor local minima. This is accomplished by allowing an occasional uphill move with the help of a random number generator and a control parameter T called the "temperature". An uphill move of size Δ is accepted with probability $\exp(-\Delta/T)$ which diminishes as the temperature declines. Note that, for a fixed T, small uphill moves have higher probabilities than large ones. A typical implementation of simulated annealing uses a pair of nested loops governed by two parameters, a cooling ratio q, $0 < q < 1$ and an integer temperature length L (see [9], [10]). In each instance of the inner loop, which is executed L times at constant temperature T, we pick a random neighbor S' of the current solution S and calculate the $\Delta = c(S') - c(S)$. If $\Delta \leq 0$ we set $S := S'$. If $\Delta > 0$ we accept S' with probability $\exp(-\Delta/T)$. In each pass of the outer loop the temperature is reduced by a factor $q < 1$, $T := qT$. The outer loop starts from some initial temperature T_0 and terminates when no further progress is being made, which corresponds to a "frozen" state.

We have implemented two algorithms which use simulated annealing to search for 2-(v,k,λ) designs. Both algorithms attempt to build a v by b incidence matrix with row sums r, column sums k and the inner products of two distinct rows λ. In the first algorithm the solution space Σ consists of all v by b {0,1}

matrices. A move in Σ consists in changing the parity of a randomly chosen entry: $a_{ij} := 1 - a_{ij}$. In the block design a move corresponds to adding element i to block j if it is not there and deleting it if it is there. The cost funtion is a weighted sum of squares of the deviations of row sums from r, column sums from k and products of rows from λ. The weights corresponding to rows, columns and row products should be chosen in the ratio $v : b : 1$, respectively. In the second algorithm we have worked with {0,1} matrices with constant row sums r and column sums k. A move consists in changing the parity of all entries in a 2 by 2 submatrix with row and column sums 1. In addition to preserving row and column sums it is known that any two matrices in this space can be obtained from each other by a finite sequence of such transformations. In the block design a move corresponds to swapping two elements belonging to two different blocks. The cost function is the sum of squares of deviations from λ of the $v(v-1)/2$ row products. To set the various control parameters in our algorithms we followed some generally accepted rules [9], [10]. The recommended cooling ratio is $q = 0.95$. The temperature length L should be a fixed multiple of the size of a typical neighborhood. The neighborhoods of the two proposed spaces are vb and $v(v-1)(r-\lambda)^2/2$, respectively. The size factors ranged from 10 to 50 in the first algorithm and from 1 to 5 in the second algorithm depending on the overall size of the incidence matrix and the available computer resources. The initial temperature 1 worked well for unit sums of weights in the first case and with no restrictions in the second case. The initial success rate (the ratio of accepted solutions in the inner loop to L) should be about 30% to insure adequate mixing of the solutions. Finally, the search was terminated if for 5 consecutive instances of the outer loop the cost of the best solution remained the same.

We now describe the results of our experiments. To our surprise, the first algorithm performed considerably worse than the second. Its only advantage is its much wider applicability to designs which are either pairwise balanced (k and r are variable) or (r, λ) designs (k variable). To be successful it required large size factors and more overall trials. The second algorithm performed well on small designs (with $vb < 400$). It sucessfully generated symmetric 2-$(4t-1, 2t-1, t-1)$ designs for $t < 6$ and 2-$(k^2+k+1, k+1, 1)$ designs for $k < 9$. For designs with small block size k a local search in which sideways moves were allowed worked best. This corresponded to setting the initial temperature to 0 (frozen state). For $k = 3$ our algorithm worked every time, usually terminating during the first pass of the outer loop with about the same efficiency as hill-climbing. The results for $k = 4$ are summarized in Table 5.

Each entry in this table is the average number of conflicts based on ten runs for each value of the parameters. For a fixed λ the conflicts grow linearly as

v \ λ	1	2	3	4	v \ λ	1	2	3	4
13	0	0	0	0	28	8.8	3.4	0.3	0
16	0	0	0	0	29			0.3	
17			0		31		5.1		0
19		0.3		0	32			0.6	
20			0		33			1.2	
21			0		34		4.8		0
22		2.3		0	36			0.3	
24			0		37	15	6	0.6	0
25	6.5	3	0	0	40	19.4	7.1	1.6	0

Table 5 Average number of conflicts in local search for 2- $(v, 4, \lambda)$ designs.

a function of v, the slope declining as λ increases. For $\lambda \geq 4$ our local search succeeds every time in a similar fashion to that for $k = 3$. There seems to be a general pattern. For every k there seems to be a $\lambda = \lambda(k)$ such that for every $\lambda \geq \Lambda$ local search is perfect ($\lambda(3) = 1$, $\lambda(4) = 3$, $\lambda(5) = 6$, etc.)

REFERENCES

[1] T. Beth, D. Jungnickel, H. Lenz, *Design Theory* (Bibliographishes Institut, Manheim-Wien-Zurich, 1985).

[2] M.J. Colbourn, Algorithmic aspects of combinatorial designs: A survey, *Ann. Discrete Math.* 26 (1985), 67–136.

[3] M.J. Colbourn, R.A. Mathon, On cyclic Steiner 2-designs, *Ann. Discrete Math.* 7 (1980), 215–253.

[4] F.N.Cole, L.D. Cummings, H.S. White, Complete classification of the triad systems on fifteen elements, *Memoirs of the National Academy of Sciences U.S.A.* 14, Second memoir (1919), 1–89.

[5] I.A. Faradzev, Constructive enumeration of combinatorial objects, *Problemes Combinatoires et Theorie des Graphes* (CNRS, Paris, 1977) 131–135.

[6] K. Engel, H.-D.O.F. Gronau, On 2-(6,3,l) designs, *Rostock. Math. Kolloq.* 34 (1988), 37–48.

[7] P.B. Gibbons, *Computing techniques for the construction and analysis of block designs*, PhD Thesis, University of Toronto (1976).

[8] A.V. Ivanov, Constructive enumeration of incidence systems, *Ann. Discrete Math.* 26 (1985), 227–246.

[9] D.S. Johnson, C.A. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation. Part I, Graph partitioning *Operations Research* 37 (1989), 865–892.

[10] D.S. Johnson, C.A. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation. Part III, Graph coloring and number partitioning, *Operations Research* 39 (1991), 378–406.

[11] E.S. Kramer, D.M. Mesner, t-designs on hypergraphs, *Discrete Math.* 15 (1976), 263–296.

[12] E.S. Kramer, S.S. Magliveras, R. Mathon, The Steiner systems S(2,4,25) with nontrivial automorphism group, *Discrete Math.* 77 (1989), 137–157.

[13] E.S. Kramer, D.W. Leavitt, S.S. Magliveras, Construction procedures for t-designs and the existence of new simple 6-designs, *Ann. Discrete Math.* 26 (1985), 247–274.

[14] D.L. Kreher, S.P. Radziszowski, Finding simple t-designs by using basis reduction, *Congressus Numerantium* 55 (1986), 235– 244.

[15] D.L. Kreher, S.P. Radziszowski, The existence of simple 6- (14,7,4) designs, *J. Combinatorial Theory* 43A (1986), 237–243.

[16] S.S. Magliveras, D.W. Leavitt, Simple 6-(33,8,36) Designs from $PGL_2(32)$, *Proceedings of the Durham Computational Group Theory Symposium* (Academic Press, 1983), 337–352.

[17] R. Mathon, Symmetric (31,10,3) designs with nontrivial automorphism group, *Ars Combinatoria* 25 (1988), 171–183.

[18] R. Mathon, Computational methods in design theory, *Surveys in Combinatorics, 1991 (Guildford, 1991)* 26 (1985), 101–117.

[19] R. Mathon, D. Lomas, A census of 2-(9,3,3) designs, *Austral. J. Combin.* 5 (1992), 145–158.

[20] R. Mathon, A. Rosa, Tables of parameters of BIBD's with $r \leq 41$ including existence, enumeration, and resolvability results, *Ann. Discrete Math.* 26 (1985), 275–308.

[21] D.R. Stinson, Hill-climbing algorithms for the construction of combinatorial designs, *Ann. Discrete Math.* 26 (1985), 321–334.

4

FINDING DESIGNS WITH GENETIC ALGORITHMS

Daniel Ashlock

Mathematics Department,
Iowa State University,
Ames, IA, 50010
danwell@iastate.edu

ABSTRACT

In this paper we use simulated evolution to solve three design theoretic problems, direct decomposition of a complete graph into smaller complete graphs, decomposition of a complete graph into smaller complete graphs with difference sets, and the seating of twelve people at three four person tables for multiple nights in an even handed fashion. The paper is intended as a tutorial for someone familiar with design theory but unfamiliar with genetic algorithms.

1 GENETIC ALGORITHMS

Genetic algorithms are algorithms that use simulated evolution to get their work done. In this article we will be using them to evolve some well known combinatorial designs. The notion of genetic algorithms is credited to John Holland in his book *Adaption in Natural and Artificial Systems* [5]. A good introduction to genetic algorithms is provided in David Goldberg's book *Genetic Algorithms in Search, Optimization, and Machine Learning* [2]. A book that might help a novice genetic programmer is the *Handbook of Genetic Algorithms* edited by Lawrence Davis [1]. A quick introduction can be found in Stephanie Forrest's article in *Science* [3].

Our genetic algorithms will operate on a population of data structures that could hold a design and do hold a collection of subsets of a set that are the right size to be a design. In biological evolution, fitness is measured by reproductive success. We will turn this notion on its head and in each generation grant

W. D. Wallis (ed.), Computational and Constructive Design Theory, 49–65.
© 1996 *Kluwer Academic Publishers.*

those design-like data structures we deem most fit, because they are closest to being designs, greater reproductive success. After many generations there is good hope that the fitness bias in the reproduction of structures will result in a structure so fit it is actually a design.

We will look at three searches in this article. In section 2 we will directly partition a complete graph into smaller complete graphs. In section 3 we will evolve difference sets that can be used to do the same task. In section 4 we will evolve a scheme for seating twelve people at three tables on several nights so that each person sits with each other person almost the same number of times. All genetic algorithms have approximately the form given in figure 1.

Generate a random population of structures.
Evaluate the fitness of each structure.
Repeat
    Select pairs of parents with a fitness bias.
    Pairs of parents use crossover to produce pairs of children.
    The children are mutated.
    Children replace members of the current population.
    Evaluate children's fitness.
Until we find a solution.

Figure 1 The steps of a genetic algorithm

Developing a genetic algorithm requires the following steps. First you decide on a data structure that can hold solutions to your problem. These data structures are the *creatures* or *genes* on which the algorithm operates. Next you find a way to fill the data structures with semantically valid but otherwise random structures. While designing your data structures and their semantics you come up with a *fitness function*, usually a function from the space of data structures into the reals or some other totally ordered set. You come up with a method or methods of combining two of your data structures (parents) to produce two others (children) so that each of the parents contributes to each of the children. These methods are called your *crossover operators* and are used as the algorithm's analog of biological sex. You develop a routine or routines that make small changes in a single creature, these are termed the *mutation*

operators. With all these parts in hand you can implement the basic genetic algorithm outlined in Fig. 1.

The trickiest part of designing a genetic algorithm for a task like locating combinatorial designs is to come up with a notion of fitness so that increasing fitness actually represents progress toward locating the design. At present this process is still an art and we will give basic, simple fitness measures in our examples.

2 DIRECT DECOMPOSITION OF K_n INTO COPIES OF K_m

0	1	4
6	2	1
3	4	2
0	5	6
2	5	4
6	2	5
1	2	4

Figure 2 An example gene

In this section we will try to produce a decomposition of K_n into copies of K_k, a $2-(n,k,1)$ design. If we let $e = n(n-1)/k(k-1)$ then the data structure we will use to hold our putative designs is an array of e arrays of k points. We will term these data structures *genes* and the arrays of k points *cells.* When the algorithm terminates with a design the cells will be blocks of the design. The semantics of this data structure require the k points in each cell be distinct, other than this there are no restrictions. Generating random genes is easy, we simply choose k of n points e times to fill the data structure. The fitness of a gene will be the size of the set of distinct unordered pairs that occur within any of its cells. Suppose that $n = 7$, $k = 3$, and $e = 7$ so that we are searching for the Fano configuration. Then the gene in figure 2 contains the unordered pairs {{0,1}, {0,4}, {0,5}, {0,6}, {1,2}, {1,4}, {1,6}, {2,3}, {2,4}, {2,5}, {2,6}, {3,4}, {4,5}, {5,6}} and so would have a fitness of 14. Viewed as copies of K_3 it covers fourteen of the twenty-one edges of K_7. A gene with a fitness of $\frac{1}{2}n(n-1)$ is in one of the designs we are searching for.

We will use a single crossover operator called *two point crossover*. We choose two locations $1 \leq x < y \leq e$ in the parents genes viewed as a list of e cells. The cells of both children will be copies of the parents cells in corresponding locations. The first child will get his cells in locations $1 \ldots x$ and $y + 1 \ldots e$ by copying the corresponding cells in the first parent. The first child's cells in locations $x + 1 \ldots y$ will be copied from the second parent. The second child will, in each location, copy whichever cell in the corresponding location for the two parents that the first child did not copy. Figure 3 gives and example of crossover. This crossover produced children that were no better than their parents. This is quite common in an evolutionary search, especially when we get close to the goal. Other methods of crossover are discussed in section 5.

Crossover Points	Parent 1			Parent 2		
	0	1	4	3	1	4
	6	2	1	5	2	0
$x\triangleright$	3	4	2	1	2	5
	0	5	6	0	3	6
	2	5	4	2	4	6
$y\triangleright$	6	2	5	1	0	3
	1	2	4	2	3	0
Fitness	14			16		

Crossover Points	Child 1			Child 2		
	0	1	4	3	1	4
	6	2	1	5	2	0
$x\triangleright$	3	4	2	1	2	5
	0	3	6	0	5	6
	2	4	6	2	5	4
$y\triangleright$	1	0	3	6	2	5
	1	2	4	2	3	0
Fitness	14			15		

Figure 3 An example of crossover

We will use a pair of mutation operators. The first, *mutation 1*, consists of selecting a random cell in a creature and replacing it with a new, random cell. The second, *mutation 2*, picks a cell at random, deletes a single point from that cell, and replaces it with a point the cell did not previously contain. Figure 4 shows the result of applying the two sorts of mutations to the two children shown in figure 3.

Mutation Point	Child 1			Child 2		
	0	1	4	3	1	4
	6	2	1	5	2	0
	3	4	2	1	2	5
	0	3	6	0	5	6
	2	4	6	2	5	4
	1	0	3	6	2	5
▷	2	0	5	4	3	0
mutation:	# 1			# 2		
Fitness	17			15		

Figure 4 An example of mutation

Replacing the sixth cell in the first child with a new random cell helped quite a bit. Replacing the 2 in the sixth cell of the second child with a 4 had no effect on the child's fitness.

A natural question at this point is, "why not try and simply try to evolve good cells?" The answer is that there is no such thing as a good cell in this example. While the individual creatures in our population have objectively computable finesses the fitness contribution of each cell in the creature is dependent on exactly which other cells are present. In the next section when we evolve difference sets instead of complete decompositions there will be a notion of good cell.

2.1 Building the algorithm.

With a gene structure and its semantics, mutation, and crossover operators in hand we can now describe our genetic algorithm. Two of the steps in the general outline of the algorithm given in figure 1 are selecting pairs of parents with a fitness bias and replacing members of the current population with new children. Exactly how the fitness bias in selection of parents and the decision of whom to kill to make a place for the children are, together, called the *model of evolution* used by the genetic algorithm. In all three examples in this paper we are using a model called *tournament selection*. To perform tournament selection the population is shuffled and then divided into groups of four creatures. The best two are selected to be parents and their children replace the worst two in the group.

An important consideration when running a genetic algorithm is the size of the population to use. The model of evolution sometimes influences population size. The tournament selection model wants a population size that is a multiple of four, for example, though this is not an important consideration. The importance of population size grows out of the tradeoff between search speed and search breadth. A small population has faster generations and so searches faster. A large population has more cells in it at the beginning and so has more raw material for the crossover operator to work with. New cells are constantly being introduced by mutation but, if mutation is not to utterly disrupt the progress toward a solution, these new cells cannot come in at too high a rate. Conventional wisdom holds that larger population converge more slowly but are more likely to eventually produce a correct answer. In this example we will run populations of size 60 and 600 for comparison.

The next consideration in building our algorithm is the mutation rate and the crossover rate. In figure 1 the outline simply states that the children produced by crossover and then are mutated. Many genetic algorithms have a null mutation and crossover in which the creatures being crossed over or mutated are left unchanged. It is necessary to decide what fraction of the time each of the available mutation or crossover operators is applied. These fractions can be made a function of time (time is measured in generations). It is instructive to note that a genetic algorithm with a population size of 1, no crossover, and a rate of mutation that decreases over time is a form of simulated annealing. Figure 5 gives the rates for the genetic algorithm used in this section.

Operation	Probability
Two Point Crossover	100%
Mutation 1	20%
Mutation 2	80%
Null mutation	20%

Figure 5 Crossover and mutation probabilities in the first algorithm.

2.2 Results of the first algorithm.

In testing the algorithm we found Steiner systems on 7, 9, and 13 points. In the run finding Steiner systems on 13 points a few of the runs did not finish within 20,000 generations, an arbitrary cutoff value. We will see in the next section

that much larger decompositions are feasible with a more efficient structure used while evolving. The results are summarized in table 1

n,k	Population Size	Generations $\mu \pm \sigma$	Generations Minimum
7,3	60	45.9±27.1	10
7,3	600	36±14.2	6
9,3	60	403±323	45
9,3	600	114±51	49
13,3	60	7510±5180	287

Table 1 First algorithm, outcome summary.

The statistics were taken over 100 different ecologies and each ecology was allowed to run either until a correct design appeared in the population or it reached a cutoff of 20,000 generations. All 100 ecologies in each of the first four data sets did produce the desired Steiner system. Some 20 of the ecologies in the last data set were cut off at 20,000 generations and so the only meaningful statistic for that data set is the minimum generations.

Earlier we said that a larger population converges more slowly but is more likely to produce a correct answer, a statement somewhat at odds with the table above. The large population sizes seem to converge faster. This is in part an artifact of measuring time in generations: each generation in the runs with population size 600 took ten times as long and so the computer spent far *more* time grinding to produce those designs. In addition we were running examples so small that they were quite likely to converge eventually with the computer resources available, a desktop PC. If we had set the generation cutoff lower a higher proportion of the $n = 9$ runs would have terminated in the runs with a population size of 600 as compared to to the runs with a population size of 60.

The five data sets above also have one obvious feature in common, a titanic standard deviation. This is a useful and exploitable phenomenon in a search for larger configurations. When the standard deviation is very high the correct technique is to run a large number of small populations *at the same time*, stopping when the first finds the desired configuration. Looking at the column of table 1 that gives the minimum generations, one sees it is practical to control the time needed to locate a configuration in this fashion. This phenomenon will become somewhat more pronounced in the next section.

3 DECOMPOSING K_n INTO K_k WITH DIFFERENCE SETS

In this section we will modify the genetic algorithm in the last section by placing a different interpretation on the gene and rewriting the fitness function. In the last section the gene we were using was an explicit list of the $e = \frac{n(n-1)}{k(k-1)}$ vertex sets of copies of K_k (blocks) that decomposed K_n. We will keep the same data structure but instead of listing all the copies of K_k we will list only $f = \frac{n-1}{k(k-1)}$ copies of K_k that form a difference set. For the translations $(mod\ n)$ of the f copies of K_k to decompose K_n we need every difference in the range $1 \leq d \leq \frac{n-1}{2}$ to appear as the difference between the vertices of some copy of K_k. This suggests the following fitness function. The fitness of a list of f copies of K_k with vertices drawn from the set $\{0, 1, \ldots, n-1\}$ is the number of distinct differences in the specified range that appear between vertices of the same copy of K_k. Imagine we are decomposing K_{19} into copies of K_3, i.e. finding a Steiner triple system on 19 points. Then the gene is three sets of three vertices. An example of a gene and accompanying fitness computation appears in figure 6.

Gene			Differences
13	5	4	1,9,8
6	12	1	5,6,8
3	10	8	2,5,7
Differences:			1,2,5,6,7,8,9
Fitness:			7

Figure 6 An example gene and its fitness while evolving a decomposition of K_{19} into copies of K_3.

The genetic algorithm is the same using the same crossover and mutation operators and probabilities as given in figure 5. We are still using tournament selection as our model of evolution. We will do all the runs on a population of size 60. The results are given in table 2.

Because we are using difference sets (mod n) the trials shown in table 2 only search for decompositions when $n \equiv 1\ (mod\ 6)$. It is interesting to note that until $n \geq 31$ it might well have been more profitable to simply test random examples than to run the genetic algorithm. This is an important benchmark you should have in mind whenever you are developing a genetic algorithm for search or optimization: as soon as you can generate random creatures and test

n,k	Population Size	Generations $\mu\pm\sigma$	Minimum
7,3	60	1±0	1
13,3	60	2.07±2.27	1
19,3	60	14.8±20.9	1
25,3	60	88.5±131	1
31,3	60	425±560	15
37,3	60	525±610	26
43,3	60	1220±1350	36
49,3	60	1530±1890	42
55,3	60	2350±2790	49

Table 2 Second algorithm, outcome summary.

their fitness do a quick check of the parts of fitness space that are hit by random examples. Try this to locate a difference set for decomposing K_{21} into copies of K_5, the results are swift.

After doing the above data acquisition runs I rebuilt the algorithm to run until it found a single example. We were able to find Steiner triple systems on n=7, 13, 19, 25, 31, 37, 43, 49, 55, 61, 67, 73, 79, 85, 91, 97, 103, 109, 115, and 121 points, as well as n=13, 25, 37, 49, k=4, and n=21, k=5. A few examples of the resulting difference sets are given in figures 7–9. The creature in figure 7 required 1425 generations, the creature in figure 8 appeared in the initial random population, and the creature in figure 9 required some 39059 generations. It seems that increasing n is much cheaper than increasing k.

In the preceding section we mentioned that when we are doing our decomposition via difference sets, there is a notion of a cell being "good". A cell is good if it realizes $k(k-1)/2$ differences. Randomly chosen cells do not have this property and the cells in a correct difference set must have this property. This isn't a property we enforce in the semantics of the data structure nor is it one our mutation operators preserve. It was the difficulty of creating mutation operators that preserve this property without using a lot of computer time that caused me to choose to explicitly ignore this notion of good cells and let evolution use it for me, implicitly. A low cost way to try incorporating this notion to some degree is to find, by testing random examples, good cells for the initial population and then simply let evolution penalize bad mutation. In the Steiner system examples I used in most of my runs for this tutorial, evolution

quickly finds good cells. If these methods are used to assault in earnest cases where k is substantially larger than 3, incorporating an initial seeding of the population with good cells may well be profitable.

Cells					
63	99	50	13	89	12
48	86	0	13	33	86
86	92	11	15	62	59
89	34	85	74	80	34
26	94	33	82	79	94
58	71	0	76	79	81
27	86	96	25	60	72
51	56	67	64	78	55
39	92	98	.	.	.

Figure 7 Difference set for a Steiner triple system on 103 points.

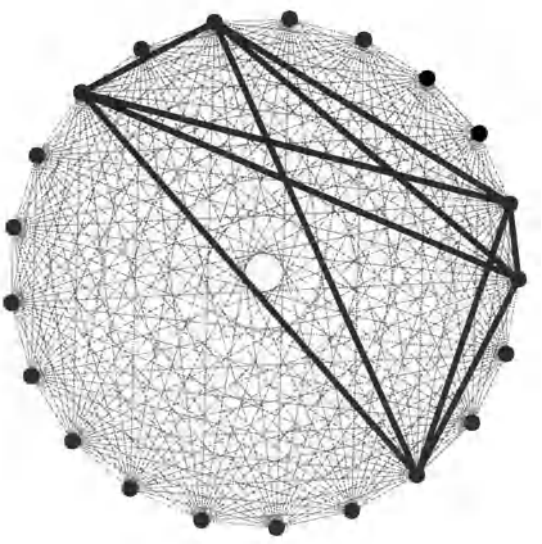
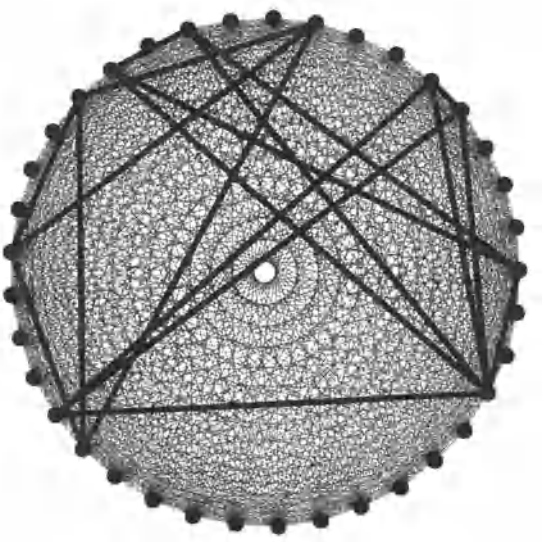


Figure 8 Difference set for decomposing K_{21} into copies of K_5.



Cells			
15	33	3	32
14	19	23	29
3	0	24	26

Figure 9 Difference set for decomposing K_{37} into copies of K_4.

4 SEATING PEOPLE ON SUCCESSIVE NIGHTS

In this section we want to attack a substantially different problem. Suppose we have twelve people whom we wish to seat at three four-person tables for r nights. We want to find seating arrangements that will seat each pair of people at the same table as close as possible to the same number of times, a *resolvable regular graph design*, $RRGD(12, 4; r)$, or *bridge club design*, as described in [6]. The first step is to construct the data structure. I will use a creature composed of r cells each of which consists of a list of the numbers 1 through 12. This is interpreted as a seating arrangement by taking each cell to be the seating arrangement for a night and letting the first, middle, and last four numbers in each list specify the groups that sit together.

When seating people for r nights each person accumulates $3r$ partners which are to be divided as evenly as possible among the other 11 people. This implies that a good seating arrangement has each pair of people seated together either roof or floor of $\frac{3r}{11}$ times. Call these numbers the acceptable pair numbers. To compute the fitness of a seating arrangement I have the computer find the number of times each pair of people are seated together and then give a fitness of 66 minus the number of pairs that are seated together a number of times that is not an acceptable pair number. This makes the fitness simply the number of pairs that attain an acceptable pairing number.

The genetic algorithm in this section uses tournament selection as its model of evolution. It has one crossover operator, two point crossover of the list of cells, as in the algorithms used in the preceding sections. It has a single nontrivial mutation operator which is used 80% of the time, pick a random cell and transpose two elements of the list of numbers. Null mutation is used the remaining 20% of the time.

In figure 10 are seating arrangements for 8, 9, 10, 12, and 13 nights. The case that works out even, $n = 11$, was enormously hard for the genetic algorithm, it never even got close. The other cases all spit out answers in a few minutes (176-2157 generations, one try needed).

5 SOME OPTIONS IN DESIGNING A GENETIC ALGORITHM

In this section we will discuss some of the other models of evolution, crossover operators, and mutation operators that are possible in a genetic algorithm. A model of evolution is said to be *elitist* if the most fit creature or creatures are immune from death in a given generation. The model of evolution I chose to genetically search for designs in our three examples, tournament selection, is an elitist model that does pretty well at preserving the diversity of a population. To recap, tournament selection shuffles the population into randomly chosen groups of four creatures and, within each group of four, the best two cross over and replace the worst two. This means that the two creatures with the highest fitness values in the population in a given generation cannot die. It also means that the most fit creature has only one child, a desirable quality if one does not want the population to be able to rush into local optima.

r	Table 1				Table 2				Table 3				Generations
8	7	6	10	2	12	4	9	3	5	11	8	1	588
	7	11	4	9	10	1	6	5	12	2	8	3	
	3	7	5	8	6	11	10	9	4	2	1	12	
	12	9	7	5	11	3	6	2	8	4	10	1	
	7	2	1	9	3	4	5	6	8	12	10	11	
	12	10	2	5	9	8	4	6	3	1	11	7	
	3	10	1	9	2	5	11	4	6	8	7	12	
	12	1	11	6	5	9	2	8	10	7	4	3	
9	6	11	2	5	8	1	7	12	3	9	10	4	929
	8	4	3	5	10	11	12	6	7	1	2	9	
	8	2	1	10	12	3	7	6	9	4	5	11	
	6	4	2	9	3	7	10	11	1	12	5	8	
	2	6	8	3	10	5	9	7	11	12	4	1	
	4	6	7	1	8	10	11	5	12	2	9	3	
	6	8	4	10	3	11	9	1	5	7	12	2	
	6	1	5	3	10	2	4	12	9	8	7	11	
	4	2	7	11	10	1	5	3	12	6	8	9	
10	7	12	1	5	10	8	9	3	11	4	6	2	527
	5	6	7	3	4	10	8	11	12	1	9	2	
	1	11	9	10	12	7	6	8	3	2	4	5	
	2	7	10	4	3	5	8	1	6	11	9	12	
	3	12	11	2	5	9	10	7	8	4	6	1	
	1	3	12	4	2	7	8	9	10	6	5	11	
	9	11	4	7	3	6	10	12	5	1	8	2	
	2	6	1	10	9	12	4	5	8	11	7	3	
	10	4	1	7	8	11	5	12	2	6	9	3	
	9	4	6	5	3	11	7	1	2	10	12	8	
12	2	7	11	8	9	12	6	5	4	10	3	1	176
	4	5	3	10	12	7	9	8	2	1	6	11	
	9	4	7	10	3	5	11	2	6	12	1	8	
	10	11	8	9	12	1	4	2	7	6	5	3	
	4	2	7	5	3	8	12	10	9	11	1	6	
	4	2	12	6	9	3	1	7	8	5	11	10	
	12	11	9	3	4	8	6	5	2	7	10	1	
	11	12	4	3	6	2	10	9	7	1	8	5	
	10	6	12	7	5	9	3	2	1	8	11	4	
	1	5	9	4	6	7	8	3	12	2	11	10	
	3	1	6	10	4	8	2	9	12	7	11	5	
	6	4	11	7	10	12	5	1	8	9	3	2	
13	9	6	3	7	10	11	1	4	12	8	5	2	2157
	2	5	4	8	1	6	11	12	3	10	7	9	
	9	12	8	1	2	3	11	7	6	5	4	10	
	2	1	6	7	10	3	8	5	11	4	12	9	
	1	3	9	5	12	2	4	6	10	8	7	11	
	5	11	1	10	4	9	7	8	3	12	2	6	
	10	7	1	2	4	5	3	12	8	9	11	6	
	12	10	7	4	2	8	11	3	6	1	9	5	
	5	7	8	2	10	9	1	12	4	11	6	3	
	12	6	8	10	2	9	11	5	1	3	7	4	
	2	4	1	11	7	5	12	9	3	10	6	8	
	9	2	10	4	5	7	6	11	12	1	8	3	
	8	4	6	1	9	3	10	2	7	5	12	11	

Figure 10 Seating arrangements for 8, 9, 10, 12 and 13 nights.

Tournament selection specifies how to select parents and place children in the population in a unified fashion. In other models these processes are divorced and one chooses a parent selection method and a child placement method separately. In *roulette selection* the parents are chosen with probability proportional to their fitness, imagine a roulette wheel with a sector for each parent whose angular width is proportional to fitness. This method does have a fair fitness bias, but an exceptionally fit creature that arises early in evolution can have a very large number of children in a single generation and come to dominate the population. This is especially true if grandchildren of this creature recombine in a fashion that imitates cloning of the grandparent. If the high, early fitness

results from a local optima the fast convergence can be a trap. A somewhat less extreme method of selection is *rank selection* in which a population of n creatures is ranked by fitness with rank 1 belonging to the least fit creature and rank n belonging to the most fit creature. The creatures are then selected to be parents with a probability proportional to their rank.

To see if a model of evolution is elitist you must look at the method of placing children into the population. You might place the children at random or have them replace population members selected with an anti-fitness bias. Neither of these methods is elitist because there is at least a small probability that the most fit creature will be killed. One way to obtain elitism is to choose random individuals from the population and replace them only if they are less fit than the child. There are a large number of variations possible and no reason for preferring one to another in general. Usefulness of a model of evolution is problem dependent.

All three of the genetic algorithms implemented in this tutorial use two point crossover. This involves swapping the genetic material from the parents to produce children between two crossover points and is thought to be a good method because it gives no special status to the ends of the genome. Another common method *one point crossover* exchanges all genetic material after a single randomly chosen point. This is a very simple crossover method but it does probabilistically link genetic loci near one or the other end of the gene. Both these crossover methods assume that the gene is itself a list or string of some sort, as was the case in all the examples presented here. When the crossover is operating on more exotic structures more exotic operators are needed and examples may be found in [1]. Even when the gene is a list much more exotic crossover operators are possible, including adaptive ones. Suppose, for example, that our gene was a list of 161 zeros and ones and that, because of the way we coded the problem, the fitness of the creature is highly dependent on the contents of loci 3, 5, 17, 66, 121, and 137. It would then be advantageous for crossover to preserve a set of positions including those loci.

One method to attain this sort of crossover is to augment our creatures with a crossover template, itself a string of ones and zeros. The first parent chosen provides the crossover template with the ones and zeros in the template showing which way the loci go from parents to children. The loci corresponding to ones in the crossover template go from parent 1 to child 1 while those loci corresponding to zeros go from parent 1 to child 2. These crossover templates then themselves are allowed to evolve with their own model of evolution. This is an example of an advanced technique whose performance is at best difficult to assess.

Mutation operators are supposed to make small changes in the creature mutated. In the examples provided in this tutorial the largest change made by a mutation was the replacement of a single block in a design. In addition this mutational change was made with a large probability to a single location in the gene. Another way of implementing mutation is to have a low probability of mutation *per loci* so that one average a very small number of loci are changed but the possibility exists for a large number of changes to occasionally occur.

When the object of a genetic algorithm is rich in algebraic structure the algebraic operations may be used as a source of mutations. If maintaining the diversity of a population seems to be important, because of observer rapid loss or for theoretical reasons, algebraic symmetries that produce equivalent (equally fit) objects can be used as neutral mutations. This may be of value because two algebraically equivalent structures may be genetically inequivelent. In addition, neutral mutation of elite members of a population can prevent the genes of the elite from dominating a population to the point of stagnation.

6 CONCLUSIONS AND SUGGESTIONS

In this paper we have solved three design theory problems with, essentially, a single genetic algorithm, varying the data structure, fitness function, and mutation operators. In all three problems the data structure is a list of cells which represent blocks or parallel classes in the design we are trying to construct. The fact that we could solve three different problems with modest changes in a single program illustrates one of the nicest features of genetic algorithms, their problem independence. Once you have a genetic algorithm running you can quickly produce new versions. The modifications of the genetic algorithm used to find difference sets that were required to make it locate seating diagrams required some two hours of work, including debugging. Another nice feature of genetic algorithms is that they parallelize well. If one has a multiprocessing machine available then a genetic algorithm can be parallelized by simply running a population on each processor. As we see from the minimum time columns in tables 1 and 2 running multiple ecologies in parallel is well worth the trouble.

The genetic algorithm presented in this paper can be implemented from a standing start with a day or two of effort by even a moderately competent programmer. It is a stripped-down, minimal example of the art. There are a large number of other techniques in the genetic algorithms literature that

can be added with a possible increase in performance. One can add nontrivial geographical structure to the population in some fashion. Once can make the probability of mutation an increasing function of time. The initial random, unfit, diverse population has little need of mutation but the later, homogeneous, relatively fit population may need to be shocked out of a local optimum or toward an upward slope. One can implement *Lamarckian* mutations which do a local search of the configurations attainable by mutation and choose the best. This latter example suggests a source of richness for genetic algorithms. Both correct evolutionary biology and its wrong turns, like Lamarck's theories of inheritance, can provide inspiration for better search software. Lamarck thought that organisms can inherit acquired characteristics. Later understanding of the mechanisms of biological inheritance showed him to be wrong but this is no barrier to using his ideas in simulated evolution.

Here are some other biologically inspired methods for improving genetic algorithms. In *niche specialization* you must have a metric on the space of possible data structures. The fitness of each creature is divided or reduced by some increasing, positive function of the number of creatures close by. The effect of this is to make a partial solution worth less for fitness if many creatures have found it. When doing optimization this prevents the population from accumulating in local optima. The biological inspiration is the niches that various creatures fill in ecologies such as large herbivore, nectar eater/pollinator, or large carnivore. The energy balance in nature that forces creatures in nature into niches is mimicked by the positive increasing function used to scale fitness in a fashion that prevents accumulation in local optima. This technique was developed by David Goldberg [2]. In [4] the author improves evolution by transforming it into coevolution. The population of putative solutions have their fitness evaluated by presenting them with a set of test cases. The test cases are themselves treated as an evolving population and their fitness is proportional to the number of solutions that they fool. This means that the fitness function is itself defined in terms of an evolving population, similar to the biological interplay between predator and prey or parasite and host.

The central problem in the theory of genetic algorithms is the characterization of the set of problems that are tractable for genetic algorithms. There are few theorems and a great deal of anecdotal evidence about the domains where genetic algorithms may or may not be useful. Because of this, I advise against forming a strong positive or negative impression of genetic algorithms from your experience trying to solve any single problem with genetic techniques. The author welcomes e-mail and conversations about discoveries in this area, even if they are anecdotal.

REFERENCES

[1] Lawrence Davis, *Handbook of genetic algorithms.* Van Nostrand Reinhold, New York, 1991.

[2] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[3] Stephanie Forrest, Genetic Algorithms: Principles of Natural Selection Applied to Computation , *Science* 261 (1993), 872-878.

[4] W. Daniel Hillis, Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure, in *Artificial Life II* (Santa Fe Institute Studies in the Sciences of Complexity 10), ed. Christopher Langton, Addison-Wesley, Reading, 1991, 313–324

[5] John H Holland, *Adaption in Natural and Artificial Systems.* The MIT Press, Cambridge, MA, 1992.

[6] G. F. Royle and W. D. Wallis, Constructing Bridge Club Designs , *Bulletin of the Institute for Combinatorics and its Applications* 11 (1994), 122–125.

5

MAKING THE MOLS TABLE

Charles J. Colbourn
*** Jeffrey H. Dinitz**

Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Ontario, CANADA N2L3G1

** Department of Mathematics and Statistics*
University of Vermont
Burlington, Vermont 05405 U.S.A.

ABSTRACT

This paper is not to be read by the faint of heart. No proofs are given, but it contains statements of a truly alarming number of constructions for transversal designs and incomplete transversal designs.

The paper is a record of an attempt to construct tables of the best results implied by known constructions for the existence of certain classes of mutually orthogonal latin squares and incomplete latin squares.

Sections §1–6 establish the mathematical background for the paper. We begin with basic definitions in §1. Then the following five sections state a fairly complete collection of construction techniques. It may well be impossible to write a complete list of variants of known constructions, and it is certainly beyond reason to do so. We content ourselves with a large battery of the constructions that have been exploited in the literature.

In §7–9, we describe a package developed in MAPLE which instantiates most (but not all) of the constructions in code. Issues in the design of this package are addressed, and a discussion of the architecture of the package is given.

1 BASIC DEFINITIONS

First, we must establish the mathematical framework in which we work.

W. D. Wallis (ed.), Computational and Constructive Design Theory, 67–134.
© 1996 *Kluwer Academic Publishers.*

A *transversal design* of *order* or *groupsize* n, *blocksize* k and *index* λ, denoted $TD_\lambda(k,n)$, is a triple $(V,\mathcal{G},\mathcal{B})$, where

1. V is a set of kn elements;
2. $\mathcal{G}$ is a partition of V into k classes (called *groups*), each of size n;
3. $\mathcal{B}$ is a collection of k–subsets of V (called *blocks*);
4. every unordered pair of elements from V is either contained in exactly one group, or is contained in exactly λ blocks, but not both.

When $\lambda = 1$, one writes simply $TD(k,n)$.

A $TD(k,n)$ is equivalent to the existence of $k-2$ mutually orthogonal latin squares of order n, and the various generalizations of transversal designs all have reasonably natural interpretations in that formulation (and also in a third disguise, as "orthogonal arrays"). We have chosen to remain with one notation as much as possible, and have chosen here to use the language of transversal designs. (But see [114] and [115] to convert to the other notations, if desired.)

An *incomplete transversal design* of *order* or *groupsize* n, *blocksize* k, *index* λ, and *holesizes* $b_1,\ldots,b_s$, denoted $ITD_\lambda(k,n;b_1,\ldots,b_s)$ for short, is a quadruple $(V,\mathcal{G},\mathcal{H},\mathcal{B})$, where

1. V is a set of kn elements;
2. $\mathcal{G}$ is a partition of V into k classes (*groups*), each of size n;
3. $\mathcal{H}$ is a set of disjoint subsets $H_1,\ldots,H_s$ of V, with the property that, for each $1 \le i \le s$ and each $G \in \mathcal{G}$, $|G \cap H_i| = b_i$;
4. $\mathcal{B}$ is a collection of k–subsets of V (*blocks*);
5. every unordered pair of elements from V is
   - contained in a hole, and contained in no blocks; or
   - contained in a group, and contained in no blocks; or
   - contained in neither a hole nor a group, and contained in λ blocks.

When $\sum_{i=1}^{s} b_i = n$, an $ITD(k,n;b_1,\ldots,b_n)$ is a *partitioned ITD*, here denoted by $PITD(k,n;b_1,\ldots,b_s)$. We often write the list $b_1,\ldots,b_s$ in "exponential

notation", so that $x_1^{y_1} \cdots x_s^{y_s}$ signifies that there are y_i holes of order x_i, for each $1 \leq i \leq s$.

Again, when $\lambda = 1$, it can be omitted from the notation. Another notation that we employ for an $ITD_\lambda(k, n; b_1, \ldots, b_s)$ is $TD_\lambda(k, n) - \sum_{i=1}^{s} TD_\lambda(k, b_i)$. It is trivial that the hole sizes could in fact be written in any order, and so when one refers to a specific hole size such as b_1, one is really speaking of an arbitrary hole size.

2 FILLING, TRUNCATION AND INFLATION

First we introduce the easiest of the constructions, treating some basic equivalences.

Lemma 2.1 *A $TD(k, n)$ is equivalent to an $ITD(k, n; b_1, \ldots, b_s)$ for any nonnegative integers $b_1, \ldots, b_s$ with $\sum_{i=1}^{s} b_i \leq 1$.*

Actually, holes of order 0 can be assumed present or absent to suit our purposes. Holes of size 1, on the other hand, can always be assumed absent if we choose, because the hole can always be replaced by a block; one cannot, however, assume them to be present unless there is a suitable block available for deletion to form the hole. It is convenient to make a simple convention which avoids treating holes of size 0 and 1 as special cases in each result, namely:

Convention 2.2 *For all nonnegative integers k, there exists a $TD(k, 0)$ and a $TD(k, 1)$.*

More than one hole of size one can occasionally be assumed:

Lemma 2.3 *[25] A $ITD(k, n; h)$ is equivalent to a $ITD(k, n; h, 1, 1)$ when $(k-1)h < n$. In particular, a $TD(k, n)$ is equivalent to a $TD(k, n) - 3TD(k, 1)$ when $k \leq n$.*

At this point, it is sensible to remark on the basic necessary condition:

Lemma 2.4 *An $ITD(k,n;h)$ exists only if $h = n$ or $(k-1)h \leq n$. When $(k-1)h = n$, all blocks have exactly one point in the hole.*

The case when $h = n$ corresponds to an incomplete transversal design which has no blocks at all, just one big hole. Nevertheless, to be explicit, we state the following:

Convention 2.5 *For n a positive integer and k a nonnegative integer, there exists a $TD(k,n) - TD(k,n)$.*

When $(k-1)h+1 = n$, simple counting shows that each element not in the hole lies on exactly one block that does not meet the hole. Deleting these blocks, we obtain:

Lemma 2.6 *An $ITD(k,(k-1)h+1;h)$ is equivalent to a $PITD(k,(k-1)h+1;h^1 1^{(k-2)h+1})$.*

Lemma 2.3 has a number of generalizations. One can remark, for example, that a simple greedy strategy always produces $1 + \lfloor \frac{n-1}{k-1} \rfloor$ disjoint blocks in a $TD(k,n)$, which can improve upon Lemma 2.3 when k is "small" relative to n. In addition, we can examine what happens when there are two or more holes assumed:

Lemma 2.7 *An $ITD(k,n;b_1,\ldots,b_s)$ always satisfies $(k-1)b_1 + b_2 \leq n$ (in particular, this holds when b_1 and b_2 are orders of the largest and second largest hole, respectively). Moreover, the ITD always has a block missing the first two holes, unless $(k-1)b_1 + b_2 = n$, $b_2 = b_3 = \ldots = b_s$ and $n = \sum_{i=1}^{s} b_i$. Consequently, an $ITD(k,n;b_1,b_2)$ with $b_1 \geq b_2 > 0$ is equivalent to an $ITD(k,n;b_1,b_2,1)$.*

We collect some other easy constructions in three main categories.

2.1 Filling

The basic result for filling an incomplete transversal design is:

Lemma 2.8 *If an* $ITD(k, n; b_1, \ldots, b_s)$ *exists, and an* $ITD(k, b_1; a_1, \ldots, a_r)$ *exists, then an* $ITD(k, n; a_1, \ldots, a_r, b_2, \ldots, b_s)$ *exists.*

For partitioned ITD, one can fill in a more general way:

Lemma 2.9 *Suppose there is a* $PITD(k, n; b_1, \ldots, b_s)$. *Let* ε *be a nonnegative integer, and suppose that, for each* $2 \leq i \leq s$, *there is a*

$$TD(k, b_i + \varepsilon) - TD(k, \varepsilon).$$

Then there exists a $TD(k, n + \varepsilon) - TD(k, b_1 + \varepsilon)$.

2.2 Truncation

Truncation is the operation of removing some points from a group. Here we examine the simplest form of truncation, when all points in a single group are deleted.

Lemma 2.10 *If a* $TD(k + 1, n)$ *exists, then a* $TD(k, n) - nTD(k, 1)$ *exists.*

Lemma 2.11 *If a* $TD(k + 1, n; h)$ *exists, then a* $TD(k, n) - TD(k, h) - (n - h)TD(k, 1)$ *exists.*

Removing a level of an ITD also has quite a useful consequence, which has been little exploited previously:

Lemma 2.12 *Suppose that an* $ITD(k+1, n; b_1, \ldots, b_s)$ *exists and* $\sum_{i=1}^{s} b_i < n$. *Suppose further that, for* $1 \leq i \leq s$, *there exists* $TD(k, b_i)$. *Then there exists a* $TD(k, n) - nTD(k, 1)$.

If instead there exists $TD(k, b_i)$ *for* $2 \leq i \leq s$, *then there exists a*

$$TD(k, n) - TD(k, b_1) - (n - b_1)TD(k, 1).$$

A somewhat similar operation can be done with partitioned ITD:

Lemma 2.13 *Suppose that a* $PITD(k+1, n; b_1, \ldots, b_s)$ *exists, with* $b_1 > 0$. *Suppose further that, for* $2 \leq i \leq s$, *there exists* $TD(k, b_i)$. *Then there exists a* $TD(k, n) - TD(k, b_1) - (n - b_1)TD(k, 1)$.

2.3 Inflation

The main form of inflation is a simple direct product:

Lemma 2.14 *Suppose that an* $ITD(k, n; b_1, \ldots, b_s)$ *and a* $TD(k, w)$ *both exist. Then a* $ITD(k, wn; wb_1, \ldots, wb_s)$ *exists.*

It bears frequent repetition that filling followed by inflation is *weaker* than inflation followed by filling. To see that it is no stronger, it suffices that each ingredient can be inflated by the same factor and the filling operation remains possible. That it is on occasion weaker follows from the fact that the ITD on wn points may exist, even when the ITD on n points does not exist.

3 PBD, GDD AND THE BOSE–SHRIKHANDE–PARKER THEOREM

A *pairwise balanced design* of *order* v and *blocksizes* K, denoted $(v, K) - PBD$, is a pair $(X, \mathcal{D})$. X is a set of v elements, and $\mathcal{D}$ is a set of subsets (*blocks*) of X for which $|D| \in K$ for each $D \in \mathcal{D}$. For every 2-subset of elements $\{x, y\} \subset X$, there is exactly one block containing x and y.

A *clear set* in a PBD is a set of pairwise disjoint blocks (also called a *partial parallel class*). A *near clear set* in a PBD $(X, \mathcal{D})$ is a subset $\mathcal{D}^c \subseteq \mathcal{D}$ defined as follows. For every block $D \in \mathcal{D}^c$, there is a distinguished element $e_D \in D$, the *tip* of D. The set $\mathcal{D}^c$ is near clear if, for each $x \in X$ contained in ι_x blocks of $\mathcal{D}^c$, x is the tip of at least $\iota_x - 1$ blocks in $\mathcal{D}^c$.

This definition seems unnaturally complicated, so perhaps some examples are needed. A set of blocks that all intersect in a single element, and are otherwise pairwise disjoint, is near clear: Simply taking the common element to be the tip of each block. A different example arises from three blocks which pairwise

intersect in one point, but the common intersection is empty. Choosing the three intersection points to be the tips of the three blocks shows that this structure is near clear.

Let us denote a $(v, K) - PBD$ $(X, \mathcal{D})$ with a near clear set $\mathcal{D}^c$ as a $(v, K_b, K_c) - PBD$, where K_c is the sizes of the blocks that actually arise in the near clear set, and K_b is the sizes of the blocks that actually arise among the remaining blocks (note that $K_b \cup K_c \subseteq K$, but equality is *not* necessary, as a $(v, K) - PBD$ need not in general realize every block size in K).

3.1 First Constructions Using PBD and GDD

Now to the basic (Bose–Shrikhande–Parker) construction:

Theorem 3.1 *Suppose that a $(v, K_b, K_c) - PBD$ exists. Suppose that, for every $m \in K_b$, there exists a $PITD(k, m; 1^m)$. Further suppose that, for every $m \in K_c$, there exists a $TD(k, m)$. Then there exists a $TD(k, v)$.*

Theorem 3.1 is a fairly standard Wilson-type construction using weight k for pairwise balanced designs [97]. The unusual feature is the use of near clear sets rather than clear sets. We content ourselves with remarking that for blocks in the near clear set, the ingredient used is actually a $TD(k, m) - TD(k, 1)$, and the $TD(k, 1)$ is chosen to coincide with the k copies of the tip element. In this vein, when ingredients exist with more than one hole of size 1, one could permit the blocks of the near clear set to have more than one tip; this would extend the definition of near clear set. However, we know of no applications of this generalization, so we omit it.

When we have a certain types of near clear sets (clear sets being one example), we can say something about incomplete TD as well:

Theorem 3.2 *Let $(X, \mathcal{D})$ be a PBD of order v. Suppose that for some subset $\mathcal{D}^c \subseteq \mathcal{D}$ of blocks, we have that there is one element $x \in X$, so that for $D, D' \in \mathcal{D}^c$, $D \cap D' \subseteq \{x\}$. Suppose that for every $D \in \mathcal{D} \setminus \mathcal{D}^c$, there exists a $PITD(k, |D|; 1^{|D|})$. Fix a block $F \in \mathcal{D}^c$, and suppose that for every block $D \in \mathcal{D}^c \setminus \{F\}$, there exists a $TD(k, |D|)$.*

Then there exists a $TD(k, v) - TD(k, |F|)$.

Next we constrain the near clear set to be a clear set to obtain:

Theorem 3.3 *Let $(X, \mathcal{D})$ be a PBD of order v. Suppose that some subset $\mathcal{D}^c \subseteq \mathcal{D}$ of blocks is a clear set. Suppose that for every $D \in \mathcal{D} \setminus \mathcal{D}^c$, there exists a $PITD(k, |D|; 1^{|D|})$. Then there exists a $TD(k, v) - \sum_{F \in \mathcal{D}^c} TD(k, |F|)$.*

In fact, letting $f = \sum_{F \in \mathcal{D}^c} |F|$, we obtain the stronger conclusion that there exists the partitioned ITD

$$TD(k, v) - (v - f)TD(k, 1) - \sum_{F \in \mathcal{D}^c} TD(k, |F|).$$

Theorem 3.4 *Let $(X, \mathcal{D})$ be a PBD of order v. Suppose that some subset $\mathcal{D}^c \subseteq \mathcal{D}$ of blocks is a clear set. Let $F \in \mathcal{D} \setminus \mathcal{D}^c$. Suppose that for every $D \in \mathcal{D} \setminus \mathcal{D}^c$, there exists a $PITD(k, |D|; 1^{|D|})$. Further suppose that for every $D \in \mathcal{D} \setminus (\mathcal{D}^c \cup \{F\})$, there exists a $TD(k, |D|)$. Then there exists a $TD(k, v) - TD(k, |F|)$.*

When the clear set is spanning (i.e., the union of the blocks is the set X of all elements in the PBD, or it is a *parallel class*), more flexibility exists. We introduce the appropriate language. A *group-divisible design* of *order* v, *blocksizes* K, and *type* $T = (t_1, \ldots, t_g)$, denoted $(v, K) - GDD$ of type T, is a triple $(X, \mathcal{C}, \mathcal{D})$, where X is a set of v elements, $\mathcal{C}$ is a partition of X into g classes (*groups*) $G_1, \ldots, G_g$, where $|G_i| = t_i$; and $\mathcal{D}$ is a set of subsets (*blocks*) of X, with the property that when $D \in \mathcal{D}$, we find $|D| \in K$. Moreover, every pair of elements appears together exactly once, either in a group or in a block. Often the type is written in exponential notation.

Now a $(v, K) - PBD$ is equivalent to a $(v, K) - GDD$ of type 1^v. A $TD(k, n)$ is equivalent to a $(kn, \{k\}) - GDD$ of type n^k. Also, converting the groups of a $(v, K) - GDD$ of type T into blocks, we obtain a $(v, K \cup T) - PBD$ in which the images of the groups form a parallel class of blocks. Then restating Theorem 3.3 when the clear set is a parallel class is equivalent to:

Theorem 3.5 *Suppose that there is a $(v, K) - GDD$ of type $T = (t_1, \ldots, t_s)$. Suppose that, for each $m \in K$, there is a $PITD(k, m; 1^m)$. Then there exists a $PITD(k, v; t_1, \ldots, t_s)$.*

Theorem 3.5 shows that one can employ the presence of a single parallel class. How can we use the presence of further parallel classes?

A PBD or GDD with element set X and block set $\mathcal{B}$ is *resolvable* if $\mathcal{B}$ can be partitioned into parallel classes. The partitioning into parallel classes is a *resolution.* Intermediate between GDD and resolvable PBD, we may have a PBD in which some, but not all, of the blocks are partitioned into parallel classes. Since resolvable PBD form a special case, we treat this more general situation.

Theorem 3.6 *Let $(X, \mathcal{D})$ be a PBD of order v. Suppose that $\mathcal{D}$ is partitioned into $r+1$ classes $\mathcal{D}_1, \ldots, \mathcal{D}_{r+1}$, where $\mathcal{D}_i$ is a parallel class for $1 \leq i \leq r$, and $\mathcal{D}_{r+1}$ is arbitrary (possibly even empty). Suppose that, for each $D \in \mathcal{D}_{r+1}$, there is a $PITD(k, |D|; 1^{|D|})$.*

Now for $2 \leq i \leq r$, let ε_i be a nonnegative integer. Suppose that, for each $2 \leq i \leq r$, and each $D \in \mathcal{D}_i$, there exists a

$$TD(k, |D| + \varepsilon_i) - TD(k, \varepsilon_i) - |D|\ TD(k, 1).$$

Let $\sigma = \sum_{i=2}^{r} \varepsilon_i$. Then there exists a

$$TD(k, v + \sigma) - TD(k, \sigma) - \sum_{D \in \mathcal{D}_1} TD(k, |D|),$$

a partitioned ITD.

Theorem 3.6 applies equally well to resolvable GDD, or GDD with parallel classes; simply treat the groups as blocks forming a parallel class of an equivalent PBD.

3.2 Incomplete PBD

Group-divisible designs are pairwise balanced designs with a spanning set of holes (the groups). Here we treat pairwise balanced designs with one hole. An *incomplete PBD* of *order* n, *blocksizes* K, and a *hole* of order h ($(v, h, K) - IPBD$) is a triple $(V, H, \mathcal{B})$. $|V| = v$, $|H| = h$, and $H \subset V$. $\mathcal{B}$ is a set of subsets of V, for which $B \in \mathcal{B}$ implies $|B| \in K$. Moreover, $(V, \mathcal{B} \cup \{H\}$ is a $(v, K \cup \{h\}) - PBD$. Since any single block can be taken to form a clear set, we obtain from Theorem 3.3:

Corollary 3.7 *Suppose there exists a $(v, h, K) - IPBD$. Suppose that for each $m \in K$, there exists a $PITD(k, m; 1^m)$. Then there exists an $ITD(k, v; h)$, and in fact there exists a $PITD(k, v; h^1 1^{v-h})$.*

In a $(v,h,K)-IPBD$ $(V,H,\mathcal{B})$, a *holey parallel class* is a set $\mathcal{P}$ of disjoint blocks, none of which meet the hole, and for which $V = H \cup \bigcup_{P\in\mathcal{P}} P$. One simple way to produce $IPBD$ with a holey parallel class is the following:

Lemma 3.8 *If there exists a $(v,K)-GDD$ of type $T=(t_1,t_2,\ldots,t_s)$, then there exists a $(v,t_1,K\cup\{t_2,\ldots,t_s\})-IPBD$ with a holey parallel class with block sizes in $\{t_2,\ldots,t_s\}$.*

Later we see other ways to produce $IPBD$ that have many holey parallel classes, so here we examine a method to use their presence:

Theorem 3.9 *Let $(V,H,\mathcal{B})$ be an $(v,h,K)-IPBD$. Partition the blocks $\mathcal{B}$ into classes $\mathcal{P}_1,\ldots,\mathcal{P}_r,\mathcal{Q}$ so that, for $1\le i\le r$, $\mathcal{P}_i$ is a holey parallel class. For $2\le i\le r$, let ε_i be a nonnegative integer. Now suppose that, for each $B\in\mathcal{Q}$, there exists a $PITD(k,|B|;1^{|B|})$. Further suppose that, for each $2\le i\le r$, and each $B\in\mathcal{P}_i$, there exists a $PITD(k,|B|+\varepsilon_i;\varepsilon_i^1 1^{|B|})$. Let $\sigma=\sum_{i=2}^r\varepsilon_i$. Then there exists a partitioned ITD*

$$TD(k,v+\sigma)-TD(k,\sigma+h)-\sum_{B\in\mathcal{P}_1}TD(k,|B|).$$

Actually, we could take $h=0$; then the $IPBD$ would be a PBD and the holey parallel classes would be parallel classes. Theorem 3.9 would then reduce to Theorem 3.6.

A $(v,h,K)-IPBD$ can have both parallel classes *and* holey parallel classes. If such an event occurs, we proceed as follows:

Theorem 3.10 *Let $(V,H,\mathcal{B})$ be an $IPBD$ with $|V|=v$ and $|H|=h$. Suppose that $\mathcal{B}$ has a partition into classes $\{\mathcal{P}_1,\ldots,\mathcal{P}_r,\mathcal{Q}_1,\ldots,\mathcal{Q}_s,\mathcal{R}\}$, where the $\{\mathcal{P}_i\}$ are parallel classes, the $\{\mathcal{Q}_i\}$ are holey parallel classes, and $\mathcal{R}$ is the remaining set of blocks (possibly empty). Suppose that $s\ge 1$. Suppose that, for every $B\in\mathcal{R}$, there exists a $PITD(k,|B|;1^{|B|})$.*

Choose nonnegative integers ε_i for $1\le i\le r$, and suppose that, for every $B\in\mathcal{P}_i$, there exists a $PITD(k,|B|+\varepsilon_i;\varepsilon_i^1 1^{|B|})$. Let $\sigma=\sum_{i=1}^r\varepsilon_i$.

Choose nonnegative integers γ_i for $2\le i\le s$, and suppose that, for every $B\in\mathcal{Q}_i$, there exists a $PITD(k,|B|+\gamma_i;\gamma_i^1 1^{|B|})$. Let $\sigma'=\sum_{i=2}^s\gamma_i$.

Then two outcomes are possible:

1. *If there exists a* $TD(k, \sigma + \sigma') - TD(k, \sigma')$, *then there exists a*

$$TD(k, v + \sigma + \sigma') - TD(k, h + \sigma') - \sum_{B \in \mathcal{Q}_1} TD(k, |B|).$$

2. *If there exists a* $TD(k, h + \sigma') - TD(k, \sigma')$, *then there exists a*

$$TD(k, v + \sigma + \sigma') - TD(k, \sigma + \sigma') - \sum_{B \in \mathcal{Q}_1} TD(k, |B|).$$

Some variants are possible, as prior to choosing the two outcomes, we find that two holes, one of size $h+\sigma'$ and the other of size $\sigma+\sigma'$, intersect in σ' elements. The last ingredients used to "break the tie" could themselves have holes, which would lead to even more holes in the final result. We do not pursue this.

However, it is necessary to explore what happens when we save back a parallel class instead of a holey parallel class. That leads to the next result:

Theorem 3.11 *Let* $(V, H, \mathcal{B})$ *be an* $IPBD$ *with* $|V| = v$ *and* $|H| = h$. *Suppose that* $\mathcal{B}$ *has a partition into classes* $\{\mathcal{P}_1, \ldots, \mathcal{P}_r, \mathcal{Q}_1, \ldots, \mathcal{Q}_s, \mathcal{R}\}$, *where the* $\{\mathcal{P}_i\}$ *are parallel classes, the* $\{\mathcal{Q}_i\}$ *are holey parallel classes, and* $\mathcal{R}$ *is the remaining set of blocks (possibly empty). Suppose that* $r \geq 1$. *Suppose that, for every* $B \in \mathcal{R}$, *there exists a* $PITD(k, |B|; 1^{|B|})$.

Choose nonnegative integers ε_i *for* $2 \leq i \leq r$, *and suppose that, for every* $B \in \mathcal{P}_i$, *there exists a* $PITD(k, |B| + \varepsilon_i; \varepsilon_i^1 1^{|B|})$. *Let* $\sigma = \sum_{i=2}^{r} \varepsilon_i$.

Choose nonnegative integers γ_i *for* $1 \leq i \leq s$, *and suppose that, for every* $B \in \mathcal{Q}_i$, *there exists a* $PITD(k, |B| + \gamma_i; \gamma_i^1 1^{|B|})$. *Let* $\sigma' = \sum_{i=1}^{s} \gamma_i$.

Suppose that a $PITD(k, h + \sigma'; (\sigma')^1 1^h)$ *exists.*

Then there exists a partitioned ITD,

$$TD(k, v + \sigma + \sigma') - TD(k, \sigma + \sigma') - \sum_{B \in \mathcal{P}_1} TD(k, |B|).$$

Again some variations are possible that we do not discuss.

One way to construct suitable $IPBD$ for Theorems 3.10 and 3.11 is to use the following result:

Lemma 3.12 *Let $(V, \mathcal{B})$ be a resolvable PBD with resolution $\{\mathcal{P}_1, \ldots, \mathcal{P}_r\}$. Choose $B \in \mathcal{P}_1$. Then $(V, B, \mathcal{B} \setminus \{B\})$ is an $IPBD$ whose blocks are partitioned into one holey parallel class $\mathcal{P}_1 \setminus \{B\}$, and $r-1$ parallel classes $\mathcal{P}_2, \ldots, \mathcal{P}_s$.*

One can go further, and consider structures in which there are many holes, and holey parallel classes associated with each. In this direction, one might consider "frames", for example. However, we do not explore this extension.

3.3 Making PBD and GDD

Making pairwise balanced designs and group-divisible designs is an industry in itself. Since Wilson's pioneering work on the asymptotic existence of designs (see [97]), constructions of PBD and GDD have flourished. Indeed one of the main reasons to construct incomplete transversal designs is to use them in constructing various other classes of designs.

We make no effort in this paper to describe all of the available constructions for PBD and GDD. Instead, we describe here some constructions from ITD; in §4 we see a number of other constructions from other classes of designs.

Let us start with easy things. A $TD(k, n)$ is itself a $(nk, \{k, n\}) - PBD$; in fact, it is a $(nk, \{k\}) - GDD$ of type n^k.

Deleting any set of elements from a PBD produces another PBD, in which each deleted element is simply omitted from each block in which it occurred (blocks of size 0, 1, or 2 may result; blocks of size 0 or 1 can be omitted if we choose). Thus every PBD gives an enormous variety of smaller PBD by this *puncturing* process. However, it should be clear that puncturing a TD randomly typically leads to a PBD with many block sizes. Since we are interested in being able to apply the theorems given earlier, we are concerned primarily with the cases when puncturing leads to relatively few block sizes.

We describe some concrete instances here. The first is obtained by puncturing points from ℓ of the groups.

Lemma 3.13 *Suppose that a $TD(k+\ell, n)$ exists. Choose integers $b_1, \ldots, b_\ell$ so that $0 \leq b_i \leq n$ for $1 \leq i \leq \ell$. Then there exists a $(kn + \sum_{i=1}^{\ell} b_i, \{k, k+1, \ldots, k+\ell\}) - GDD$ of type $n^k b_1^1 b_2^1 \cdots b_\ell^1$.*

Of course, in Lemma 3.13, blocks of sizes $\{k, k+1, \ldots, k+\ell\}$ are all *possible*. But whether a block of a particular size arises depends on the structure of the TD and the actual points deleted. Nevertheless, we can apply Theorem 3.5 to this GDD. Knowing the actual block sizes could result in a stronger application of that theorem. We return to this point in §5.7.

Truncating a single group can yield useful $IPBD$:

Lemma 3.14 *Suppose that a $TD(k+1, n)$ exists. Let $0 \leq \rho \leq n-1$. Then a $(kn+\rho, \rho, \{k, k+1, n\}) - IPBD$ exists having one holey parallel class of type n^k, $n-\rho$ parallel classes of type k^n, and the remaining ρn blocks of size k.*

An extreme case of Lemma 3.14 is when the whole group is deleted. This is equivalent to the following well–known result:

Lemma 3.15 *A resolvable $TD(k, n)$ is equivalent to a $TD(k+1, n)$.*

Puncturing partitioned ITD leads to GDD with groups arising from the holes:

Lemma 3.16 *Suppose that there exists a $PITD(k+1, n; 1^n)$. Let $0 \leq \alpha \leq n$. Then there exists a $(nk+\alpha, \alpha, \{n, k+1, k\}) - IPBD$ with a holey parallel class of type n^k, a parallel class of type $(k+1)^\alpha k^{n-\alpha}$, and all other blocks of sizes k (whenever $\alpha < n$) and $k+1$ (whenever $\alpha > 0$).*

Puncturing one group of an incomplete TD with one hole leads to:

Lemma 3.17 *Suppose that there exists an $ITD(k+1, n; h)$. Let $0 \leq \alpha < h$. Suppose that there exists a $PITD(k, k+1; 1^{k+1})$ and a $PITD(k, k; 1^k)$. Then there exists a $PITD(k, nk+\alpha, (kh+\alpha)^1 k^{n-h})$.*

Actually, more can be said since $h - \alpha$ holey parallel classes of blocks of size k missing the hole of size $kh + \alpha$ are present, and we have used only a single one here.

We see more sophisticated ways to puncture a TD in §5.7; we give one of the simpler cases here:

Lemma 3.18 *Suppose that a $TD(k+\ell, n)$ exists with $\ell \geq 2$. Let $1 \leq \alpha \leq n$. Then a $(nk+\alpha+\ell-1, \{k, k+1, k+2, k+\ell\}) - GDD$ of type $n^k \alpha^1 1^{\ell-1}$ exists.*

Using resolvable TD, we also obtain:

Lemma 3.19 *Suppose that a $TD(k+\ell+1, n)$. Then there is a $(nk+\ell, k+\ell, \{k, k+1, n\}) - IPBD$ having a holey parallel class of type k^{n-1}, a parallel class of type $n^k 1^\ell$, and $n-1$ parallel classes of type $(k+1)^\ell k^{n-\ell}$.*

Another useful puncture is to delete points from a block, rather than from a group:

Lemma 3.20 *Suppose that a $TD(k, n)$ exists. Let ρ be an integer satisfying $0 \leq \rho \leq k$. Then there exists a $(k(n-1)+\rho, \rho, \{k, k-1, n, n-1\}) - IPBD$ and a $(k(n-1)+\rho, \{k, k-1, \rho\}) - GDD$ of type $n^\rho (n-1)^{k-\rho}$.*

In the $IPBD$, blocks of size n appear only if $\rho > 0$ and blocks of size $n-1$ appear only if $\rho < k$. In both the $IPBD$ and the GDD, blocks of size $k-1$ appear if and only if $\rho < k$; blocks of size k appear if and only if $\rho > 0$ or $k \leq n$.

Deleting a whole block gives, on two occasions, PBD that ought to be noted.

Lemma 3.21 *If a $TD(n+1, n)$ exists, then there is a $(n^2-1, \{n\})$-GDD of type $(n-1)^{n+1}$. (In fact, they are equivalent.) Further deleting all elements in one group, we obtain a resolvable $(n(n-1), \{n-1\}) - GDD$ of type n^{n-1}.*

When a block is deleted from a resolvable TD, information about parallel classes can be retained:

Lemma 3.22 *Suppose that a $TD(k+1, n)$ exists. Let ρ be an integer satisfying $0 \leq \rho \leq k$. Then there exists a $(k(n-1)+\rho, \rho, \{k, k-1, n, n-1\}) - IPBD$ having one holey parallel class of type k^{n-1}, one parallel class of type $(n-1)^\rho n^{k-\rho}$, and $n-1$ parallel classes of type $k^{n-k+\rho}(k-1)^{k-\rho}$.*

3.4 The Bose–Shrikhande–Parker Theorem

The (general form of the) Bose-Shrikhande-Parker theorem [18, 25] exploits additional structure occurring in some PBD. We generalize the notion of parallel class. An *α-parallel class* in a PBD $(V, \mathcal{B})$ is a set $\mathcal{C} \subseteq \mathcal{B}$ of blocks, with the property that every $x \in V$ appears in *exactly* α blocks of $\mathcal{C}$. Evidently, a 1–parallel class is just a parallel class.

An α-parallel class $\mathcal{C}$ is *symmetric* if every block in $\mathcal{C}$ has size α. It is easy to verify in this case that the number of blocks in $\mathcal{C}$ coincides with the number of elements in V — hence the term symmetric.

A *separable PBD* is one whose blocks can be partitioned into 1–parallel classes and symmetric parallel classes; within each class, all blocks have the same size.

Now we can state the Bose–Shrikhande–Parker theorem:

Theorem 3.23 *Let $(V, \mathcal{B})$ be a $(v, K) - PBD$, and suppose that $\mathcal{B}$ can be partitioned into classes $\mathcal{P}_1, \ldots, \mathcal{P}_r, \mathcal{S}_1, \ldots, \mathcal{S}_s$. For $1 \leq i \leq r$, $\mathcal{P}_i$ is a parallel class. For $1 \leq i \leq s$, $\mathcal{S}_i$ is a symmetric α_i-parallel class. Now let $\varepsilon_i \in \{0, 1\}$ for $1 \leq i \leq s$, and suppose that a $PITD(k, \alpha_i; 1^{\alpha_i})$ exists if $\varepsilon_i = 0$, and that a $PITD(k, \alpha_i + 1; 1^{\alpha_i + 1})$ exists if $\varepsilon_i = 1$. Let $\sigma = \sum_{i=1}^{s} \varepsilon_i \alpha_i$.*

Let γ_i be a positive integer, for $2 \leq i \leq r$. Suppose that, for $2 \leq i \leq r$ and for each $B \in \mathcal{P}_i$, there exists a $PITD(k, |B| + \gamma_i; \gamma_i^1 1^{|B|})$ exists. Let $\sigma' = \sum_{i=2}^{r} \gamma_i$.

Then, if $r \geq 1$, there exists the partitioned ITD

$$TD(k, v + \sigma + \sigma') - TD(k, \sigma + \sigma') - \sum_{B \in \mathcal{P}_1} TD(k, |B|).$$

If $r = 0$, we instead obtain the partitioned ITD

$$TD(k, v + \sigma) - TD(k, \sigma) - vTD(k, 1).$$

3.5 Making Separable PBD

The Bose–Shrikhande–Parker theorem is a generalization of Theorem 3.6 to separable PBD. However, it is difficult to find examples of separable PBD that are not resolvable. We see in §§4 that examples arise from cyclic block

designs and symmetric block designs. The only other general construction for separable PBD is due to Brouwer [22]:

Theorem 3.24 *Let q be a prime power, and let $0 < t \leq q^2 - q + 1$. Then there exists a separable $(t(q^2+q+1), \{t, q+t\}) - PBD$ which has a partitioning with $q^2-q+1-t$ parallel classes of blocks of size t, and one symmetric $(q+t)$-parallel class.*

4 STEINER SYSTEMS, SYMMETRIC DESIGNS AND DIFFERENCE SETS

A *Steiner system* of *order* v and *blocksize* κ, denoted $S(2, \kappa, v)$, is a $(v, \{\kappa\}) - PBD$. (This is actually a *Steiner 2-design*, but we only have occasion to use the case of $t = 2$ here; it is also a (balanced incomplete) block design, but we only treat the case when $\lambda = 1$. For these reasons, we have adopted the Steiner system notation here.)

An $S(2, \kappa, v)$ is *symmetric* when the number of blocks in the design, namely $\frac{v(v-1)}{\kappa(\kappa-1)}$ is equal to v (i.e., $v = \kappa(\kappa-1)+1$). A symmetric $S(2, \kappa, v)$ is equivalent to a *projective plane* of *order* $n = \kappa - 1$, with $v = n^2 + n + 1$ elements (and $n^2 + n + 1$ blocks or lines). In a projective plane, every two distinct blocks intersect in one element.

An $S(2, \kappa, v)$ is *cyclic* when there is an automorphism of the design that is a v–cycle.

First, the basics:

Lemma 4.1 *Removing one element from a projective plane of order n, and treating the resulting blocks of size n as groups, a $TD(n+1, n)$ is produced.*

We can also remove a whole block:

Lemma 4.2 *Removing one block from a projective plane of order n (or one group from a $TD(n+1, n)$, a resolvable $TD(n, n)$ (*affine plane of order n, *or $S(2, n, n^2)$) results.*

Moreover, every $TD(n,n)$ is resolvable and can be extended to a projective plane.

4.1 Arbitrary $S(2,\kappa,v)$

An $S(2,\kappa,v)$ is, of course, itself a PBD. However, sometimes truncating this special type of PBD can lead to extra information. We can truncate points from a single block:

Lemma 4.3 *If an $S(2,\kappa,v)$ exists, then for $0 \leq x \leq \kappa$, a $(v-x, \kappa-x, \{\kappa, \kappa-1\}) - IPBD$ exists having x holey parallel classes of type $(\kappa-1)^{(v-\kappa)/(\kappa-1)}$. It has blocks of size κ unless $x = \kappa$ and the $S(2,\kappa,v)$ is symmetric.*

Next we delete a small number of points, not all from the same block.

Lemma 4.4 *If an $S(2,\kappa,v)$ exists, then there exists a $(v-3, \{\kappa-2, \kappa-1, \kappa\}) - PBD$ in which there are exactly three blocks of size $\kappa-2$, and they form a near clear set.*

If an $S(2,\kappa,v)$ exists, then there exists a $(v-4, \{\kappa-2, \kappa-1, \kappa\}) - PBD$ in which there are exactly four blocks of size $\kappa-2$, and they form a near clear set.

We can naturally delete points all over if we so desire, but to obtain useful results we want to minimize the number of different block sizes. With this in mind, we give another definition. If A is a set of s points, no three of which lie on a block, then A is an *s-arc*.

Lemma 4.5 *If an $S(2,\kappa,v)$ exists having an s-arc, then for all $1 \leq x \leq s$, there exists a $(v-x, \{\kappa-2, \kappa-1, \kappa\}) - PBD$. Blocks of size $\kappa-2$ always occur when $x > 1$. Blocks of size $\kappa-1$ appear unless $v = 1 + (x-1)(\kappa-1)$. Blocks of size κ always appear.*

Examples of designs with useful arcs appear in §4.3.

Existence of block designs is a central problem in combinatorial design theory, and there is a huge literature. For existence results, see [117]. Much is known about resolvability of block designs, furnishing many examples of resolvable PBD.

4.2 Cyclic $S(2, \kappa, v)$

Cyclic $S(2, \kappa, v)$ have been studied extensively; see [113]. A cyclic $S(2, \kappa, v)$ can exist only if $v \equiv 1, \kappa \pmod{\kappa(\kappa-1)}$. When $v \equiv 1 \pmod{\kappa(\kappa-1)}$, all block orbits under the cyclic automorphism have length v (they are *full*). When $v \equiv \kappa \pmod{\kappa(\kappa-1)}$, one block orbit has length $\frac{v}{\kappa}$ (it is *short*), and the rest are full.

A full orbit of blocks can be easily checked to be a κ-parallel class. Hence every cyclic $S(2, \kappa, v)$ is separable, with $\lfloor \frac{v}{\kappa(\kappa-1)} \rfloor$ κ-parallel classes, and one parallel class if the short orbit is present, none otherwise.

Hence we can apply the Bose–Shrikhande–Parker theorem to cyclic $S(2, \kappa, v)$.

4.3 Symmetric Designs

First we remark on a basic filling result that does not follow from filling the corresponding PBD [56].

Lemma 4.6 *If a symmetric $S(2, \kappa, v)$ exists and a $TD(k, \kappa)$ exists, then a $TD(k, v)$ exists.*

Next observing that a symmetric design $S(2, \kappa, v)$ is itself a single $((v-1)/(\kappa-1))$-parallel class, we can apply the Bose–Shrikhande–Parker theorem to obtain:

Lemma 4.7 *If a symmetric $S(2, \kappa, v)$ exists and a $PITD(k, \kappa+1; 1^{\kappa+1})$ exists, then a $TD(k, v + \frac{v-1}{\kappa-1}) - TD(k, \frac{v-1}{\kappa-1})$ exists.*

See also [56].

Certain projective planes have large arcs:

Theorem 4.8 *The desarguesian projective plane of order q (a prime power) contains a $(q+1)$-arc (an* oval*) if q is odd, and contains a $(q+2)$-arc (a* hyperoval*) if q is even.*

4.4 Line–flips in Affine Planes

Suppose that a $TD(n,n)$ exists; this is an affine plane of order n. Now choose an integer x with $1 \leq x < n$, and choose one block B. Delete all points from x of the groups *except* those on block B. Next delete all points on block B in the remaining $n-x$ groups. The resulting PBD has blocks of five types:

1. a single block on x points, which is the truncation of B;
2. $n-1$ disjoint blocks each of size $n-x$, which are the truncations of the blocks disjoint from B in the affine plane;
3. $n-x$ disjoint blocks each of size $n-1$, which are the truncations of the remaining groups;
4. blocks of size $n-x-1$ that do not intersect the truncation of B (in the affine plane, they did intersect B);
5. blocks of size $n-x+1$ that do intersect the truncation of B.

Types (1) and (2) form a parallel class; so also do types (1) and (3). Thus adding a point at infinity to the blocks of type (1) and (2), called a *type A extension*, gives a $\{n-x-1, n-x+1\}$-GDD of type $(n-1)^{n-x}(x+1)^1$. On the other hand, adding a point at infinity to blocks of types (1) and (3) gives a $\{n-x-1, n, n-x+1\}$-GDD of type $(n-x)^{n-1}(x+1)^1$ (a *type B extension*). Greig [42] observes that either GDD can be extended with a further point at infinity to form a PBD on $(n-x)(n-1)+x+2$ with block sizes $\{n, n-x-1, n-x+1, x+2^\star\}$. The superscript $\star$ indicates that a block of size $x+2$ is present, and that all other blocks have sizes from $\{n, n-x-1, n-x+1\}$.

4.5 Difference Sets

Singer [79] showed that the desarguesian projective plane of order q (a prime power) has a representation as a cyclic difference set. This provides a mechanism for finding other configurations in desarguesian planes.

Let Γ be an additively written group of order v. A κ-subset D of Γ is a (v, κ, λ)-*difference set* of *order* $n = \kappa - \lambda$ if every nonzero element of Γ has exactly λ representations as a difference $d - d'$ of distinct elements from D. The difference set is *abelian* or *cyclic* if the group Γ has the corresponding property.

The development of a difference set D under the action of the group Γ is a symmetric design; when $\lambda = 1$, it is a projective plane of order n. Thus our earlier remarks apply to the symmetric design. But here we may obtain more information.

We consider a $(q^2+q+1, q+1, 1)$–difference set D over the cyclic group Z_{q^2+q+1}, using the usual representation over the integers modulo $q^2 + q + 1$. For any divisor d of q^2+q+1, denote by $D_{i,d}$ the elements of D that are congruent to i modulo d. For an arbitrary subset $\mathcal{I} \subset \{0, 1, \ldots, d-1\}$, let $D_{\mathcal{I},d} = \bigcup_{i \in \mathcal{I}} D_{i,d}$. Then we have the following result, first studied by Brouwer [22] and later extended by Greig [42]:

Theorem 4.9 *Let D be a $(q^2+q+1, q+1, 1)$–difference set over the integers modulo q^2+q+1. Let d be a divisor of q^2+q+1, and $\mathcal{I} \subseteq \{0, 1, \ldots, d-1\}$. Then the collection of blocks*

$$\{\{x+i\} : 0 \le i \le q^2+q,\ x+i \bmod d \in \mathcal{I}\}$$

is a pairwise balanced design on $|\mathcal{I}|\frac{q^2+q+1}{d}$ elements.

The relevance of Theorem 4.9 is that it produces a PBD having at most d different block sizes.

4.6 Configurations in Projective Planes

In §4.5, we saw that projective planes arising from difference sets can embed a pairwise balanced design that often has "few" block sizes. We are interested in this phenomenon for a number of reasons. It provides a way to construct pairwise balanced designs, of course. But what is more critical for us is that it tells us something about the structure of the $TD(n+1, n)$ that arises from the plane — and this information can be helpful in predicting the block sizes that result when we puncture the TD. There is a third reason as well, namely that when PBD live in a projective plane, we can use this to produce more PBD. We pursue this in §4.7, but for now we explore results on when PBD live in projective planes.

Ovals, Hyperovals and Denniston Arcs

Arcs (ovals and hyperovals) form one important class of pairwise balanced designs inhabiting projective planes (Theorem 4.8), although the PBD themselves

are quite trivial. However, associated with the exterior lines of a hyperoval are a number of important PBD contained in the plane [77]:

Theorem 4.10 *The desarguesian projective plane of order $q = 2^\alpha$ contains*

1. *a resolvable $(\binom{q}{2}, \{\frac{q}{2}\}) - PBD$;*
2. *a $(\binom{q+2}{2}, \{\frac{q}{2}+1, q+1\}) - PBD$; and*
3. *a resolvable $(\binom{q+1}{2}, \{\frac{q}{2}, q\}) - PBD$.*

Denniston arcs [34] provide a generalization of these:

Theorem 4.11 *The desarguesian projective plane of order $q = 2^\alpha$ contains, for every $1 \leq \beta < \alpha$,*

1. *a resolvable $(2^{\alpha+\beta} - 2^\alpha + 2^\beta, \{2^\beta\}) - PBD$;*
2. *a $(2^{2\alpha} + 2^{\alpha+1} - 2^{\alpha+\beta} - 2^\beta + 1, \{2^\alpha - 2^\beta + 1, 2^\alpha + 1\}) - PBD$; and*
3. *a resolvable $((2^\alpha - 2^\beta)(2^\alpha + 1), \{2^\alpha - 2^\beta, 2^\alpha\}) - PBD$.*

Greig [42] employs ovals in planes of odd order to prove:

Theorem 4.12 *If q is an odd prime power, the desarguesian plane of order q contains*

1. *a GDD on $\binom{q}{2}$ points with uniform group size $\frac{q-1}{2}$, and block sizes in $\{\frac{q-1}{2}, \frac{q+1}{2}\}$; and*
2. *a GDD on $\binom{q+1}{2}$ points with uniform group size $\frac{q+1}{2}$, and block sizes in $\{\frac{q+1}{2}, \frac{q+3}{2}\}$.*

Subplanes and Baer Subplanes

More complex examples are given by subplanes of a plane. Simple numerical arguments show that a projective plane of order q can have a projective subplane of order p only if $q \geq p^2$. In the positive direction, we have [4]:

Lemma 4.13 *The desarguesian projective plane of order p^α has a subplane of order p^β whenever $\beta \mid \alpha$.*

The extreme case when $\alpha = 2\beta$ is especially important. In this case, the subplane is a *Baer subplane*, and some elementary counting arguments provide us with useful information. Let $q = p^\beta$ and $q^2 = p^{2\beta}$. Let $(V, \mathcal{B})$ be the plane of order q^2, and $(X, \mathcal{D})$ be its Baer subplane of order q.

Lemma 4.14 *1. Every point $x \in X$ lies on $q+1$ lines of $\mathcal{B}$ that intersect X in $q+1$ points, and on $q^2 - q$ lines of $\mathcal{B}$ that contain only x from X.*

2. Every point of $V \setminus X$ lies on one line of $\mathcal{B}$ that intersects X in $q+1$ points, and lies on q^2 lines of $\mathcal{B}$ that intersect X in one point.

3. Hence, all lines of $\mathcal{B}$ intersect X in either 1 or $q+1$ points.

Removing the points in X from the plane yields a $(q^4 - q, \{q^2 - q, q^2\}) - PBD$. Considering any point $x \in X$, we find that the blocks containing X form a parallel class of this PBD, and hence we in fact obtain a $(q^4 - q, \{q^2 - q, q^2\}) - GDD$ of type $(q^2 - q)^{q+1}(q^2)^{q^2-q}$, $(Y, \mathcal{C})$.

Now consider a block of size $q^2 - q$ in $\mathcal{C}$. It cannot intersect any group of size $q^2 - q$ (lines meet at a single point in the projective plane of order q^2), so it must intersect all groups of size q^2. In fact, all blocks and groups of size $q^2 - q$ are disjoint, so we have [80]:

Lemma 4.15 *If a projective plane of order q^2 has a Baer subplane of order q, there exists a $(q^4 - q, \{q^2\}) - GDD$ of type $(q^2 - q)^{q^2+q+1}$.*

Now a block of size q^2 from $\mathcal{C}$ must intersect all groups of size q^2, and precisely q of the groups of size $q^2 - q$. Thus we can delete all but x of the groups of size q^2 to obtain:

Lemma 4.16 *If a projective plane of order q^2 has a Baer subplane of order q, then for all $0 \leq x \leq q^2 - q$, there exist*

1. *a $((q^2-q)(q+1)+xq^2, \{q+x, x\})-GDD$ of type $(q^2-q)^{q+1}(q^2)^x$; and*
2. *a $((q^2-q)(q+1)+xq^2, \{q+x, q^2\})-GDD$ of type $(q^2-q)^{q+1}x^{(q^2)}$.*

Baer subplanes can be exploited further yet; see, for example, [51] for the following:

Lemma 4.17 *The desarguesian projective plane of order q^2 can be partitioned into q^2-q+1 element-disjoint Baer subplanes (each on q^2+q+1 points).*

Considering any line of the plane, simple counting shows that it intersects one Baer subplane of this partition in $q+1$ points, and the remaining q^2-q subplanes in one point each. So retaining points of t of the subplanes in the partition, we obtain:

Lemma 4.18 *From the desarguesian projective plane of order q^2 (q a prime power), for each $1 \leq t \leq q^2-q$, we obtain a $(t(q^q+q+1), \{q+t, t\})-PBD$ in which the blocks of size t are partitioned into $q^2-q+1-t$ parallel classes.*

Other specific planes have subplanes of interest: the Hughes plane of order 9 has a subplane of order 2 [33], and the Hughes plane of order 25 contains subplanes of orders 2 and 3 [70]. A complete survey of subplanes is *not* attempted here.

Affine Subplanes

Now we examine other structures in planes. In the direction of affine planes residing in projective planes, Ostrom and Sherk [66] and Rigby [71] proved:

Theorem 4.19 *The desarguesian projective plane of order q (a prime power) contains an affine plane of order 3 (an $S(2,3,9)$) if and only if $q \equiv 0, 1 \pmod 3$.*

The notion of "containment" in Theorem 4.19 is that a subset of the points is selected, and the intersections of all lines with these points induce shorter

lines; then keeping all such truncated lines on two or more points gives the affine plane.

Subsquares

Often a subplane (projective or affine) is not present, but useful portions are. For example, considering the standard construction of the desarguesian plane, we find [30]:

Lemma 4.20 *In the desarguesian plane of order p^α, for each $0 \leq \beta \leq \alpha$, there is embedded a $(3p^\beta + 1, \{3, p^\beta + 1\}) - PBD$ having three blocks of size $p^\beta + 1$ meeting in a single point, and all other blocks of size three.*

Actually, a more convenient way to express this is to observe that, when we remove the point common to the three "long" blocks, we form a $TD(p^\alpha+1, p^\alpha)$. Truncating to the $TD(3, p^\alpha)$ on the three special groups, and interpreting this TD as a latin square, we are essentially noting in Lemma 4.20 that this latin square of size p^α has a subsquare of size p^β.

Using the structure of the finite field, one can extend this to obtain:

Lemma 4.21 *In the desarguesian plane of order p^α, for each $0 \leq \beta \leq \alpha$, there is embedded a $((p+1)p^\beta + 1, \{p+1, p^\beta + 1\}) - PBD$ having $p+1$ blocks of size $p^\beta + 1$ meeting in a single point, and all other blocks of size $p+1$.*

When $\beta = \alpha - 1$ in Lemma 4.21, all lines of the plane meet the sub-TD in $p^\beta + 1$, $p+1$, or 1 points. The latter tangent lines induce a design in the dual plane; this tangent design is a $\{p^\alpha, p^\alpha - p^{2\beta-\alpha}+1\}$-GDD of type $(p^\alpha - p^\beta)^{p^\alpha+p^\beta}(p^\alpha - p^{\alpha-\beta})^1$. Further extensions are studied in [29].

4.7 Line-flips in Projective Planes

When a PBD is embedded in a projective plane, we can exploit the structure of the enclosing plane to form other PBD. Simply taking all points of the plane *not* in the PBD, for example, gives:

Lemma 4.22 *If a $(v, K) - PBD$ is embedded in a projective plane of order n, and there is a one–to–one correspondence between blocks of the PBD and lines of the plane so that each block is extended to the corresponding line (which may require adding blocks of sizes 0 and 1 to the PBD), then there exists a $(n^2 + n + 1 - v, \overline{K}) - PBD$, where $\overline{K} = \{n + 1 - s : s \in K\}$.*

In fact, the number of blocks of size $n + 1 - s$ in the resulting PBD is the same as the number of blocks of size s in the original PBD.

One can also do a "line–flip", by choosing some block of the PBD, deleting the points on this block and instead adding the points on the line of the plane which extends this block, but not on the block itself [42]. One obtains the following:

Lemma 4.23 *Suppose that a $(v, K) - PBD$ is embedded in a projective plane of order n, and there is a one–to–one correspondence between blocks of the PBD and lines of the plane so that each block is extended to the corresponding line (which may require adding blocks of sizes 0 and 1 to the PBD). Suppose further that the embedded PBD has a block of size s. Let $\widehat{K} = \{s - 1, s + 1 : s \in K\}$. Then there exists a $(v + n + 1 - 2s, n + 1 - s, \widehat{K}) - IPBD$.*

Examples of PBD that inhabit projective planes are given in §4.6, and also arise from Theorem 4.9.

5 WILSON'S THEOREM

Wilson's theorem, and all of its variants discussed here, start with a transversal design (or incomplete transversal design) of order t of blocksize $k + \ell$. We refer to this design as the *master design.*

The master design is always taken to be $(V, \mathcal{G}, \mathcal{B})$, although it may have additional structure, or holes. Let $\mathcal{G} = \{G_1, \ldots, G_k, E_1, \ldots, E_\ell\}$. Let $E_i = \{x_{i1}, \ldots, x_{it}\}$. For each $x_{ij} \in \bigcup_{i=1}^{\ell} E_i$, let w_{ij} be a nonnegative integer, the *weight* of x_{ij}. For each block $B \in \mathcal{B}$, let $w_i^B = w_{ij}$ when $B \cap E_i = \{x_{ij}\}$.

5.1 Transversal Designs as Master Designs

First we give what has come to be accepted as the basic form of Wilson's theorem, although Wilson [96] gave it in the case that $w_{ij} \in \{0,1\}$ for all $1 \le i \le \ell$, $1 \le j \le t$, and $\lambda = \mu = 1$.

Theorem 5.1 *Suppose that a* $TD_\mu(k+\ell, t)$ *exists. Suppose that for each* $B \in \mathcal{B}$, *there exists*

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Then there exists a

$$TD_{\lambda\mu}(k, mt + \sum_{i=1}^{\ell}\sum_{j=1}^{t} w_{ij}) - \sum_{i=1}^{\ell} TD_{\lambda\mu}(k, \sum_{j=1}^{t} w_{ij})$$

Of course, if we can fill some or all of the holes, further incomplete transversal designs will result.

When $\mu = 1$, we can obtain different holes as well:

Theorem 5.2 *Suppose that a* $TD_1(k+\ell, t)$ *exists. Let* $F \in \mathcal{B}$. *Suppose that for each* $B \in \mathcal{B} \setminus \{F\}$, *there exists*

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Suppose that, for $1 \le i \le \ell$, *there exists*

$$TD_\lambda(k, \sum_{j=1}^{t} w_{ij}) - TD_\lambda(k, w_i^F).$$

Then there exists

$$TD_\lambda(k, mt + \sum_{i=1}^{\ell}\sum_{j=1}^{t} w_{ij}) - TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^F).$$

We now change the structure of some of the ingredients. Let $\mathcal{B}_1$ be the blocks B for which $x_{11} \in B \in \mathcal{B}$, and let $\mathcal{B}_2 = \mathcal{B} \setminus \mathcal{B}_1$.

Theorem 5.3 *Suppose that a* $TD_1(k+\ell, t)$ *exists. Suppose that for each* $B \in \mathcal{B}_2$, *there exists*

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Let $\alpha_1, \alpha_2, \ldots, \alpha_r$ *be positive integers with* $\sum_{i=1}^{r} \alpha_i \leq m$. *Suppose that for each* $B \in \mathcal{B}_1$, *there exists*

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{r} TD_\lambda(k, \alpha_i) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Then there exists

$$TD_\lambda(k, mt + \sum_{i=1}^{\ell}\sum_{j=1}^{t} w_{ij}) - \sum_{i=1}^{r} tTD_\lambda(m, \alpha_i) - \sum_{i=1}^{\ell} TD_\lambda(k, \sum_{j=1}^{t} w_{ij}).$$

Finally we change the structure of all of the ingredients:

Theorem 5.4 *Suppose that a* $TD_\mu(k+\ell, t)$ *exists. Let* $\alpha_1, \alpha_2, \ldots, \alpha_r$ *be positive integers with* $\sum_{i=1}^{r} \alpha_i \leq m$. *Suppose that for each* $B \in \mathcal{B}$, *there exists*

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{r} TD_\lambda(k, \alpha_i) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Then there exists

$$TD_{\lambda\mu}(k, mt + \sum_{i=1}^{\ell}\sum_{j=1}^{t} w_{ij}) - \sum_{i=1}^{r} TD_{\lambda\mu}(m, \alpha_i t) - \sum_{i=1}^{\ell} TD_{\lambda\mu}(k, \sum_{j=1}^{t} w_{ij}).$$

5.2 Incomplete Transversal Designs as Master Designs

In the preceding constructions, we saw how incomplete transversal designs can be used in conjunction with a master design that is a transversal design. Here we examine variants where the master design itself is incomplete.

Theorem 5.5 *Let $\beta_1, \ldots, \beta_u$ be positive integers with $\sum_{a=1}^{u} \beta_a \leq t$. Suppose that there exists a master design, a*

$$TD_\mu(k+\ell, t) - \sum_{a=1}^{u} TD_\mu(k+\ell, \beta_a).$$

For $1 \leq a \leq u$, let O_a be the points in the ITD that lie in the a^{th} hole of size β_a (so that O_a contains $\beta_a(k+\ell)$ elements in total). Let $z_{ia} = \sum_{x_{ij} \in (E_i \cap O_a)} w_{ij}$. Suppose that for each $B \in \mathcal{B}$, there exists

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Suppose further that, for $1 \leq a \leq u$, there exists

$$TD_{\lambda\mu}(k, m\beta_a + \sum_{i=1}^{\ell} z_{ia}) - \sum_{i=1}^{\ell} TD_{\lambda\mu}(k, z_{ia}).$$

Then there exists a

$$TD_{\lambda\mu}(k, mt + \sum_{i=1}^{\ell} \sum_{j=1}^{t} w_{ij}) - \sum_{i=1}^{\ell} TD_{\lambda\mu}(k, \sum_{j=1}^{t} w_{ij})$$

Actually, the holes that arise on the extra ℓ levels, and the holes that arise from the holes in the master design in the construction are not disjoint. In Theorem 5.5, we have elected to fill the latter and leave the former. In Theorem 5.6, we do the opposite. Since the theorems differ only in the last set of ingredients, they look cosmetically similar. Nevertheless, we state the conditions of the theorem in their entirety.

Theorem 5.6 *Let $\beta_1, \ldots, \beta_u$ be positive integers with $\sum_{a=1}^{u} \beta_a \leq t$. Suppose that there exists a master design, a*

$$TD_\mu(k+\ell, t) - \sum_{a=1}^{u} TD_\mu(k+\ell, \beta_a).$$

For $1 \leq a \leq u$, let O_a be the points in the ITD that lie in the a^{th} hole of size β_a (so that O_a contains $\beta_a(k+\ell)$ elements in total). Let $z_{ia} = \sum_{x_{ij} \in (E_i \cap O_a)} w_{ij}$.

Suppose that for each $B \in \mathcal{B}$, *there exists*

$$TD_\lambda(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{\ell} TD_\lambda(k, w_i^B).$$

Suppose further that, for $1 \leq i \leq \ell$, *there exists*

$$TD_{\lambda\mu}(k, \sum_{j=1}^{t} w_{ij}) - \sum_{a=1}^{u} TD_{\lambda\mu}(k, z_{ia}).$$

Then there exists a

$$TD_{\lambda\mu}(k, mt + \sum_{i=1}^{\ell}\sum_{j=1}^{t} w_{ij}) - \sum_{a=1}^{u} TD_{\lambda\mu}(k, m\beta_a + \sum_{i=1}^{\ell} z_{ia}).$$

5.3 Du Variations

Du [38] considers another use of incomplete transversal designs in Wilson's theorem. Denote by $ITD^s(k, t; h)$ an $ITD(k, t; h)$ that has s disjoint holey parallel classes of blocks. An $ITD^1(k, t; h)$ is equivalent to a $PITD(k, t; h^1 1^{t-h})$. Then we have:

Theorem 5.7 *Suppose that a* $ITD^s(k, t; h)$ *exists. Suppose that a* $TD(k, m)$ *exists. Let* $w_1, \ldots, w_s$ *be nonnegative integers, and suppose that an* $ITD(k, m + w_i; w_i)$ *exists for each* $i = 1, \ldots, s$. *Then a* $TD(k, mt + \sum_{i=1}^{s} w_i) - TD(k, mh + \sum_{i=1}^{s} w_i)$ *exists.*

If $w_i = 0$ *for some* i, $1 \leq i \leq s$, *then the stronger result is obtained that a*

$$TD(k, mt + \sum_{i=1}^{s} w_i) - TD(k, mh + \sum_{i=1}^{s} w_i) - (t - h)TD(k, m)$$

exists.

5.4 Another Variant

Colbourn [28] establishes the following:

Theorem 5.8 *If there exists a* $ITD(k, n+h; h)$ *for which* $(k-2)h = n$, *and there exists a* $TD(k, m)$, *then there exists an* $ITD(k, mn+(m-1)h; n^m(h(m-1))^1)$.

5.5 Wojtas Structures

For ease of exposition, we assume henceforth that $\lambda = \mu = 1$; the extensions to higher index are, for the most part, routine.

An examination of the propositions used in Brouwer [20] reveals that many, due to Wojtas, arise by inflating objects before filling them. We develop a framework here for presenting such constructions generally.

A *partial transversal design* of *order* or *groupsize* n, *blocksize* k, denoted here by $PTD(k, n)$, is a triple $(V, \mathcal{G}, \mathcal{B})$, where

1. V is a set of kn elements;
2. $\mathcal{G}$ is a partition of V into k classes (called *groups*), each of size n;
3. $\mathcal{B}$ is a collection of k–subsets of V (called *blocks*);
4. every unordered pair of elements from V is either contained in exactly one group, or is contained in *at most one* block. Elements appearing together in a group do not appear together in a block.

A *hole* H of order h in a $PTD(k, n)$ $(V, \mathcal{G}, \mathcal{B})$ is a set $H \subseteq V$ with $|H \cap G| = h$ for each $G \in \mathcal{G}$, and $H \cap B = \emptyset$ for each $B \in \mathcal{B}$.

Two holes H_1 and H_2 are *compatible* if $H_1 \cap H_2 \subset G$ for some $G \in \mathcal{G}$. Compatibility is a *weaker* condition than disjointness. A *Wojtas structure* of *order* n, *blocksize* k, and *holesizes* $\mathcal{M}$, denoted $WS(k, n, \mathcal{M})$, is a $PTD(k, n)$ $(V, \mathcal{G}, \mathcal{B})$ together with a set $\mathcal{H}$ of holes, so that

$$\mathcal{M} = \{\frac{|H|}{k} : H \in \mathcal{H}\}.$$

In addition, every pair of distinct holes from $\mathcal{H}$ are compatible. Moreover, every pair $\{x, y\} \subset V$ that does not appear in a group, *either* appears in exactly one block of $\mathcal{B}$, *or* appears in exactly one hole of $\mathcal{H}$, *but not both.*

If, in the definition, we replaced the single word "compatible" by the stronger word "disjoint", we would repeat the definition of incomplete transversal design. We make some simple (but important) observations.

Lemma 5.9 *A* $WS(k, n, \{1\})$ *is equivalent to a* $TD(k, n)$.

Lemma 5.10 *If a* $WS(k, n, \mathcal{M})$ *exists, and a* $TD(k, m)$ *exists for each* $m \in \mathcal{M}$, *then a* $TD(k, n)$ *exists. Indeed, a* $TD(k, n) - TD(k, m)$ *exists for every* $m \in \mathcal{M}$.

Lemma 5.11 *If* $WS(k, n, \mathcal{M})$ *and* $TD(k, w)$ *both exist, a* $WS(k, wn, \{wm : m \in \mathcal{M}\})$ *exists.*

Of course, when the holes are all disjoint, we have incomplete transversal designs. Sometimes we can employ other useful patterns of holes as well. Many variations are possible, but we just develop one generalization here.

A Wojtas structure $WS(n, k, \mathcal{M})$ $(V, \mathcal{G}, \mathcal{H}, \mathcal{B})$ is *partitionable of type*

$$(e_1, \ldots, e_\ell; \mathcal{N}, c),$$

denoted $PWS(k, n; e_1, \ldots, e_\ell; \mathcal{N}, c)$, if $\mathcal{M} = \mathcal{N} \cup \{e_1, \ldots, e_\ell\}$, and the holes are

$$\mathcal{H} = \{H_{\infty_1}, H_{\infty_2}, \ldots, H_{\infty_\ell}\} \cup \{H_{ij} : 1 \leq i \leq c, 1 \leq j \leq d\},$$

so that, for each $1 \leq i \leq c$, $H_{\infty_1}, H_{\infty_2}, \ldots, H_{\infty_\ell}, H_{i1}, H_{i2}, \ldots, H_{id}$ are all *disjoint*, $\mathcal{N}$ contains all hole sizes among the $\{H_{ij}\}$, and, for all $1 \leq i \leq c$, $\bigcup_{j=1}^{\ell} H_{\infty_j} \cup \bigcup_{j=1}^{d} H_{ij} = V$. Moreover, $H_{\infty i}$ is a hole of size e_i.

One might think about the $\{H_{ij} : 1 \leq j \leq d\}$, $1 \leq i \leq c$ as being "partial parallel classes of holes"; each, together with the special holes $\{H_{\infty_j}\}$, forms a "parallel class of holes".

In fact, we have the following equivalence:

Lemma 5.12 *A* $PITD(k, n; b_1, \ldots, b_s)$ *is equivalent to a*

$$PWS(k, n; b_1, \ldots, b_s; \emptyset, c)$$

for all c. *For every* $0 \leq \ell \leq s$, *it implies the existence of a*

$$PWS(k, n; b_1, \ldots, b_\ell; \{b_{\ell+1}, \ldots, b_s\}, 1).$$

Lemma 5.13 *A resolvable $TD(k,n)$ is equivalent to a $PWS(k,n;\emptyset;\{1\},n)$.*

Inflation, as in Lemma 5.11 works again, but we obtain a stronger result:

Lemma 5.14 *If a $PWS(k,n;e_1,\ldots,e_\ell;\mathcal{M}^\star,c)$ exists and a $TD(k,w)$ exists, so also does a $PWS(k,wn;we_1,\ldots,we_\ell;\{wm:m\in\mathcal{M}^\star\},c)$.*

Naturally, since a PWS is a WS, one can apply Lemma 5.10 to fill the holes. However, the structure of the partitioning can be exploited to obtain a more sophisticated result:

Lemma 5.15 *Suppose that there exists a $PWS(k,n;e_1,\ldots,e_\ell;\mathcal{M}^\star,c)$. Let $\gamma_1,\ldots,\gamma_c$ be nonnegative integers, and write $\sigma=\sum_{i=1}^{c}\gamma_i$. Suppose that, for every $m\in\mathcal{M}^\star$, and every $1\le i\le\ell$, there exists*

$$TD(k,m+\gamma_i)-TD(k,\gamma_i).$$

Suppose further that, for $2\le i\le\ell$, there exists

$$TD(k,e_i+\sigma)-TD(k,\sigma).$$

Then there exists

$$TD(k,n+\sigma)-TD(k,e_1+\sigma).$$

Again, we have the phenomenon that in the middle of the construction, we have ℓ holes of sizes $\sigma+e_i$ for $1\le i\le\ell$, but they all intersect in a hole of size σ. Lemma 5.15 gives one way to fill all but one of the holes. Here is another:

Lemma 5.16 *Suppose that there exists a $PWS(k,n;e_1,\ldots,e_\ell;\mathcal{M}^\star,c)$. Let $\gamma_1,\ldots,\gamma_c$ be nonnegative integers, and write $\sigma=\sum_{i=1}^{c}\gamma_i$. Suppose that, for every $m\in\mathcal{M}^\star$, and every $1\le i\le\ell$, there exists*

$$TD(k,m+\gamma_i)-TD(k,\gamma_i).$$

Suppose further that, for $1\le i\le\ell$, there exists

$$TD(k,e_i+\sigma)-TD(k,e_i).$$

Then there exists

$$TD(k,n+\sigma)-\sum_{i=1}^{\ell}TD(k,e_i).$$

When $\ell = 1$ and $\mathcal{M} = \{m\}$, we can also avoid filling one of the parallel classes of holes, to obtain:

Lemma 5.17 *Suppose that a $PWS(k, n; e; \{m\}, c)$ exists. Let $\gamma_1, \ldots, \gamma_c$ be nonnegative integers, and write $\sigma = \sum_{i=1}^{c} \gamma_i$. Suppose that there exists, for every $1 \leq i \leq c$, a*

$$TD(k, m + \gamma_i) - TD(k, \gamma_i).$$

Then there exists

$$TD(k, n + \sigma) - TD(k, e + \sigma) - \sum_{i=1}^{(n-e)/m} TD(k, m).$$

The power of Wojtas structures in general is that, rather than filling them immediately, one can inflate them and then fill them. This can often yield better results than are obtained by filling them and then inflating.

The additional power of partitioned Wojtas structures is the more sophisticated manner in which they can be filled.

It appears that Wojtas structures and partitioned Wojtas structures can lead to new incomplete transversal designs, but of course we have seen no ways to construct them except via equivalences to transversal designs and incomplete transversal designs.

5.6 Making Wojtas Structures

By now, it should come as no surprise that one way to construct Wojtas structures is to use Wilson's theorem in its many disguises. But an easier way to get some Wojtas structures is by removing one group from incomplete TD:

Lemma 5.18 *1. If a $TD(k+1, n)$ exists, then a $PWS(k, n; 0; \{1\}, n)$ exists.*

2. If an $ITD(k + 1, n; h)$ exists, then a $PWS(k, n; h; \{1\}, h)$ exists. If in addition a $TD(k, h)$ exists, then a $PWS(k, n; 0; \{1\}, n - h)$ exists.

3. If an $ITD(k + 1, n; b_1, \ldots, b_s)$ exists, and $TD(k, b_i)$ exist for $2 \leq i \leq s$, then a $PWS(k, n; b_1; \{1\}, b_1)$ exists.

Using Wilson–type constructions, more general Wojtas structures can be made. Here is a variant of Theorem 5.1, using the same notation:

Theorem 5.19 *Suppose that a $TD(k+\ell,t)$ exists. Let $\mathcal{D} \subset \mathcal{B}$. so that if $B \cap D \subset E_i$ for distinct $B, D \in \mathcal{D}$ and $1 \leq i \leq \ell$, then $w_i^B = 0$. Suppose that for each $B \in \mathcal{B} \setminus \mathcal{D}$, there exists*

$$TD(k, m + \sum_{i=1}^{\ell} w_i^B) - \sum_{i=1}^{\ell} TD(k, w_i^B).$$

Suppose that for each $1 \leq i \leq \ell$, there exists a

$$TD(k, \sum_{j=1}^{t} w_{ij}) - \sum_{j=1}^{t} TD(k, w_{ij}).$$

Then there exists a

$$WS(k, mt + \sum_{i=1}^{\ell}\sum_{j=1}^{t} w_{ij}, \mathcal{M})$$

where $\mathcal{M} = \{m + \sum_{i=1}^{\ell} w_i^B : B \in \mathcal{D}\}$.

When a single level is used, one can in fact make partitioned WS:

Theorem 5.20 *Suppose that a $TD_\mu(k+1,t)$ exists. Suppose that for each $B \in \mathcal{B}$, there exists*

$$TD_\lambda(k, m + w_1^B) - TD_\lambda(k, w_1^B),$$

whenever w_1^B *is nonzero. Let ζ be the number of $\{w_{1j} : 1 \leq j \leq t\}$ which are zero. Then we obtain a Wojtas structure $WS(mt + \sum_{j=1}^{t} w_{ij}, \{1, m, \sum_{j=1}^{t} w_{ij})\}$ in which all blocks of size m form ζ holey parallel classes for the hole of size $\sum_{j=1}^{t} w_{ij}$.*

Numerous variants are possible as well, but we do not consider them here.

5.7 Thwarts

Naturally, applications of Wilson's theorem depend on the presence of appropriate ingredients, and a natural question is to determine the ways in which

the blocks of a $TD(k+\ell,t)$ intersect the points of nonzero weight in the ℓ "extra" groups. With this in mind, we give a definition. Let ℓ be a nonnegative integer, and let $\mathcal{I} = \{i_1, \ldots, i_s\}$ with $0 \leq i_1, i_2, \cdots, i_s \leq \ell$. Further suppose that $0, s_1, s_2, \cdots, s_\ell \leq t$. Let $(X, \mathcal{G}, \mathcal{B})$ be a $TD(k+\ell,t)$ with $\mathcal{G} = \{G_1, \ldots, G_k, E_1, \ldots, E_\ell\}$. Then an $(\ell, \mathcal{I}, s_1, s_2, \ldots, s_\ell)$ *-thwart* is a set $S = \bigcup_{j=1}^{\ell} S_j$, where $S_j \subseteq E_j$ with $|S_j| = s_j$ for each $1 \leq j \leq \ell$, such that for every $B \in \mathcal{B}$, $|B \cap S| \in \mathcal{I}$.

Thwarts provide a convenient notation for simpler applications of Wilson's theorem, in which it is sufficient to know the number of points of intersection of each block with the points of nonzero weight in the extra groups. When different weights are chosen, however, more detailed structural information is required. Here we consider the structure of various thwarts, and we defer discussion of weightings until §9.3.

Given a set $\mathcal{I}$, let $\overline{\mathcal{I}}_\ell = \{\ell - i : i \in \mathcal{I}\}$.

Lemma 5.21 *If a $TD(k+\ell,t)$ contains an $(\ell, \mathcal{I}, s_1, \ldots, s_\ell)$-thwart, it also contains an $(\ell, \overline{\mathcal{I}}_\ell, t-s_1, \ldots, t-s_\ell)$-thwart.*

Levels

The simplest thwarts are found by simply truncating ℓ groups in each possible way to obtain:

Lemma 5.22 *Let ℓ be a positive integer, and let a $TD(k+\ell,t)$ exist. Then for all choices of integers $s_1, \ldots, s_\ell$ satisfying $0 \leq s_i \leq t$ for $1 \leq i \leq \ell$, the TD contains a $(\ell, \{0, 1, 2, \ldots, \ell-1, \ell\}, s_1, \ldots, s_\ell)$-thwart.*

Often we refer to such a thwart as ℓ *levels* in the TD.

Spikes and Stairs

If all of the points of nonzero weight are on a single block, we obtain an $(\ell, \{0,1\}, 1, 1, \ldots, 1)$–thwart, which we call a *spike*. Every $TD(k+\ell,t)$ contains such a spike.

We can choose one point to be of nonzero weight on each level, so that no block intersects the points of nonzero weight in more than two points. We call

the resulting $(\ell, \{0,1,2\}, 1, 1, \ldots, 1)$–thwart a *stair*. Stairs are essentially the analogue of arcs in projective planes. In fact, if the $TD(k+\ell, t)$ is the truncation of the $TD(t+1, t)$ arising from the desarguesian plane, the existence of an oval in the plane ensures that the stair is present for all choices of ℓ. If the $TD(k+\ell, t)$ arises in another way, we cannot assume to inherit the structure of a plane. Nevertheless, if $\binom{\ell-1}{2} < t$, simple counting ensures that the stair is present.

One can find an intermediate structure between spikes and stairs. Suppose that one point of nonzero weight is chosen on each extra group so that there are s blocks that are disjoint on the extra levels and intersect the extra levels in $x_1, \ldots, x_s$ positions, respectively. This is an $(\ell, \{0,1,2,x_1,\ldots,x_s\}, 1, 1, \ldots, 1)$–thwart, provided that no block intersects one of the s chosen blocks in more than two points of nonzero weight in the extra levels. Call this a *generalized stair*, and use the notation $(\ell; x_1, \ldots, x_s)$–*stair*. The presence of such thwarts is not as easily checked, but when $s = 1$ it suffices to ensure that $\binom{\ell-1}{2} - \binom{x_1}{2} + 1 < t$. Another case is when $s = 2$ and $x_1 + x_2 = \ell$; then the thwart is always present.

Stairs, Spikes and Levels

One can take a stair, spike or generalized stair on some levels, and truncate the remaining levels. For example, a spike involving u levels and a v disjoint levels truncated to $s_1, \ldots, s_v$ points leads to a $(u+v, \{0, 1, \ldots, v+1, v+u\}, 1^u s_1 s_2 \cdots s_v)$–thwart, in which only one block of size $v+u$ is present if $u > 1$.

Similarly, one can take a stair or generalized stair together with some truncated levels. For example, to an $(\ell; x)$–stair, we can append a truncated level on $s \leq t - \binom{\ell}{2} + \binom{x}{2} + 1$ points in a $TD(k,t)$ to obtain a $(\ell+1, \{0,1,2\}, 1^\ell s^1)$–thwart (in other words, adding a truncated level does not introduce a new intersection, provided that the truncated level is short enough).

Numerous variations are possible; the stair or spike can meet the added level(s) in one (or more) point(s). An $(\ell; x_1, x_2)$–stair could also have a further truncated level, and the blocks of size x_1 and x_2 could each meet or miss the truncated level.

Subplanes

When a projective plane of order t contains a subplane of order s, the corresponding $TD(t+1,t)$ contains an $(s+1,\{0,1,s+1\},s,s,\cdots,s)$–thwart. If the subplane is a Baer subplane, then the "0" can be omitted.

Instead deleting a point outside the subplane from the projective plane, we obtain an $(s^2+s+1,\{0,1,s+1\};1^{s^2+s+1})$–thwart.

Subsquares

When a $TD(k,t)$ contains a $TD(3,t)$ having a "subsquare", i.e. a sub-$TD(3,s)$, then the $TD(k,t)$ contains a $(3,\{0,1,3\},s,s,s)$–thwart. When $t=2s$, the "0" can be omitted.

Affine Subplane

When a projective plane of order t contains an affine subplane of order s, the $TD(t+1,t)$ contains a $(s+1,\{0,1,s\},s-1,s-1,\cdots,s-1)$–thwart.

Trinity

Wojtas [100] observed that one can truncate three levels but obtain a thwart with blocks interecting in 1, 2 and 3 points only — none in 0 points. The precise condition under which one can obtain such a $(3,\{1,2,3\},s_1,s_2,s_3)$–thwart is *open* for $TD(k,t)$ in general, although some bounds on s_1,s_2,s_3 are given in [30]. When the $TD(3,t)$ involved arises from a cyclic latin square (which can be assumed if we are free to choose three groups of the $TD(t+1,t)$ from the desarguesian plane of prime order t), a sufficient condition is that $s_1+s_2+s_3\geq 2t-1$. The conditions for arbitrary $TD(k,n)$ seem very difficult; see [30] for some other observations in this regard.

6 DIRECT CONSTRUCTIONS

Until this point, we have concentrated on recursive methods, and despite a large collection of constructions being introduced, we have failed to construct any examples. Let us remedy that situation. Lemma 4.1, together with the well–

known fact that projective planes exist (at least) for all prime power orders, gives the main set of basic ingredients:

Theorem 6.1 *If t is a prime power, a $TD(t+1,t)$ exists.*

Surprisingly little else in the way of *general* direct constructions is known, although much is known from clever hand and machine computations in specific cases. The main device used is to assume that the TD has a "reasonably large" automorphism group acting on it, and to use the structure of the automorphisms to reduce the computational search.

We require some basic definitions. Let $(\Gamma, \odot)$ be a group of order g. A $(g,k;\lambda)$-*difference matrix* is a $k \times g\lambda$ matrix $D = (d_{ij})$ with entries from Γ, so that for each $1 \le i < j \le k$, the multiset

$$\{d_{i\ell} \odot d_{j\ell}^{-1} : 1 \le \ell \le g\lambda\}$$

contains every element of Γ λ times. When Γ is abelian, typically additive notation is used, so that differences $d_{i\ell} - d_{j\ell}$ are employed.

A $(g,k;\lambda,\mu;u)$-*quasi-difference matrix* (QDM) is a matrix $Q = (q_{ij})$ with k rows, with each entry either empty (usually denoted by $-$) or containing a single element of Γ. Each row contains exactly λu empty entries, and each column contains at most one empty entry. Furthermore, for each $1 \le i < j \le k$, the multiset

$$\{q_{i\ell} - q_{j\ell} : 1 \le \ell \le \lambda(g-1+2u)+\mu, \text{ with } q_{i\ell} \text{ and } q_{j\ell} \text{ not empty}\}$$

contains every nonzero element of Γ λ times, and contains 0 μ times.

The essential connections with transversal designs follow:

Lemma 6.2 *1. A $(g,k;\lambda)$-difference matrix gives a $TD_\lambda(k+1,g)$.*

2. A $(g,k;\lambda,\mu;u)$-quasi-difference matrix with $\mu \le \lambda$ gives a $TD_\lambda(k,g+u)-TD_\lambda(k,u)$.

3. A $(g,k;1,0;u)$-quasi-difference matrix gives an $ITD^{g-u(k-2)}(k,g+u;u)$.

The latter statement gives one means to construct master designs for Du's variation, Theorem 5.7.

One main device for constructing quasi–difference matrices is to use $V(m,t)$ vectors. See [27] for a definition, and for the following:

Theorem 6.3 *A $V(m,t)$ vector gives a $(mt+1, m+2; 1, 0; t) - QDM$. A $V(m,t)$ vector exists if m and t are not both even,*

1. *whenever $mt+1 \leq 5000$, $m-1 \leq t$, $m \leq 10$ and $mt+1$ is prime, except when $m = 9$ and $t = 8$, as no $V(9,8)$ exists.*
2. *whenever $mt+1 \leq 5000$, $m-1 \leq t$, $m \leq 6$ and $mt+1$ is a prime power, except when $m = 3$ and $t = 5$, as no $V(3,5)$ exists.*

Related computational constructions for $(mt+1, m+2; 1, 0; t) - QDM$ are reported in [28]:

Theorem 6.4 *A $(mt+1, m+2; 1, 0; t) - QDM$ exists when $mt+1$ is a prime, $mt+1 < 5000$, $m-1 \leq t$, and*

1. *$t \equiv 2 \pmod 4$ and $m \in \{2, 4, 6\}$; or*
2. *$t \equiv 4 \pmod 8$ and $m \in \{2, 4\}$.*

Abel [2] found another useful family:

Theorem 6.5 *For $11t+1$ a prime, there exists a $(11t+1, k; 1, 0; t) - QDM$ exists for $k = 11$ if $198 < 11t+1 < 600$ and for $k = 12$ if $600 < 11t+1 < 992$.*

He also found $(9 \cdot 4 + 1, 9; 1, 0; 4)-$ and $(9 \cdot 8 + 1, 9; 1, 0; 8) - QDM$.

Despite these few more general computational results, most direct constructions are one–of–a–kind. For TD, we summarize in Table 1 known direct constructions, not obtained by one of the three previous constructions of QDM.

In some cases, a direct construction yields an idempotent $TD(k,t)$, which is a $PITD(k,t;1^t)$. In Table 2, direct constructions of idempotent TD having the same blocksize as the largest known TD on the same parameters are reported. Ganter, Mathon and Rosa [41] actually construct a $TD(4,10)$ having four disjoint parallel classes, the maximum known to date.

Order	Blocksize	Reference(s)
6	3	[89]
10	4	[18]
12	7	[53]
14	5	[90]
15	6	[76]
18	5	[94]
20	6	[91]
21	7	[65]
22	5	[94]
24	7	[3, 72, 104]
26	6	[26]
28	6	[1, 73]
30	6	[5]
33	7	[2]
34	6	[2]
36	7	[73]
38	6	[5]
39	6	[60]
40	9	[3]
42	5	[94]
44	6	[5]
48	7	[3, 73]
51	7	[2]
52	6	[1]
55	7	[60]
56	9	[60]
80	11	[3]
112	15	[3]
160	11	[3]
176	16	[3]
208	16	[3]
224	15	[3]
352	20	[3]
416	20	[3]
544	20	[3]
640	11	[3]
896	15	[3]

Table 1 Direct Constructions for TD

Order	Blocksize	Reference(s)
6	3	–
10	4	[41]
14	5	[11]
18	5	[106]
20	6	[2]
22	5	[6]
34	6	[2]
38	6	[2]
42	5	[11]
44	6	[2]
52	6	[2]

Table 2 Direct Constructions for Idempotent TD

A large number of *ad hoc* constructions for incomplete TD appear in the literature; we do not attempt to catalogue them all here. In Table 3, we report some of the small incomplete TD that have been constructed directly.

$ITD(4,6;2)$	[40]	$ITD(5,8;2)$	[84]
$ITD(6,10;2)$	[23]	$PITD(5,9;2^1 1^7)$	[108]
$PITD(5,11;2^1 1^9)$	[85]	$ITD(5,12;3)$	[84]
$PITD(5,12;2^6)$	[10]	$ITD(6,15;3)$	[28]
$PITD(5,14;3^1 1^{11})$	[4]		

Table 3 Some Incomplete TD

Of course, many more incomplete TD have been constructed directly. Some constructions of TD have proceeded by making an ITD with a hole and filling the hole; see [1, 5, 26, 94]. For $ITD(4,t;h)$, see [49] and references therein. For $ITD(5,t;h)$, see [4, 38, 39] and references therein for a number of direct constructions. For $ITD(4,hn;h^n)$, see [36, 86]. For $ITD(5,hn;h^n)$, see [10, 12, 55, 64, 85]; Dinitz and Stinson [36] also give some $ITD(6,2n;2^n)$ and $ITD(8,2n;2^n)$.

Colbourn [27] gives a number of constructions for QDM leading to ITD, and some sporadic examples appear in [61].

In terms of sporadic designs that find uses, we mention also the elliptic semiplane of Baker [8], which is a $\{7\}-GDD$ of type 3^{15}; and the $\{9\}-GDD$ of type 3^{33} by Mathon [59].

7 A KNOWLEDGE BASE FOR ITD

In planning the *CRC Handbook of Combinatorial Designs*, it became amply clear that one of the fundamental tables used by combinatorial design theorists, and by many researchers in applications areas, is the table of the best current lower bounds on the number of mutually orthogonal latin squares. Brouwer's table [20] has had an almost unbelievable impact on our ability to establish existence theorems for designs of all types.

However, as the years have passed since the last circulation of [20], constructions have required progressively more general types of incomplete MOLS, and tables

of these have not been compiled. Our first task was to decide what objects to tabulate for the *Handbook*, and our second was to decide what objects to tabulate for uses in constructions to make the first set of tables.

Our next task was to determine how to incorporate knowledge of the many constructions in §2–6 into an "inference engine" to apply the constructions. Finally, the main task was to apply this battery of constructions to produce the output tables desired.

§7–9 gives a summary of our efforts to build a computational environment. We first report on the fundamental objectives. Then we develop an outline of the architecture of the system, and discuss basic implementation decisions. We then describe some of the routines in more detail, in particular explaining the variants of Wilson's theorem actually coded. Finally, we examine what we believe to be the failures and successes of our approach.

7.1 Fundamental Objectives

We quickly realized that we could not report all of the consequences of applying every variant of every construction to all known ingredients. Thus we made some decisions initially about what information to report in the *Handbook*. Brouwer's table [20] set a standard in reporting the existence of MOLS to order 10000, and we felt that any shorter table would negatively impact the usefulness of the results. In view of the importance of *idempotent* MOLS ($PITD(k,t;1^t)$), we set as a goal the inclusion in this main table of an indication of when an idempotent solution can be found, again to order 10000.

Turning to incomplete MOLS, the candidates for what to tabulate are numerous. However, even a cursory examination of the literature shows that incomplete MOLS with one hole ($ITD(k,t;h)$) and uniform MOLS with holes ($PITD(k,hn;h^n)$) both play major roles. We determined to tabulate MOLS of side t with one hole of side h for $t \leq 1000$ and $h \leq 50$, and uniform MOLS with holes of type h^n for $h \leq 50$ and $n \leq 20$.

These fundamental objectives place minimum requirements on what must be tabulated and computed, but place no maximum on how much information the system may store and use to produce the basic output tables.

Other essential objectives beyond the output became clear. Justifications for table entries are needed in a machine–generated form that can be read (perhaps

with effort) and deciphered. We did not set as a basic requirement that these justifications be presented with the table, as we determined that the approximately 50000 justifications for the table of MOLS with one hole would not be terribly edifying.

A final objective is that the environment must support the introduction of new ingredients, and new constructions, without starting "from scratch". In particular, efficiency of updating the table is a serious concern.

8 ARCHITECTURE OF THE SYSTEM

Here we explore the major decisions made in designing the computational environment. The first main decisions concern how to represent the knowledge contained in the results reviewed in §2–6. The required knowledge comes in a number of flavours:

1. parameters for which an ITD is known to exist;
2. parameters for which other combinatorial structures are known to exist;
3. constructions for ITD and other structures.

We first examine how this knowledge is stored, and then examine how consequences of the constructions are determined.

8.1 The Data Base

We have already examined the question of what structures are to be recorded for output. Two remaining decisions have a great impact on the behaviour of the system: what structures to *store*, and what structures to *use*. The distinction that we are drawing here is perhaps subtle. Structures to be "stored" are those to be explicitly placed as a record in our database. Structures to be "used" are those stored externally, and those generated by procedures in our environment.

We treat the question of what to store, and how to interact with this data base, here. A careful reading of the constructions in §2–6 suggests that a wide variety of structures are employed in constructions. We decided not to store anything except for TD and ITD; knowledge about other structures is provided

by procedures that can either (a) store a table internally, or (b) read a different database. Having limited our attention to ITD, we needed to make a decision concerning which types of TD and ITD to store. In principle, essentially any ITD might be useful in one or more constructions. However, realistic limits on storage, and the need for constructions to have ingredients that are likely to be useful, dictates that one choose only certain types of ITD. After an examination of the literature, and *much* discussion, we decided to store

1. $TD(k,t)$ for $t \leq 10000$;
2. $PITD(k, hn; h^n)$ for $h = 1$ and $n \leq 10000$, and for arbitrary $h \leq 50$ and $hn \leq 1000$;
3. $ITD(k,t;h)$ for $t \leq 1000$ and $h \leq 50$; and
4. $PITD(k, a+bn; a^1b^n)$ for $a = b = 1$ and $n \leq 10000$, and for $a, b \leq 50$ and $a + bn \leq 1000$.

We also store an *authority* with each entry, and this authority is a plain text string.

To be precise, we do not store any actual ITD. What is stored, for parameters t and h, for example, is the value $k-2$ where k is the largest integer for which an $ITD(k,t;h)$ exists. Numbers stored are actually numbers of incomplete MOLS rather than the block size of the ITD, but of course this is a simple translation. One might argue that certain information is lost by storing only the value $k-2$, and not the whole ITD — some of the constructions given do depend on the structure of the ITD. However, the dependence on the structure of the ITD occurs primarily for $TD(t+1,t)$ arising from desarguesian planes, and these can be recognized from the parameters alone. Moreover, since a $TD(k, 10000)$ has 10^8 blocks, one questions the wisdom of storing it.

The wide variety of ingredients up to order 1000 suffices to handle constructions of TD and idempotent TD up to order 10000. But even more ingredients of order less than 1000 might be useful. We elected to "process" such ingredients as they are generated, rather than storing them.

Interaction with the database is carefully controlled. Each routine is permitted to read any database entry. However, updating the database is under the control of a small set of routines. The first of these, `interesting`, determines whether the ITD lies within bounds on blocksize, holesize, and order. The main routine is `record`. It first determines whether the ITD is interesting,

and then performs some error–checking on it (more about this later). If it fails basic error tests, the system stops without saving any information from the current run — errors are considered to be fatal, always. If the ITD passes, a determination is made by checking the current database entry to see if it is new. If it is interesting, apparently correct, and new, then the database is updated. Whenever an update occurs, output to the user is generated in the form of a command that indicates precisely which record was updated, and the authority for the update. The `record` routine goes on to explore certain consequences of the update; we return to this in the next section.

In order to permit distributed computation (loosely speaking), there is also a mechanism for updating the database entry without exploring the consequences of the update. The routine `rerecord` carries this out (the "interesting" check and error–checking are done regardless, so that one cannot subvert the error–checking by simply using `rerecord` instead of `record`).

This strategy isolates all of the database updates in one main routine. It prevents an entry from being updated without authority, and ensures that proper comparison against the previously best entry is made prior to update.

8.2 Procedural Knowledge

Our database excludes knowledge of other combinatorial structures, and also knowledge about the constructions of §2–6. We treat these in turn.

We planned for the eventuality that extensive databases might exist for block designs, PBD, symmetric designs, and so on. Since adequate databases of these do not exist at the present time, we encapsulated knowledge about existence results for each structure in a procedure. Each procedure contains a table as an internal data structure. However, the procedure is designed so that it can be easily replaced by a filter accessing a secondary database.

The representation of the constructions is a complex issue. Arguably, the constructions themselves are a form of data, and ought to be stored as such. One can imagine a large list of constructions, along with a general routine that "knows" how to apply a generic construction. We did not choose this path for two reasons. Firstly, the generality of the constructions mitigates against finding an efficient general mechanism for applying an arbitrary construction; perhaps more honestly, we should say that *we* saw no way to do this with reasonable efficiency. Secondly, it was not clear to us how to gain selective control

over application of the constructions. Some are more fruitful than others, and more effort in exploring unusual parameter sets for the ingredients is worthwhile. For these reasons, we decided to represent constructions as procedures.

8.3 Marrying the Procedures and the Data

Procedures can be loosely classified into the following categories:

- *database interface*: routines that determine when and how to update the database.
- *generators*: routines that generate parameter sets for existing combinatorial objects. Some generate by simple table look–up in an internal data structure; others access the database and repeatedly apply a construction from §2–6.
- *fillers*: Once an object is generated, a fill routine for that class of object incorporates knowledge about equivalences to other objects, and about ways in which to "fill holes".
- *output and tracing routines*: routines to extract information in tabular form from the database. These do not alter the database.
- *utility routines*: simple procedures for commonly repeated operations.

Drawing a distinction between generators and fillers is key in our approach. Let us take an easy example. Parameters for existing difference sets can be generated; one wants to explore the consequences of the existence of one. It may lead directly to a number of pairwise balanced designs by considering residue classes. It certainly leads to a symmetric design. This is in turn a block design (which is a PBD). The block design can be truncated to form $IPBD$ and GDD. Each has its own way to be "filled" to form ITD of various types, and these ITD may themselves imply the existence of further ITD by truncation, filling, or inflation. So our initial observation that it is easy to generate parameters for difference sets, while true, tells only a small part of the story. The situation is perhaps even more complicated with Wojtas structures, where the interaction of inflation and filling leads to numerous possibilities.

We examined every known construction with a goal of identifying what object(s) the construction generates, and how this object is converted (perhaps in many stages) to an interesting ITD. We then decoupled each construction into

a generator routine that makes the first intermediate object, and a sequence of filler routines that convert one object into another, ultimately appealing to the database interface to decide whether our efforts have met with success. The main advantage is that, once a filler for a type of object is included, *all* routines (generators and other fillers) that make this class of object can simply pass it to the fill routine. This philosophy permeates the constructions of §2–5; in Wilson's Theorem 5.1, for example, the ITD produced is in general not interesting by our rigid standard — but a filler for ITD with disjoint holes may succeed in producing a large number of interesting ITD from it!

One benefit that we anticipated (and encountered) is that constructions in the literature rarely exploit systematically the consequences of the intermediate objects constructed – it is just too tedious. Having fillers acting autonomously, however, permits unforeseen consequences to be discovered. The picture is not completely rosy, however. Often constructions that generate and fill in one step capture the interesting ways to fill the intermediate structure; although other ways to handle it may exist, they may lead to ITD that have blocksize too small, or order too large, to be useful.

The normal "discovery" of an ITD leads from a generator, through a sequence of fillers, to the database interface. There is no global plan; each routine knows what objects it can pass on to which other routines. Once at the database interface, we have discussed some of the work that `record` does. But `record` also explores the consequences of what is reported to it, so it in essence acts as a filler itself. Specifically, `record` first determines whether the order of the object is too large for it to lead to anything interesting. Fortunately, all of the constructions make ITD from ITD (and other objects) of smaller or equal order. The constructions also all have the property that they do not increase the blocksize. If the ITD given to `record` is too large or the blocksize is too small, it simply abandons it.

If the ITD given to `record` has small enough order and large enough blocksize, then *if it is not interesting*, or *it improves the blocksize in a database entry*, `record` passes it to `fill`. The `fill` routine carries out the filling and truncation operations in §2, and also makes the Wojtas structures if possible as in Lemma 5.18. It also may generate a sequence of further `record` operations. Once `fill` has attempted to record new entries, it invokes `inflate` to carry out the standard inflation from §2, and invokes `fillwojtas` to handle the Wojtas structures. Some care is required here to ensure that one doesn't simply cycle through these filler routines and `record`. The primary danger is with equivalent objects; these are filled only when they are *new* and *interesting*.

Evidently, a substantial effort is invested in every ITD that is reported to `record`; this is consistent with our philosophy that it does not matter where the ingredient came from — one wants to capture its consequences. This is particularly important if the ITD itself is not interesting, i.e. it alone will not alter a database entry. Any consequences of it not captured at this point are lost, unless it is generated again (by the same or a different method).

Nevertheless, we do draw the line short of exploring all consequences. We do not explore, for each ITD coming to `record`, its uses beyond simple truncation in making PBD and GDD; nor do we explore its use either as a master design or as an ingredient in Wilson's theorem. Interesting ITD may update a database entry, and if they do, their consequences in making PBD and GDD, and in Wilson's theorem, can be explored by the appropriate generator at a later time.

8.4 Operation of the System

The system starts with an empty database. A maximum order, a maximum holesize, and a minimum blocksize, are selected; these serve as parameters for `interesting` to determine what can be placed in the database. One then simply invokes generators in any order desired. Of course, some are likely to be more effective than others. Wilson's theorem with no ingredients has no consequences, so one must first "prime the pump" (or "prime power the pump"). A sensible regimen is to first apply generators arising from direct constructions – prime powers, difference and quasi–difference matrices, and `inputcases`. Then some fodder is present for the generators using symmetric designs, block designs, and PBD. Next constructions that make PBD and GDD from ITD, and Wilson's theorem can be applied.

No matter what order one applies the generators in, however, there is a problem. The generators cannot be ordered so that one generator only uses consequences of an earlier generator. Generators that employ ingredients from the database must be repeatedly applied until all consequences relevant to the blocksize, holesize and order specified are uncovered.

Even this hides an important problem. If, when a database entry is updated, `record` passes the ITD to `fill` but `fill` does not find appropriate ingredients to fill it, we need a mechanism to discover when these ingredients become available. It won't suffice simply to generate the ITD again, as `record` will recognize that the ITD does not cause a database update the second time, and hence rejects it without passing to `fill`. The answer may appear terribly

easy — just fill it anyways! But it is an enormous waste of time to repeatedly fill ITD whose consequences have been explored already. It seems difficult, moreover, to determine *a priori* whether new ingredients are now present that might lead to new consequences. Hence we added a facility for doing a forced fill; `forcefill` is a simple routine that passes each database entry in turn to `fill`, with the result that the new consequences, if any, are uncovered.

The system can run interactively or in batch mode. In either case, the permanent database is updated only at the end of the run, so that any abnormal or error termination does not corrupt the database.

8.5 Error Checking

A major concern in making tables is ensuring that the resulting information is accurate. The main error checking is in `record`, although every generator and filler routine attempts basic error checks as well. Since we do not have the actual ITD, there is no way to be sure that the construction actually produces the object. Hence we resorted to three types of error checks. The first is completely automated, and verifies that the parameters of the ITD meet known necessary conditions. Some redundancy is present to facilitate simple error checks; for example, for a $PITD(k, a+bn; a^1 b^n)$, the values $a+bn$, a, b and n are all separately reported, allowing a basic cosmetic test to be made.

Initial runs of the system were provided with a second database of TD, kindly provided by Andries Brouwer and run in 1993 (this database contains information from the constructions in [20, 25], along with some recent direct constructions). Any variations between the database entries for TD and this revised table of Brouwer were reported, and each was examined by hand to ensure the correctness of the construction used. This was, without a doubt, the best debugging tool that we had; hand verification of all of the TD entries would be nightmarish.

Finally, `record` reports explicitly objects for which the blocksize exceeds the square root of the order. These are rare and, although permitted by the necessary conditions, are remarkable enough to warrant some specific hand checks. It would be irresponsible to suggest that no errors could pass these error checks unnoticed. Nevertheless, one's confidence can be increased by substantial hand–checking and tracing, and this we have done.

8.6 Authorities

Perhaps the major disadvantage of having all of the routines decoupled is that no one routine or construction is solely responsible for causing a database update to occur. Nevertheless, one must be able to reproduce the "reasoning" applied by a sequence of routines in eventually effecting a database update. Brouwer [20] assigned to each entry a number, which indicated the single proposition responsible for the entry. The reader is left to deduce the parameters employed, which can typically be done with little effort. Our case is somewhat more complex; the set of constructions is richer, the ingredients are more general, and the constructions themselves are multistage. Our authorities are commensurately more complex. The price one pays for this is that the table of authorities is approximately seventy times larger than the table of actual numerical entries; hence reporting even an abbreviated authority becomes problematic. Simply saying "Wilson" is too little information to be useful; one must at least say which variant of the basic Wilson theorem, which thwart, and what weights are used. This in turn often leads to an intermediate structure, and one must say with what authority it was filled. We abandoned the thought of making a small table of authorities, and decided to retain authorities online that suffice for us to reconstruct the mechanism that led from generator to database entry.

Our rule of thumb is that each routine that generates or fills an object says, in broad terms, what it did to the object. A generator for symmetric designs, for example, may use the simple authority `symdes(9)` for the projective plane of order 9. Fillers indicate where they got the main ingredient that they are filling (i.e., they employ the authority provided for it), and in general do *not* indicate what ingredients are being used in the fill. For example, `fillbibd(symdes(9))` is an authority for a basic block design fill of a projective plane of order 9. Now `fillbibd` does not say what it did to the object; each routine is meant to be simple enough that this can be deduced without guesswork, but perhaps with some effort.

The biggest headache is with variants of the Wilson construction. Here we typically give in the authority the variant used, the type of the thwart, the parameters of the master design, and the weight m on the main levels; the luckless reader is left to deduce the weights on the extra levels, and the parameters of the other ingredients. For example, `W1 N(9) wt 7` is an application of Theorem 5.1 using $m = 7$ on a $TD(k, 9)$; one must deduce from the parameters of the design reported what weights have been used, and hence what ingredients are needed. This is not as hard as it sounds, with some practice. One has the

advantage of having a table of what ingredients exist, so that most potential choices for weights are eliminated immediately.

One might envisage recording sufficient information that each authority could be systematically translated into an existence proof. We saw no compelling reason to do this; it is true that a library would then exist that one could consult for verification of every entry — but it would be, for the most part, a very boring library! Therefore, we adopted the approach that authorities would remain internal to the environment, for verification purposes. Moreover, authorities are not proofs; rather, they constitute a high level trace which, together with the actual code, can be used with diligence to construct a proof.

9 IMPLEMENTING THE SYSTEM

Here we discuss some issues in the implementation of the system, and provide some detail on what precisely is realized from §2–6 in the system as it stands today (December 8, 1995). We do not attempt a complete description of every variant, but content ourselves to present the primary content. If no statement is made to the contrary, one should assume that the variants alluded to are those mentioned in §2–6. Applications of Wilson's theorem form a special case; there we are more explicit about what is included.

9.1 The Platform

Choosing an underlying platform on which to build the environment outlined is not an easy task. We have two main data structures with which to deal: tables of integers and tables of strings representing authorities. While the latter is primarily being written to and rarely consulted (by the system), the former is consulted very frequently and updated on occasion. Dynamic hash tables provide the right combination of easy access, sparse storage, and extendability.

This requirement led us to consider MAPLE, which provides dynamic hash tables as a primitive data structure. MAPLE also afforded the simple number–theoretic routines that we needed. Naturally, a major concern with any symbolic algebra environment is efficiency, but we decided that the presence of appropriate data structures, the support for mathematical computations, and the ease with which new routines could be prototyped and developed, outweighed our concerns about efficiency.

Hence the system has been developed as a package in MAPLE, and currently it runs only under Release 2 of MAPLE V.

9.2 Generators I: Other Designs

We have already seen the basic mechanism by which an entry is recorded, and its easier consequences explored automatically by `fill`, `inflate`, and `fillwojtas`. This frees the remaining routines from concern with the simpler bookkeeping. Here we consider the classes of generators, other than Wilson–type; the discussion must be read in conjunction with the discussion of fillers to obtain a reasonable understanding.

The simplest generator is `inputcases`. It just invokes `record` to capture sporadic direct constructions that are known; see §6. Next simplest are a number of generators that encapsulate direct constructions of ITD: prime powers, $V(m,t)$ vectors and related QDM, and the Dinitz–Stinson construction [36].

Turning to more complex objects, `bibd` is a generator that contains information about existence of block designs, cyclic block designs, resolvable block designs, and sporadic PBD. This information is taken essentially as is from [113, 116, 117]. Each is passed to an appropriate fill routine. In addition, truncations as in §4.1 are carried out, and the PBD and GDD produced are passed to appropriate fillers. Along similar lines, `symdes` carries information about symmetric designs; it invokes routines to examine the consequences of subplanes, ovals, and line–flips on these. The routine `diffset` concerns residue classes in difference sets. Here the difference sets have to be known explicitly – parameters alone won't do.

The next main class contains routines `tdgdd`, `itdgdd` and `abntdgdd`. Each makes PBD and GDD from ITD as discussed in §3.3. We reiterate that only those ITD that have been stored in the database become candidates for making PBD and GDD. Some information is lost thereby, since truncations of objects with three different hole sizes are not treated, for example.

Before proceeding to Wilson's theorem, we want to spend a moment to point out that, while Wilson's theorem is by far the most prolific generator, one ought not to dismiss the more "classical" techniques using designs. Two reasons come to mind. Wilson's theorem permits many of the same truncations as does general truncation of PBD; comparing §3.3 and §5.7 reveals this. However, a quick comparison is not sufficient to notice that truncations for PBD constructions

are more general — for example, removing a block from a TD gives useful PBD [9], while no analogous thwart is of any use in Wilson's theorem. A second reason is that while Wilson's theorem provides the best bound in most cases, the bound is rarely large. Sporadic bounds that arise from configurations in projective planes, while not frequent, are often substantially better. Thus care must be taken to obtain consequences of other designs; this is especially true since Wilson's theorem can be voracious in its appetite for ingredients.

Adding new generators is a relatively easy task, whether they make block designs with extra properties, PBD with extra properties, GDD, symmetric designs, cyclic difference sets, or some flavour of ITD. If they make something else that may be useful, corresponding filler routines must be supplied as well.

9.3 Generators II: Wilson's Theorem

There can be little doubt that the many variations of Wilson's theorem produce the vast majority of the results. This has caused an epidemic of versions of the theorem; some get so complex that we cannot find any ingredients to actually apply them, and some are just too constrained to take advantage of the ingredients that we know. Striking a balance is important, but not easy. Which variants should be used, with which types of thwarts, and what weightings on the thwarts? We have adopted the strategy here of starting with general statements and evolving them into useful constructions in two stages; first, in §5.7 we described possible structures on the points of nonzero weight in the extra levels, and here we describe weights applied to those points in the thwart.

The amazing flexibility of Wilson's theorem is a blessing and a curse. The blessing is obvious, the curse perhaps less so. It is that there does not appear to be any sane manner in which to explore all of the consequences of the theorem, even with the known ingredients and a reasonable bound on the order. Almost any ITD might be useful as an ingredient, and almost any thwart might happen to have sufficient ingredients present for the right weights.

Our initial goal was to have generators to cover the applications of Wilson's theorem and its variants that actually appear in the literature. We exceeded this goal in many areas, and fell short in others. In retrospect, the reason for falling short is completely obvious: We do not have enough ingredients stored to cover every application discussed in the literature. The two worst omissions are $ITD(k,t;h_1,h_2)$ and $ITD^s(k,t;h)$; in both cases, we were able to generate sufficient information about these "on the fly" to cover the primary

applications, but their omission necessarily limits what the environment can discover.

We proceeded by implementing generators for a number of simple thwarts with simple weighting functions, and generalized those that were productive to more complex thwarts and/or more general weights. It cannot be said often enough that the approach of "implement every routine as generally as possible" is doomed to fail unless one is Methuselah with a lot of computers. This means inevitably that we are reporting *some* consequences of Wilson's theorem. Hopefully we cover the most interesting and most productive variants, but we have no assurance even of that. With this proviso in mind, it is perhaps most important that we say what is included. We also report on some utility routines that are used throughout the Wilson–type generators.

We classify the applications by the type of thwart.

One Level

When the thwart is a single level, the most extensive implementation has been done. All of the variants of Wilson's theorem are treated to some extent here. In particular, we try all possible assignments of weights to the points of the thwart. This is accomplished by using a simple routine for discrete convolution, as follows. For each choice of weight m on the main levels, we determine a vector $(x_0, x_1, \ldots, x_c)$ in which x_i is the largest value of k for which an $ITD(k+2, m+i; i)$ is known. The limit c can be easily calculated given a lower bound on the block size of interest, and m. The *s–fold convolution* of the vector is a vector $(y_0, \ldots, y_{cs})$, where

$$y_j = \max(\min(x_{\gamma_1}, \ldots, x_{\gamma_s}) : 0 \le \gamma_i \le c \text{ for } 1 \le i \le s, \text{ and } \sum_{i=1}^{s} \gamma_i = j).$$

A utility routine calculates s–fold convolutions, permitting us to determine the best way to put a total weight of j on a level with s points of nonzero weight.

Variants using incomplete master designs (with one hole), and incomplete ingredients on the main levels (with spanning sets of holes) are included.

Two Levels

We are unable to implement all weightings on all two level thwarts, because we have stored no information explicitly about ITD with two holes. Nevertheless,

a substantial amount can be done. First we choose a weight m on the main levels. In the simplest implementation, we put weights 0 and 1 on the points of one level, and then determine what the maximum block size is should we put a weight of i on a point of the second level. Again we obtain a vector $(x_0, x_1, \ldots, x_c)$ as in the one level case; discrete convolution then permits us to try all possible distributions of weights on the second level.

This is generalized somewhat. Suppose that d is a divisor of m and write $m = de$. For each choice of divisor d, we assign weights on one level from $S = \{0, 1, d\}$, $\{0, d\}$, or $\{1, d\}$. Since a weight of d is to be permitted on the first level, we must determine whether an $ITD(k, m+d+i; d, i)$ exists to decide whether a weight of i can be used in the second level. We do not have this information, *but* if a $PITD(k, m + d + i; i^1 d^{e+1})$ exists, and a $TD(k, d)$ exists, we can produce such an ingredient. In this way we again determine the possible weights on the second level and use discrete convolution.

Fewer variants have been explored here, due to the complexity of the weightings encountered. Only weights of 0 and 1 have been considered on the extra levels when the master design is incomplete, for example.

One Spike

In the case of one spike (§5.7), again all possible weight assignments to the spike have been examined, using discrete convolution. The implementation actually replaces the role of the spike by a sub-TD (a spike is just a sub-TD of order 1), and weights on all of the points of the sub-TD among the extra levels are chosen independently.

In [25], the Remark following Corollary 1.4 indicates that one ingredient is not needed if the spike has maximum size; however, this appears not to be correct, and our spike implementation does not use that remark.

Spike and a Level

One spike and one level are treated as follows. In the first place, weights 0 and 1 are placed on points of the level, and arbitrary weights on points of the spike. Then secondly, weights 0 and 1 are placed on points of the spike, and arbitrary weights on points of the level. If the spike meets the level, the "corner" point plays a special role as any block meeting the corner which is not the spike block

meets the thwart in *only* the corner point. This enables us to weight it with fewer constraints.

Spike and Two Levels

A very basic implementation of a spike and two levels, using weights 0 and 1, was coded but led to few results, but much computation time!

Generalized Stair and a Level

A number of different thwarts were tried here. First, a thwart is tried which is an $(\ell; x_1)$–stair and a single level, where the points of the level do not lie on a block that meets the stair in two or more points. The level may, however, share a "corner" with the generalized stair, which we assume (if present) to be on the block of size x_1. The generalized stair is given weight 1, and all possible weight distributions on the points of the level are considered.

Next we try a "goalpost" thwart. This is an $(a+b; a, b)$–stair with the a–block and b–block on different levels; then a level is added in which all points do not occur on blocks that meet the stair in two or more points. There is the possibility to add either or both of the corner points to the level, where the a– and b–blocks meet it. We may also consider the case where the a– and b–blocks meet the level in the same point. All of these variants are tried, but just for weight 1.

When no extra level is present, a goalpost is essentially two spikes, which may or may not meet at a corner, but are otherwise on disjoint levels. Here we tried weight 1 on one spike, arbitrary weights on the other spike, and an arbitrary weight on the corner (which is less constrained).

Next we try a "trident" thwart, which is a $(a+b+c; a, b, c)$–stair. Only weight 1 is attempted.

Trinity

The three level thwarts that have all blocks meeting them in 1, 2 or 3 points (and the complementary thwarts which blocks meet in 0, 1 or 2 points) were vexating. Computing the bounds from [30] for such thwarts is not computationally easy. We implemented a routine which calculates a table that says, given the sizes of two levels, what the bounds are for the size of the third level so that the thwart

can be assumed present. However, the only computation run to completion was for the case that the master TD arises from a plane of prime order, making the bounds easy to compute. All weights used on the trinity three level thwart and its complementary thwart were taken to be 1. A variant was explored that extends the three level thwart to four levels, permitting intersections of size 1, 2, 3 and 4; however, it was not terribly productive.

Subsquares

Here we tried weight 1 on subsquare thwarts and their complementary thwarts. We also tried the $(k, \{0, 1, k\}; t^k)$–thwarts arising from sub-TD.

Projective Subplanes

We employed thwarts arising from Baer subplanes and the partitions into Baer subplanes, with all weights being 1. We employed projective subplanes of order 2 to make $(7, \{0, 1, 3\}; 1^7)$–thwarts, and assigned weight 1.

Affine Subplanes

We employed the $(4, \{1, 3, 4\}; (t-2)^4)$ complementary thwart arising from puncturing a point from an $AG(2, 3)$, using weight 1.

This description is meant to convey the flavour of what is present for Wilson–type generators in the system. Each also generates Wojtas structures for inflation prior to filling, and each employs at least a rudimentary application of Wilson's theorem using ingredients that are idempotent on the main levels.

The Du variations (§5.3) are only implemented for master designs that arise from direct constructions of quasi–difference matrices, and no effort has thus far been made to extend their use in the system (lack of ingredients being a big hurdle to surmount).

9.4 Fillers

Fillers are provided for numerous intermediate structures encountered. We have examined `fill`, a basic filler for ITD, already; we have also encountered

`fillwojtas`. It implements the filling and inflation of Wojtas structures discussed in §5.5.

Fillers exist in addition for block designs (using arcs when known), GDD, PBD, resolvable block designs, resolvable PBD, and symmetric designs. For separable PBD, the Bose–Shrikhande–Parker theorem is implemented as a filler.

One can see that fillers may invoke a number of other fillers in addition to `record`.

10 EXPERIENCE WITH THE SYSTEM

In at least one regard, our venture was a success. We succeeded in producing tables of TD and ITD for the *CRC Handbook of Combinatorial Designs*; see [114] and [115]. In some ways, our expectations were exceeded. For example, the surprisingly good results for idempotent TD were not anticipated. Nor did we expect to make substantial improvements in the existence of $ITD(5, t; h)$ where substantial effort has been invested [38, 39]; but the computations provided a valuable set of updates, reported in [4] (along with new direct constructions).

10.1 A *Post Mortem*

The successes were matched by some disappointments. We review them here, so that when the inevitable time comes to make the next generation of MOLS table, some of our experience can be profitably used. The real disappointment is efficiency.

One would like to just tell the system that a new ingredient has been found, and have it be "clever" enough to explore what impact that might have; but this process is agonizingly slow. Depending on the ingredient, any time from seconds to days can be expended exploring its consequences. Generality and efficiency are fighting each other here, and our desire is to make the generators and fillers *even more general.* Clearly, some significant effort must be made to store more and recompute less; the balances here in generality/efficiency, and storage/time, are subtle. We cannot but feel that useful auxiliary information could be stored, either to reduce recomputation, or to equip the generators

with priorities that permit them to explore rich veins first. Having said this, however, we see no easy mechanism to accomplish it.

A valuable type of information that we have not been able to collect is when a construction "nearly" succeeds. Constructions that lack one key ingredient suggest important directions for research. Our approach, however, makes this determination well nigh impossible. As researchers, we tend to ask three types of questions:

- Given certain parameters, what is the largest blocksize that can be made with current constructions?
- Given a new ITD (or other object), what consequences does it have taken with known constructions and known ingredients?
- Given a construction, what does it make with known ingredients?

Our approach emphasizes the last question. The first can be answered in our system only by table look-up. The second question poses the worst problems, as we have seen. Emphasizing construction–based approaches appears to be sensible for making tables, but some real thought should be given to how to insert new ingredients efficiently.

Having read the first sections, the reader would be forgiven for having a feeling of *deja vu* in the latter sections. Indeed many of the constructions are quite similar, and this poses a problem with redundancy. Two generators need not make disjoint sets of objects, but of course it is more efficient if they do. Some of the inefficiencies that we have encountered relate to the difficulty of having each object generated by a single generator, insofar as that is possible.

Another type of redundancy occurs in the code. This happens especially where an object is created with a set of incompatible holes; a filler which "knew" how to fill a partial TD with prescribed holes intersecting in prescribed ways would simplify a number of the Wilson–type generators. We tried adding such a filler routine, but efficiency considerations forced us to retreat.

The authorities ought to be easier to read by an expert in the field, and not require that one pore over code to see what happened. While that occurrence is fortunately rare, one feels that a more systematic mechanism for assigning authorities would eliminate the need to refer to a specific routine in the package.

Excepting efficiency, these disappointments are somewhat minor, given our objectives.

10.2 Some Concluding Remarks

Even such a lengthy discussion fails to convey all of the details, and that is as it must be. Rather than recapitulating earlier remarks, we close with some speculations.

Perhaps the most interesting speculation is what the true values of the number of MOLS are. We haven't succeeded in finding 15 MOLS of order 8360. Can it really be possible that they fail to exist? Frankly, we doubt it. (In fact, we would be surprised if that result weren't improved upon within a year.) But what is the truth here? 20? 200? 2000? 8000? We have no idea. It is diverting to imagine what would happen to combinatorial design theory if we suddenly constructed $\frac{t}{2}$ MOLS for every order $t > 6$. Making tables as we have done shouldn't be taken as evidence that the lower bounds obtained are in any way close to the truth; or that the known constructions are related even remotely to the ones that reveal something close to the truth. We hope that another breakthrough of the magnitude of Wilson's theorem does occur, but as to the form it might take, we make no guesses, educated or otherwise.

Another interesting speculation is whether it is feasible with current technology to build a "Design Theory Expert". More precisely, could we build one that manages within a finite amount of time to say something interesting? We would have to equip it with an enormous variety of constructions, and a huge library of data. That seems possible, although the effort involved would be daunting. Its mandate could be to find new designs with existing constructions; or it could be to find new constructions. People succeed at this (sometimes), partly because they acquire a sense of what of the current knowledge is relevant to the task at hand. Their knowledge may be incomplete, but nevertheless the right ingredients all hove in view at the same time, and presto! Managing to code what we mean by "interesting" and "relevant" is the real problem. We expect that systems with more limited expertise are more likely to be feasible. Our system falls short of being such an expert, primarily because it has no sense of when a construction is likely to succeed or likely to fail; and it doesn't learn about situations where it has wasted its time before, nor why. It does, however, have a substantial library of knowledge in the database and procedures upon which to build expertise and learning.

Acknowledgments

Our primary indebtedness is to Andries Brouwer, for blazing the trail many years ago in [20], and for his participation in [114]. Too many people to mention here have provided useful pointers to the literature and comments; thanks to Julian Abel, Frank Bennett, Malcolm Greig, Esther Lamken, Ron Mullin, Paul Schellenberg, Doug Stinson, Yin Jian Xing, Hantao Zhang, Xiao Jun Zhu, and Zhu Lie. Julian Abel deserves special mention for his great contributions in checking the tables and improving the results.

REFERENCES

[1] R.J.R. Abel, Four mutually orthogonal latin squares of orders 28 and 52, *J. Combinatorial Theory* A58 (1991), 306–309.

[2] R.J.R. Abel, private communications, 1993-94.

[3] R.J.R. Abel and Y.W. Cheng, Some new MOLS of order $2^n p$ for p a prime power, *Austral. J. Combin.* 10 (1994), 175–186.

[4] R.J.R. Abel, C.J. Colbourn, J.X. Yin and H. Zhang, Existence of transversal designs with block size five and any index λ, preprint.

[5] R.J.R. Abel and D.T. Todorov, Four MOLS of order 20, 30, 38 and 44, *J. Combinatorial Theory* A64 (1993), 144–148.

[6] R.J.R. Abel, X. Zhang and H. Zhang, Three mutually orthogonal idempotent latin squares of orders 22 and 26, *J. Stat. Plan. Infer.*, to appear.

[7] R.D. Baker, Whist tournaments, *Congressus Num.* 14 (1975), 89–100.

[8] R.D. Baker, An elliptic semiplane, *J. Combinatorial Theory* A25 (1978), 193–195.

[9] F.E. Bennett, Pairwise balanced designs with prime power block sizes exceeding 7, *Ann. Discrete Math.* 34 (1987), 43–64.

[10] F.E. Bennett, C.J. Colbourn and L. Zhu, Existence of certain types of three HMOLS, *Discrete Math.*, to appear.

[11] F.E. Bennett, K.T. Phelps, C.A. Rodger and L. Zhu, Constructions of perfect Mendelsohn designs, *Discrete Math.* 103 (1992), 139–151.

[12] F.E. Bennett and L. Zhu, Existence of HSOLSSOM(h^n) where h is even, preprint.

[13] T. Beth, D. Jungnickel and H. Lenz, *Design Theory*, Cambridge University Press, Cambridge, 1986.

[14] R.C. Bose, On the applications of properties of Galois fields to the problem of construction of hypergraecolatin squares, *Sankhya* 3 (1938), 328–338.

[15] R.C. Bose, I.M. Chakravarti and D.E. Knuth, On methods of constructing sets of mutually orthogonal latin squares using a computer, *Technometrics* 2 (1960), 507–516.

[16] R.C. Bose, I.M. Chakravarti and D.E. Knuth, On methods of constructing sets of mutually orthogonal latin squares using a computer II, *Technometrics* 3 (1961), 111–118.

[17] R.C. Bose and S.S. Shrikhande, On the construction of sets of mutually orthogonal latin squares and the falsity of a conjecture of Euler, *Trans. Amer. Math. Soc.* 95 (1960), 191–209.

[18] R.C. Bose, S.S. Shrikhande and E.T. Parker, Further results on the construction of mutually orthogonal latin squares and the falsity of Euler's conjecture, *Canad. J. Math.* 12 (1960), 189–203.

[19] R.K. Brayton, D. Coppersmith and A.J. Hoffman, Self-orthogonal latin squares of all orders $n \neq 2, 3, 6$, *Bull. Amer. Math. Soc.* 80 (1974), 116–118.

[20] A.E. Brouwer, The number of mutually orthogonal latin squares — a table to order 10000, Research Report ZW 123/79, Math. Centrum, Amsterdam, 1979.

[21] A.E. Brouwer, On the existence of 30 mutually orthogonal latin squares, Math. Centrum Report ZW77, Amsterdam, 1980.

[22] A.E. Brouwer, A series of separable designs with application to pairwise orthogonal latin squares, *Europ. J. Combin.* 1 (1980), 39–41.

[23] A.E. Brouwer, Four MOLS of order 10 with a hole of order 2, *J. Stat. Plan. Infer.* 10 (1984), 203–205.

[24] A.E. Brouwer, Recursive constructions of mutually orthogonal latin squares, *Ann. Discrete Math.* 46 (1991), 149–168.

[25] A.E. Brouwer and G.H.J. van Rees, More mutually orthogonal latin squares, *Discrete Math.* 39 (1982), 263–281.

[26] C.J. Colbourn, Four MOLS of order 26, *J. Comb. Math. Comb. Comput.* 17 (1995), 147–148.

[27] C.J. Colbourn, Some direct constructions for incomplete transversal designs, *J. Stat. Plan. Infer.*, to appear.

[28] C.J. Colbourn, Construction techniques for mutually orthogonal latin squares, in: *Combinatorics Advances*, Kluwer, to appear.

[29] C.J. Colbourn, J.H. Dinitz and D.R. Stinson, More thwarts in transversal designs, *Finite Fields Applic.*, to appear.

[30] C.J. Colbourn, J.H. Dinitz and M. Wojtas, Thwarts in transversal designs, *Des. Codes Crypt.* 5 (1995), 189–197.

[31] C.J. Colbourn, J. Yin and L. Zhu, Six MOLS of order 76, *J. Comb. Math. Comb. Comput.*, to appear.

[32] D.J. Crampin and A.J.W. Hilton, On the spectra of certain types of latin squares, *J. Combinatorial Theory* A19 (1975), 84–94.

[33] R.H.F. Denniston, Subplanes of the Hughes plane of order 9, *Proc. Cambridge Phil. Soc.* 64 (1968), 589–598.

[34] R.H.F. Denniston, Some maximal arcs in finite projective planes, *J. Combinatorial Theory* A6 (1969), 317–319.

[35] M.J. de Resmini, On the Dempwolff plane, in: *Finite Geometries and Combinatorial Designs*, Amer. Math. Soc., 1987, pp. 47–64.

[36] J.H. Dinitz and D.R. Stinson, MOLS with holes, *Discrete Math.* 44 (1983), 145–154.

[37] D.A. Drake and H. Lenz, Orthogonal latin squares with orthogonal subsquares, *Archiv der Math.* 34 (1980), 565–576.

[38] B. Du, On incomplete transversal designs with block size five, *Util. Math.* 40 (1991), 272–282.

[39] B. Du, On the existence of incomplete transversal designs with block size 5, *Discrete Math.* 135 (1994), 81–92.

[40] L. Euler, Recherches sur une nouvelle espèce de quarrés magiques, *Verh. Zeeuw. Gen. Weten. Vlissengen* 9 (1782), 85–239.

[41] B. Ganter, R. Mathon and A. Rosa, A complete census of (10,3,2) designs and of Mendelsohn triple systems of order ten. II. Mendelsohn triple systems with repeated blocks, *Congressus Num.* 22 (1978), 181–204.

[42] M. Greig, Designs from configurations in projective planes, unpublished, 1992.

[43] R. Guérin, Aspects algebraiques de problème de Yamamoto, *C.R. Acad. Sci.* 256 (1963), 583–586.

[44] R. Guérin, Sur une généralisation de la méthode de Yamamoto pour la construction de carrés latins orthogonaux, *C.R. Acad. Sci.* 256 (1963), 2097–2100.

[45] R. Guérin, Existence et propriétés des carrés latin orthogonaux II, *Publ. Inst. Stat. Univ. Paris* 15 (1966), 215–293.

[46] H. Hanani, On the number of orthogonal latin squares, *J. Combinatorial Theory* 8 (1970), 247–271.

[47] A.S. Hedayat and E. Seiden, On the theory and application of sum composition of latin squares and orthogonal latin squares, *Pacific J. Math.* 54 (1974), 85–113.

[48] K.E. Heinrich, Near–orthogonal latin squares, *Util. Math.* 12 (1977), 145–155.

[49] K.E. Heinrich and L. Zhu, Existence of orthogonal latin squares with aligned subsquares, *Discrete Math.* 59 (1984), 241–248.

[50] K.E. Heinrich and L. Zhu, Incomplete self–orthogonal latin squares, *J. Austral. Math. Soc.* A42 (1987), 365–384.

[51] J.W.P. Hirschfeld, *Projective Geometries over Finite Fields*, Oxford University Press, 1979.

[52] J.D. Horton, Sub–latin squares and incomplete orthogonal arrays, *J. Combinatorial Theory* A16 (1974), 23–33.

[53] D.M. Johnson, A.L. Dulmage and N.S. Mendelsohn, Orthomorphisms of groups of orthogonal latin squares, *Canad. J. Math.* 13 (1961), 356–372.

[54] T.P. Kirkman, On the perfect r–partitions of $r^2 - r + 1$, *Transactions of the Historic Society of Lancashire and Cheshire* (1850), 127–142.

[55] E.R. Lamken, The existence of orthogonal partitioned incomplete latin squares of type t^n, *Discrete Math.* 89 (1991), 231–251.

[56] J.H. van Lint, *Combinatorial Theory Seminar*, *Lect. Notes Math.* 832, Springer, 1974.

[57] H.F. MacNeish, Euler squares, *Ann. Math. (NY)* 23 (1922), 221–227.

[58] H.B. Mann, The construction of orthogonal latin squares, *Ann. Math. Statist.* 13 (1942), 418–423.

[59] R.A. Mathon, unpublished.

[60] W.H. Mills, Some mutually orthogonal latin squares, *Congressus Num.* 19 (1977), 473–487.

[61] R.C. Mullin, Finite bases for some PBD-closed sets, *Discrete Math.* 77 (1989), 217–236.

[62] R.C. Mullin, P.J. Schellenberg, D.R. Stinson and S.A. Vanstone, Some results on the existence of squares, *Ann. Discrete Math.* 6 (1980), 257–274.

[63] R.C. Mullin and D.R. Stinson, Holey SOLSSOMs, *Util. Math.* 25 (1984), 159–169.

[64] R.C. Mullin and L. Zhu, The spectrum of HSOLSSOM(h^n) where h is odd, *Util. Math.* 27 (1985), 157–168.

[65] A.V. Nazarok, Five pairwise orthogonal latin squares of order 21, *Issled. oper. i ASU*, 1991, pp. 54–56.

[66] T.G. Ostrom and F.A. Sherk, Finite projective planes with affine subplanes, *Can. Math. Bull.* 7 (1964), 549-560.

[67] E.T. Parker, Construction of some sets of mutually orthogonal latin squares, *Proc. Amer. Math. Soc.* 10 (1959), 946–949.

[68] E.T. Parker, Nonextendability conditions on mutually orthogonal latin squares, *Proc. Amer. Math. Soc.* 13 (1962), 219–221.

[69] C. Pellegrino and P. Lancelotti, A construction of pairs and triples of k–incomplete orthogonal arrays, *Ann. Discrete Math.* 37 (1988), 251–256.

[70] L. Puccio and M.J. de Resmini, Subplanes of the Hughes plane of order 25, *Arch. Math.* 49 (1987), 151–165.

[71] J.F. Rigby, Affine subplanes of finite projective planes, *Can. J. Math.* 17 (1965), 977-1009.

[72] C.E. Roberts Jr., Sets of mutually orthogonal latin squares with 'like subsquares' , *J. Combinatorial Theory* A61 (1992) 50–63.

[73] C.E. Roberts Jr., Sets of mutually orthogonal latin squares with 'like subsquares' II, preprint.

[74] R. Roth and M. Peters, Four pairwise orthogonal latin squares of order 24, *J. Combinatorial Theory* A44 (1987), 152–155.

[75] F. Ruiz and E. Seiden, Some results on the construction of orthogonal latin squares by the method of sum composition, *J. Combinatorial Theory* A16 (1974), 230–240.

[76] P.J. Schellenberg, G.H.J. van Rees and S.A. Vanstone, Four pairwise orthogonal latin squares of order 15, *Ars Combinatoria* 6 (1978), 141–150.

[77] E. Seiden, A method of construction of resolvable BIBD, *Sankhya* A25 (1963), 393–394.

[78] E. Seiden and C.J. Wu, Construction of three mutually orthogonal latin squares by the method of sum composition, in: *Essays in Probability and Statistics*, Shinko Publishing, Tokyo, 1976.

[79] J. Singer, A theorem in finite projective geometry and some applications to number theory, *Trans. Amer. Math. Soc.* 43 (1938), 377–385.

[80] D.A. Sprott, A series of symmetrical group divisible incomplete block designs, *Ann. Math. Statist.* 30 (1959), 249–251.

[81] D.R. Stinson, On the existence of 7 and 8 mutually orthogonal latin squares, *Ars Combinatoria* 6 (1978), 113–115.

[82] D.R. Stinson, On the existence of 30 mutually orthogonal latin squares, *Ars Combinatoria* 7 (1979), 153–170.

[83] D.R. Stinson, A generalization of Wilson's construction for mutually orthogonal latin squares, *Ars Combinatoria* 8 (1979), 95–105.

[84] D.R. Stinson, The equivalence of certain incomplete transversal designs and frames, *Ars Combinatoria* 22 (1986), 81–87.

[85] D.R. Stinson and L. Zhu, On sets of three MOLS with holes, *Discrete Math.* 54 (1985), 321–328.

[86] D.R. Stinson and L. Zhu, On the existence of MOLS with equal–sized holes, *Aequat. Math.* 33 (1987), 96–105.

[87] D.R. Stinson and L. Zhu, On the existence of three MOLS with equal–sized holes, *Austral. J. Combin.* 4 (1991), 33–47.

[88] K. Szajowski, The number of orthogonal latin squares, *Applicationes Math.* 15 (1976), 85–102.

[89] G. Tarry, Le problème de 36 officiers, *Ass. Franc. Av. Sci.* 29 (1900), 170–203.

[90] D.T. Todorov, Three mutually orthogonal latin squares of order 14, *Ars Combinatoria* 20 (1985), 45–48.

[91] D.T. Todorov, Four mutually orthogonal latin squares of order 20, *Ars Combinatoria* 27 (1989), 63–65.

[92] W.D. Wallis, Three orthogonal latin squares, *Congressus Num.* 42 (1984), 69–86.

[93] W.D. Wallis and L. Zhu, Orthogonal latin squares with small subsquares, *Lect. Notes Math.* 1036 (1983), 398–409.

[94] S.P. Wang, *On self-orthogonal latin squares and partial transversals of latin squares*, Ph.D. thesis, Ohio State University, 1978.

[95] R.M. Wilson, A few more squares, *Congressus Num.* 10 (1974), 675–680.

[96] R.M. Wilson, Concerning the number of mutually orthogonal latin squares, *Discrete Math.* 9 (1974), 181–198.

[97] R.M. Wilson, Constructions and uses of pairwise balanced designs, in: *Combinatorics*, Math. Centrum Amsterdam Tracts 55 (1974), 18–41.

[98] M. Wojtas, On seven mutually orthogonal latin squares, *Discrete Math.* 20 (1977), 193–201.

[99] M. Wojtas, The construction of mutually orthogonal latin squares, Kommunikat 172, Inst. Math. Politechniki Wroclaw, 1978.

[100] M. Wojtas, A note on mutually orthogonal latin squares, Kommunikat 236, Inst. Math. Politechniki Wroclaw, 1978.

[101] M. Wojtas, New Wilson–type constructions of mutually orthogonal latin squares II, Inst. Math. Politechniki Wroclaw, 1979.

[102] M. Wojtas, New Wilson–type constructions of mutually orthogonal latin squares, *Discrete Math.* 32 (1980), 191–199.

[103] M. Wojtas, Some new matrices–minus–diagonal and MOLS, *Discrete Math.* 76 (1989), 291–292.

[104] M. Wojtas, Five mutually orthogonal latin squares of orders 24 and 40, preprint.

[105] K. Yamamoto, Generation principles of latin squares, *Bull. Inst. Internat. Stat.* 38 (1961), 73–76.

[106] X. Zhang and H. Zhang, Three mutually orthogonal idempotent latin squares of order 18, preprint.

[107] L. Zhu, A short disproof of Euler's conjecture concerning orthogonal latin squares, *Ars Combinatoria* 14 (1982), 47–55.

[108] L. Zhu, Pairwise orthogonal latin squares with orthogonal small subsquares, Research Report CORR 83–19, University of Waterloo, 1983.

[109] L. Zhu, Some results on orthogonal latin squares with orthogonal subsquares, *Util. Math.* 25 (1984), 241–248.

[110] L. Zhu, Orthogonal latin squares with subsquares, *Discrete Math.* 48 (1984), 315–321.

[111] L. Zhu, Six pairwise orthogonal latin squares of order 69, *J. Austral. Math. Soc.* A37 (1984), 1–3.

[112] L. Zhu, Incomplete transversal designs with block size five, *Congressus Num.* 69 (1989), 13–20.

The following references are sections of the *CRC Handbook of Combinatorial Designs*, (C.J. Colbourn and J.H. Dinitz, editors), CRC Press, Boca Raton FL, 1996.

[113] R.J.R. Abel, Difference families.

[114] R.J.R. Abel, A.E. Brouwer, C.J. Colbourn and J.H. Dinitz, Mutually orthogonal latin squares.

[115] R.J.R. Abel, C.J. Colbourn and J.H. Dinitz, Incomplete MOLS.

[116] R.J.R. Abel and S.C. Furino, Resolvable and near–resolvable designs.

[117] R.A. Mathon and A. Rosa, Balanced incomplete block designs.

6

ON WRITING ISOMORPHISM PROGRAMS

William Kocay

Department of Computer Science,
University of Manitoba,
Winnipeg, Manitoba, Canada R3T 2N2
bkocay@cs.umanitoba.ca

ABSTRACT

This is a self-contained exposition on how to write isomorphism programs. It is intended for people who want to write isomorphism programs for combinatorial structures, such as graphs, designs, digraphs, posets, etc.

1 INTRODUCTION

This is an expository article aimed at graduate students or advanced undergraduates who want to write isomorphism programs, whether for graphs, designs, set systems, posets, or other combinatorial structures. The emphasis is on programming techniques. The program presented here is based on graphs, but the principles of the algorithms for other combinatorial structures are very similar. The most powerful general purpose graph isomorphism program currently available is without doubt B.D. McKay's C-language program *Nauty* [23][1]. See McKay [24] for a mathematical description of the algorithm used. The prototype of Nauty was the Fortran program GLABC, which formed a part of McKay's Ph.D. thesis [22]. Several other authors [10,19,21] have independently written general purpose graph isomorphism programs, including Kocay, whose *Groups & Graphs* [18][2] package contains a highly efficient Pascal graph isomorphism program.

This work was supported by an operating grant from the
Natural Sciences and Engineering Research Council of Canada.

[1]Nauty is available via anonymous ftp from dcssoft.anu.edu.au, in /pub/nauty19.

[2]Groups & Graphs is available at ftp.cc.umanitoba.ca in the directory /pub/mac.

W. D. Wallis (ed.), Computational and Constructive Design Theory, 135–175.
© 1996 *Kluwer Academic Publishers.*

These programs are different implementations of very similar algorithms. To the best of my knowledge, all existing general purpose graph isomorphism programs are based on the method of partition refinement. This is the most efficient method known to date. These algorithms are currently exponential in the worst case, but in practice are extremely efficient for most graphs. It may be possible to develop them into general polynomial algorithms. A polynomial algorithm for all graphs seems a very real possibility. One reason for this article is to make the algorithm and programming techniques more accessible to a wide audience. There is ample room for research into graph isomorphism algorithms. As will be seen, the programs are quite complicated and subtle. This is something of an obstacle, but not an insurmountable one.

In 1980, Furst, Hopcroft and Lux [12] discovered a sub-exponential algorithm for computing the automorphism group of a trivalent graph. It was later refined and improved by Galil, Hoffman, Lux, Schnorr, and Weber [14] into a $O(n^3 \log n)$ algorithm. This is a very efficient algorithm, based on finding block systems in a 2-group. It does not extend very well to graphs of higher valence (see Luks [20] and Hoffmann [16]). It seems to me that partition refinement offers more hope for a general polynomial-time algorithm. There are also a number of methods based on the linear algebra of the adjacency matrix (eg., see Bennett and Edwards [4]). These often work well for showing that two graphs are not isomorphic, but do not work for all graphs. In [15], Gismondi and Swart attempted to transform the graph isomorphism problem into a linear program and use the simplex method. The technique did not work, because it could not guarantee integral solutions, but uses some very interesting ideas.

There is a vast literature on graph isomorphism. We do not attempt to survey it here. For more information, see the bibliographies in [11] and [3]. A great many algorithms for graph isomorphism have been presented, mostly based on partition refinement. To the best of my knowledge, *Nauty* out-performs them all. This is an indication of the power of partition refinement and the importance of programming techniques. Nevertheless, even *Nauty* takes exponential time on certain difficult graphs. Development of a polynomial algorithm will require some new ideas and techniques. Corneil and Goldberg have presented a non-factorial algorthm in [9]. They use partition refinement with *sections* in order to prove that the algorithm is sub-factorial. Since most graphs don't seem to have sections, this means that existing algorithms based on partition refinement are already sub-factorial. It may be possible that the notion of sections can be extended to give a faster isomorphism algorithm.

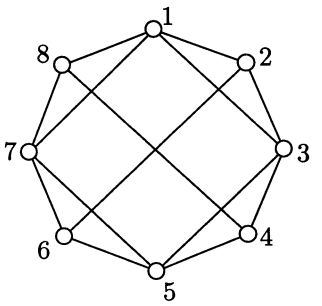
2 PARTITION REFINEMENT

Let G be a graph. We assume that G is undirected, with no loops or multiple edges, but the algorithm can be fairly easily extended to apply to directed graphs and/or non-simple graphs and/or designs. The vertex and edge sets of G are $V(G)$ and $E(G)$. Each edge is an unordered pair of vertices. If $u, v \in V(G)$, we write the pair $\{u, v\}$ as uv. We write $u \rightarrow v$ to indica te that u is adjacent to v. Since G is undirected we also have $v \rightarrow u$. This can also be expressed as $uv \in E(G)$.

The graph isomorphism program we are describing produces a *certificate* for G, written cert(G). Two graphs G and H are isomorphic if and only if they have equal certificates, cert(G) = cert(H) (see Read and Corneil [26]). One way of defining a certificate (the commonest way) is this. Let G have n vertices. $A(G)$, the adjacency matrix of G, is a symmetric matrix with a diagonal of zeroes. Changing the ordering of the rows and columns will change the matrix $A(G)$. The upper triangle contains $\binom{n}{2}$ bits which can be written as a single binary number, row after row, or column after column. Each ordering of $V(G)$ defines a bit string in this way. Fig. 1 shows a graph G and an adjacency matrix $A(G)$. The bit string defined by taking the upper triangle of $A(G)$ column by column is then 111001001101001100011100l001. It is more concise to interpret these bit strings as character strings, by grouping them 6 or more bits at a time.

These bit strings can be ordered lexicographically, and the *smallest* (or largest) can be taken as cert(G). We say that cert(G) corresponds to the *smallest adjacency matrix* for G. When defined in this way, cert(G) is obviously independent of the original ordering of the vertices. The disadvantage is that there are $n!$ different orderings of $V(G)$. There are several ways of reducing this to something more manageable. We describe one way. If G contains vertices of different degree, we can first sort the vertices by degree, and consider only those orderings of $V(G)$ which preserve the degree of the vertices. This idea can be extended as follows.

An *ordered partition* Π of $V(G)$ is a list of *cells*, $\Pi = (C_1, C_2, \ldots, C_p)$, where each cell C_i is a set containing one or more vertices, $\cup_i C_i = V(G)$, and $C_i \cap C_j = \emptyset$, for $i \neq j$. The degree sequence of G defines an ordered partition of G. A partition Π is said to be *equitable* or *stable* if whenever $u, v \in C_i$, u and v are both joined to the same number of vertices of C_j, for all $j = 1, 2, \ldots, p$. If Π is not stable, then we can *refine* it, as defined by



*	1	1	0	0	0	1	1
1	*	1	0	0	1	0	0
1	1	*	1	1	0	0	0
0	0	1	*	1	0	0	1
0	0	1	1	*	1	1	0
0	1	0	0	1	*	1	0
1	0	0	0	1	1	*	1
1	0	0	1	0	0	1	*

Figure 1 A graph and its adjacency matrix

the following algorithm. We assume that there is a global array *Degree*[·], such that *Degree*[v] is a cell invariant for Π. Initially we can assume that

Degree[v] = 0, for all $v = 1, 2, \ldots, n$.

```
Refine(Π: partition)
{Π is an ordered list of cells}
{each cell of Π is initially marked uncounted}
Begin
    C := first cell of Π
    repeat
        {first count the adjacencies to cell C}
        for each u ∈ C do
            for each v → u do
                Degree[v] := Degree[v] + 1
        mark C counted
        for each cell C' do begin
            sort the vertices of C' by Degree[·]
            if Degree[·] is not constant on C' then
                split C' into new cells of equal degree
                all new cells created are marked uncounted
        end
        C := an uncounted cell of Π
    until Π is stable
End  {Refine}
```

For example, let us start with the *unit* partition Π_0 for the graph of Fig. 1. This is the unique partition with a single cell $C_0 = \{1, 2, 3, 4, 5, 6, 7, 8\}$. The *Refine* algorithm first takes $C := C_0$ and counts the cell C. This has the effect of computing the degree of each vertex of G. The vertices of C are then sorted by degree. C contains vertices of degree 3 and 4. The next step sorts C and splits it into two cells, $C_1 = \{2, 4, 6, 8\}$ of degree 3, and $C_2 = \{1, 3, 5, 7\}$ of degree 4. So we now have $\Pi_1 = (C_1, C_2)$, where every $v \in C_1$ has $Degree[v] = 3$ and every $v \in C_2$ has $Degree[v] = 4$. In order to determine that Π_1 is in fact equitable, the repeat loop runs two more iterations. It first takes $C := C_1$ and counts this cell. Each $v \in C_1$ is adjacent to one vertex of C_1 and two vertices of C_2. Since $Degree[\cdot]$ is accumulative, the stored degrees of the vertices of C_1 will increase from 3 to 4, and those of C_2 will increase from 4 to 6. Since C_1 and C_2 still have constant degrees, there is no splitting of the cells at this point. C_1 is marked *counted*. The algorithm then takes $C := C_2$ and counts it. Each $v \in C_2$ is adjacent to two vertices of C_1 and two vertices of C_2. The stored degrees of the vertices of C_1 will increase from 4 to 6, and those of C_2 will increase from 6 to 8. Again there is no splitting of any of the cells. C_2 is now marked counted. Since every cell of Π_1 is counted, the algorithm deduces that the partition is stable, and the repeat loop terminates with an equitable partition. Thus an equitable partition is stable with respect to refinement.

The algorithm chooses C as the first uncounted cell of Π. It is obvious that with a specific choice like this, the equitable partition constructed will be independent of the original ordering of the vertices. In general, many ordered partitions correspond to the same unordered partition. If the cells C chosen to be counted are selected by a different rule, a different ordered partition will result. However it will always correspond to the same unordered partition. See Mathon [21] for a proof. A sketch of the proof follows. The set of all unordered partitions of V forms a partially ordered set. Π is smaller than Π' if it is finer than Π'. The set of partitions has a unique minimum element, the discrete partition, and a unique maximum element, the unit partition containing only one cell. Given any two partitions Π and Π', there is a unique partition $\Pi \wedge \Pi'$ which is the largest partition smaller than both Π and Π'. It is obtained by forming all pairwise intersections of the cells of Π and Π'. There is also a unique partition $\Pi \vee \Pi'$ which is the smallest partition larger than both Π and Π'. It is obtained by taking the unions of intersecting cells of Π and Π'. See Fig. 2. If we formed a bipartite graph of the cells of Π versus the cells of Π', and joined intersecting cells, then the cells of $\Pi \vee \Pi'$ would correspond to the connected components

of this graph. The cells of $\Pi \wedge \Pi'$ would correspond to the edges of this bipartite graph.

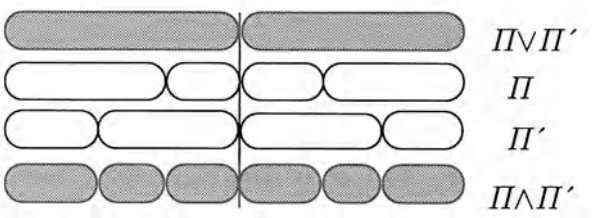


Figure 2 Partitions form an ordered set

Lemma 2.1 *Suppose that* Π *and* Π' *are equitable. Then so is* $\Pi \vee \Pi'$.

This can be proved by counting adjacencies using the diagram above. Note however, that $\Pi \wedge \Pi'$ is not always equitable. See McKay [25]. Now consider the refinement procedure in which a non-equitable partition Π is to be refined. A cell C of Π is selected and counted. This causes some of the cells of Π to split into two or more cells. Let Π_C be the equitable unordered partition obtained after refining. If the refinement were begun with another cell C' instead of C, an equitable unordered partition $\Pi_{C'}$ would be obtained. If $\Pi_C \neq \Pi_{C'}$, then $\Pi_C \vee \Pi_{C'}$ would be an equitable partition. Since $\Pi_C \leq \Pi$ and $\Pi_{C'} \leq \Pi$, it follows that $\Pi_C \vee \Pi_{C'} \leq \Pi$. But this is impossible. Thus for any unordered partition Π there is a unique largest equitable unordered partition Π_e that is smaller than Π. There are a number of equitable ordered partitions that correspond to Π_e. The refinement algorithm will construct one of them. It must choose the cell C to count by the same rule each time, in order to ensure that the ordering of Π_e that it constructs is unique.

The termination of the loop "repeat ... until Π is stable" requires that the program can recognize when a partition is stable. This will occur if every cell has been counted, and no further splitting of any cell has taken place. Whenever a new cell is created by splitting C', the new cells are marked uncounted. The program selects the cell C to count as an uncounted cell of Π. Consequently every cell will eventually be counted, and the loop will terminate only when no further splitting of cells has occurred. Therefore the repeat loop will terminate only when Π is equitable. So one way to recognize an equitable partition is to continue until every cell has been

counted. However one must keep in mind that eventually the isomorphism algorithm will construct a *discrete* partition of $V(G)$, that is, a partition whose every cell contains only one vertex. A discrete cell cannot split any further. Therefore *it is a good idea to remove discrete cells from the partition after they have been counted*, to avoid duplication of work. We will say more about discrete cells later. It is also possible that when a cell C' is split, that it splits into discrete cells. The entire partition Π can become discrete at such a step.

When this happens, there is no point in continuing to count the uncounted cells of Π, for a discrete partition must be equitable. So the criteria for detecting when Π is stable are:

1. if there is no uncounted cell C, then Π is stable;
2. if Π is discrete, then it is stable.

The first condition is very easy to detect. The second condition is not so easy. When the cell C' is being sorted according to $Degree[v]$, where $v \in C'$, the sorting algorithm can detect if all degrees of C' are distinct. If so, it can mark C' *discrete*, instead of actually splitting C' into new cells. A counter of the total number of vertices in discrete cells can be maintained. When it equals the number of vertices, we know that Π is discrete.

Data Structures

The *Refine* algorithm is tricky to program. The actual program will depend on the data structures chosen. In *Groups & Graphs*, I have used the following data structures. A graph is stored as an array of linked lists. For each vertex u, $Graph[u]$ denotes the list of vertices adjacent to u. This allows the loop "for all $v \rightarrow u$ do" to be programmed efficiently. An adjacency matrix is also stored, since it is needed for comparing different orderings of $V(G)$ when finding cert(G). The current ordering of the vertices is stored by an array $P[1..n]$. I have represented an ordered partition as a linked list of cells, where each cell is a record as follows.

```
CellPtr = ^Cell
Cell = record
          FirstPt, LastPt: Integer
          Counted, Discrete: Boolean
          NextCell: CellPtr
end
```

FirstPt and *LastPt* are pointers into the array $P[\cdot]$. The vertices in each cell are stored as contiguous entries in the array. *Counted* indicates whether the cell has been counted, and *Disrete* indicates whether a cell is composed of one or more discrete cells. *NextCell* is a pointer to the next cell in the partition.

This is a very convenient way to represent graphs and partitions, but it does have certain disadvantages. The main one is that a typical graph isomorphism calculation computes many thousands of partition refinements, and the constant creation and deletion of cells requires a great deal of memory management, which can be slow. Partition refinement is the main operation which takes place in graph isomorphism. Most of the time required is consumed by partition refinement. Therefore it must be made as fast as possible. McKay [22,24] has stored a graph as an adjacency matrix in such a way that each row is a packed bit-vector. This means that counting the cells in a partition can be accomplished using full-word boolean operations on the rows of the adjacency matrix. This counts all vertices adjacent to a vertex u in a constant number of steps (up to a maximum size). This gives a significant increase in speed. A difficulty which then arises is that it becomes more difficult to compare two orderings, since the adjacency matrix has been packed into bit vectors. On a parallel computer, partition refinement could take place much faster.

The algorithm *Refine*(Π) accepts an ordered partition Π and transforms it into an equitable ordered partition Π'. Viewed as unordered partitions, Π' is the largest equitable partition which is less than or equal to Π. The program assumes that initially *Degree*$[v]$ is a cell-invariant of Π (*Degree*$[v] = 0$ the first time *Refine* is called). When it terminates, Π has been transformed into an equitable partition Π', so that *Degree*$[v]$ will still be a cell-invariant of Π'. This feature is significant, because it means that the *program does not need to re-intiallize Degree*$[v]$ *to 0*. This saves an enormous amount of execution time.

Complexity

Suppose that a cell $C \in \Pi$ is to be counted, where $|C| = m$. If we assume that the graph is stored as adjacency lists, the number of steps required to count C is proportional to the sum of the degrees of the vertices of C. All cells in Π must then be sorted according to the array *Degree*$[\cdot]$. In theory a radix sort could be used to sort the cell in time $O(m)$, since the degrees are in the range $1..m$. However in practice this is not very useful, since it requires setting up m buckets, one for the vertices of each degree.

After the vertices have been placed in buckets, the buckets must be linked together, and this requires m steps, even if many of the buckets are empty. For example, m could be around 100, and there may be only 3 different degrees. Nevertheless, all 100 buckets would have to be created. Every vertex would have to be moved twice, once into a bucket, and once into its proper position. All 100 buckets would have to be tested to see if they are non-empty.

When a partition becomes equitable, all vertices in it have the same degree. It is extremely helpful to have a sort algorithm that only makes one pass through the vertices if they are already in order, leaving them unmoved. I have found that a simple insertion sort works very well for most graphs up to reasonable size. For larger graphs the quadratic complexity of the insertion sort starts to become noticeable. A very useful technique is to use a quicksort, with a cut-off M of around 16 to 32. When the number of vertices to be sorted is less than M, the quicksort procedure uses an insertion sort to order the vertices. When the number is M or more, a pivot is found and a recursive call is made. See Weiss [27] for an excellent implementation of quicksort.

Quicksort has an average complexity of $O(n \log n)$. Each time a cell is split the number of cells increases by at least one. If the number of vertices is n, the maximum number of splittings that can take place is n, giving a worst-case average complexity of $O(n^2 \log n)$ for the sorting stages of *Refine*. (The radix sort gives a theoretical worst-case complexity of $O(n^2)$, but it is not effective in practice.) If the graph has ε edges, the number of steps needed to count a cell C is at most $O(\varepsilon)$. If Π is already equitable, the total number of steps required to detect this is 2ε, by counting every cell. Since there are at most n splittings, the total number of steps used in counting cells is at most $O(n\varepsilon)$. Thus refinement is fairly efficient. As mentioned above, it is the single most time-consuming part of the algorithm. In order to make a polynomial graph isomorphism algorithm, a way must be found to reduce the number of discrete partitions to a polynomial number.

The Automorphism Group

The automorphism group of G consists of the set of all permutations of $V(G)$ that leave $E(G)$ invariant set-wise. It is denoted $\mathrm{Aut}(G)$. If $\pi, \tau \in \mathrm{Aut}(G)$, the image of vertex v under π is denoted v^π. Permutations are composed from left to right, so that $v^{\pi\tau}$ indicates the image of v under the product $\pi\tau$ (first π, then τ). A cell C of a partition Π is *fixed* by $\mathrm{Aut}(G)$ if $C^\pi = C$, for every $\pi \in \mathrm{Aut}(G)$, that is, π maps every element

of C to an element of C. If $\Pi = (C_1, C_2, \ldots, C_p)$, then Π^π denotes the ordered partition whose cells are $(C_1^\pi, C_2^\pi, \ldots, C_p^\pi)$. The connection between partition refinement and the automorphism group is the following.

Lemma 2.2 *Let Π_0 be a partition of $V(G)$ such that $\mathrm{Aut}(G)$ fixes every cell of Π_0. Let Π be obtained from Π_0 by refinement. Then $\mathrm{Aut}(G)$ fixes every cell of Π.*

Proof. The proof is by induction on the number of iterations of the repeat loop. Initially all automorphisms fix every cell of Π_0. On each iteration a cell C is counted. The vertices of each cell C' are then sorted according to the number of adjacencies they have to C, and C' may be split into two or more new cells. If $\pi \in \mathrm{Aut}(G)$ maps $u \in C$ to $v \in C$, then π also maps the vertices adjacent to u to the vertices adjacent to v. So π fixes every new cell split from C'. It follows that π fixes every cell of Π.

Since the unit partition is always fixed by $\mathrm{Aut}(G)$, for every graph G, we can always take Π_0 to be the unit partition initially, and refine it to an equitable partition Π. If H is any graph isomorphic to G and the same refinement procedure is applied to H, then any isomorphism from G to H must map Π, cell by cell, to the corresponding partition of H. The number of such mappings is no longer $n!$, but $\prod_{i=1}^{p} |C_i|!$, where $\Pi = (C_1, \ldots, C_p)$. A Π-*ordering* of G is any permutation of $V(G)$ that fixes every cell of Π setwise. The number of Π-orderings is $\prod_{i=1}^{p} |C_i|!$. We could then define $\mathrm{cert}(G)$ as the smallest adjacency matrix over the set of Π-orderings of $V(G)$. For example, if we compute $\mathit{Refine}(\Pi_0)$ for the graph of Fig. 3, we find that the resulting equitable partition is discrete. This uniquely defines the certificate for G.

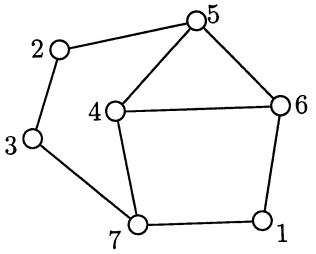


Figure 3 A graph with discrete partition

3 STABILISING VERTICES

If G is a regular graph, then the unit partition is already equitable, and refinement will produce no change in it. A number of recursive techniques can be used to reduce the number of orderings of $V(G)$ that must be considered in order to compute a certificate. The most common is vertex-stabilisation. Let Π be a non-discrete equitable partition of G. We can assume that Π has no discrete cells. Let Γ be a subgroup of $\mathrm{Aut}(G)$ such that Γ fixes Π. If C is any cell of Π, and $u \in C$ is any vertex of C, then in some of the Π-orderings of $V(G)$, u will be the first vertex of C. In order to focus on those orderings in which u is first, we split C into two cells, $\{u\}$ and $C - u$. Call the resulting partition Π'_u. Γ no longer fixes Π'_u, but the subgroup $\Gamma_u = \{\gamma \in \Gamma \mid u^\gamma = u\}$ that fixes u must also fix $C - u$ and all other cells of Π'_u.

3.1 Let Γ be a permutation group acting on a set V. The subgroup Γ_u consisting of those permutations that fix u is called a *stabiliser* subgroup.

We now refine Π'_u to an equitable partition Π_u. By Lemma 2.2, Γ_u fixes Π_u. This process of fixing a vertex u in a partition Π and refining Π'_u to an equitable partition Π_u is called *vertex stabilisation*. In most graphs it produces a considerable refinement of Π.

3.2 Given a vertex $u \in C \in \Pi$, Π_u always denotes the equitable partition obtained by splitting C and refining the resulting partition.

The number of Π_u-orderings of V will be much less than the number of Π-orderings. Since some $u \in C$ must be first, we do this for each $u \in C$, and then apply this idea recursively. This gives the following skeleton of a recursive procedure. The algorithm saves the ordering of V that gives the smallest adjacency matrix so far. This is called the best ordering.

```
Stabilise(Π: partition) {first version}
{refine Π to an equitable partition, with no discrete cells. If it is
  not discrete, fix a point from the first cell in all possible ways.}
Begin
    Refine(Π)
    if Π is discrete then begin
```

```
            Π defines an ordering of the vertices
            compare it with the best ordering found so far,
              replacing the best if necessary
            return
        end
        {otherwise Π is not discrete}
        C := first cell of Π
        for each u ∈ C do begin
            make a copy Π_u of Π in which C is split into
                {u} and C − {u}
            Stabilise(Π_u)
            Dispose(Π_u)
            {since some vertex of C must be first, and we have tested
                all vertices of C, at this point we have cert(G)}
        end
    End  {Stabilise}
```

The calling program begins by setting up a unit partition of V and calling *Stabilise*(Π). The recursion defines a search tree. A *leaf* in a search tree is a node with no descendants. Each leaf of the search tree corresponds to a discrete partition of V. Each discrete partition defines an ordering of V. The certificate of G is now *defined* as the smallest adjacency matrix with respect to this restricted set of orderings. Any ordering that gives the smallest adjacency matrix is called a *canonical* ordering. The program maintains a global array $B[1..n]$ of the best ordering of V found so far. This is the ordering that gives the smallest adjacency matrix. When the program terminates B will be a canonical ordering. It also maintains a global array $V[1..n]$ of the current ordering of the vertices, as constructed by partition refinement. At this point we need to say a word about the discrete cells produced by refinement. When *Refine*(Π) is executed, one or more discrete cells may be produced. Once a discrete cell has been counted, we remove it from Π, since a discrete cell cannot split any further. We place these deleted vertices on a third global array $F[1..n]$ of *fixed* points, *in the order in which they are encountered.* It may be that 5 or 6 points become fixed during a refinement. The number of fixed points currently on F is stored in a variable *NFixed*. We use the array F to compare with B. We compare the entries in the upper triangle of $A(G)$ as shown below. Comparing two orderings is quite time consuming, since it requires a lot of access to the 2-dimensional array A. The function compares F and B between subscripts

i and j, which will be $i = 1$ and $j = n$ when orderings arising from discrete partitions are compared.

```
Function Compare(i, j: integer): integer
{compare the orderings of A defined by arrays F and B
  between entries i and j. Returns −1, 0, or 1 according
  as F is worse than, equivalent to, or better than B}
begin
    if i = 1 then i := 2   {nothing to compare in column 1}
    for c := i to j do   {column c}
        for r := 1 to c − 1 do begin   {row r}
            if A[F[r], F[c]] < A[B[r], B[c]] then return(1)
            if A[F[r], F[c]] > A[B[r], B[c]] then return(−1)
        end
    return(0)
end   {Compare}
```

We now define cert(G) as the smallest adjacency matrix with respect to the orderings produced by *Stabilise*(Π). For most graphs there will be much less than $n!$ orderings. In fact, in a probabilistic sense, for "almost all" graphs, the partition obtained by refining the unit partition will be discrete [2]. However, most interesting graphs require more work.

Let us consider the complete graph K_n. All the partitions obtained by fixing a vertex are equitable. Refinement produces no improvement. The depth of the recursion will be n, corresponding to 0, 1, 2, ..., $n-1$ vertices being fixed. Clearly the search tree contains $n!$ leaf nodes; but they obviously all give the same certificate, so it is not really necessary to search them all. (Actually K_n can easily be detected since it has $\binom{n}{2}$ edges.)

3.3 If π is a permutation of V, A^π denotes the adjacency matrix obtained from A by permuting the rows and columns by π. Two orderings π_1 and π_2 of V are called *equivalent* if $A^{\pi_1} = A^{\pi_2}$. Notice that in this case, $A^{\pi_1\pi_2^{-1}} = A$, so that $\pi_1\pi_2^{-1} \in \mathrm{Aut}(G)$.

Lemma 3.1 *The number of inequivalent orderings of V is $n!/|\mathrm{Aut}(G)|$.*

Proof. Two orderings π_1 and π_2 are equivalent if and only if $\pi_1\pi_2^{-1} \in \mathrm{Aut}(G)$. So each coset of $\mathrm{Aut}(G)$ in the symmetric group $\mathrm{Sym}(V)$ gives a set of equivalent orderings.

In particular, K_n has only one (inequivalent) ordering of the vertices. If G has no automorphisms, then there are $n!$ inequivalent orderings of V. Any two leaf nodes of the search tree are inequivalent. Notice that this does not mean that the search tree has $n!$ inequivalent leaf nodes. I don't think anyone has been able to accurately estimate the number of leaf nodes of the search tree. Corneil and Goldberg [9] have shown that the number is sub-factorial, but this is still a weak bound. A proof that there is a polynomial number would give a polynomial algorithm for graph isomorphism. However the behaviour of all current graph isomorphism algorithms on certain difficult graphs suggests that they are not polynomial in general. The difficult graphs are invariably those that have very few automorphisms, but a high degree of "regularity", by which I mean that they have a large number of equitable partitions – the cell-adjacencies are very regular in an equitable partition. In other words, the regularity makes a graph look as though it were symmetric, when fact it may not be. The search tree will then contain a very large number of inequivalent leaf nodes.

Permutation Groups

We will need a number of basic properties of permutation groups and stabilisers. They are listed below. They are easy to verify from first principles. Here Γ is a permutation group acting on V, π is any element of Γ, u is any element of V and $v = u^\pi$. The order of a group Γ is denoted $|\Gamma|$. See the books by Biggs and White [5] or Wielandt [28] for more information on permutation groups.

3.4 The coset $\Gamma_u\pi$ consists of all elements of Γ that map u to $v = u^\pi$.

3.5 The *orbit* of u is the set of all points that Γ maps u to, that is, $\mathrm{Orb}(u) = u^\Gamma = \{u^\gamma \mid \gamma \in \Gamma\}$. It follows from 3.4 that $|u^\Gamma|.|\Gamma_u| = |\Gamma|$.

3.6 The *conjugate* of a permutation $\gamma \in \Gamma$ by π is $\pi^{-1}\gamma\pi$, which is also denoted by γ^π. If γ maps u to w, then γ^π maps u^π to w^π.

3.7 The conjugate of the subgroup Γ_u is $\pi^{-1}\Gamma_u\pi = \Gamma_u^\pi = \Gamma_v$, that is, the conjugate of the stabiliser of u by π is the stabiliser of $v = u^\pi$.

3.8 If Γ_u fixes a partition Π, then Γ_u^π fixes Π^π.

Lemma 3.2 *Suppose that* $\Gamma \leq \mathrm{Aut}(G)$ *fixes* Π, *and let* Π_u *be obtained by partition refinement. If* π *is any element of* Γ, *let* $v = u^\pi$. *Then* $\Pi_v = \Pi_u^\pi$.

Proof. Let $\Pi = (C_1, C_2, \ldots, C_p)$. Then $\Pi^\pi = (C_1^\pi, C_2^\pi, \ldots, C_p^\pi)$. If $u \in C_i$, then $u^\pi \in C_i^\pi = C_i$. When Π_u is constructed by refinement, a number of adjacencies in the graph G are counted. Since π is an automorphism of G, it follows that if a vertex w is adjacent to k vertices of a cell C_j of Π, that w^π will also be adjacent to k vertices of $C_j^\pi = C_j$ in Π^π, and so forth. Thus at every step of the refinement process, the cells of Π_u can be mapped by π to those of Π_v. When the process terminates, $\Pi_u^\pi = \Pi_v$. Notice that Γ_u fixes Π_u and Γ_u^π fixes Π_u^π.

In general let $\Pi_0, \Pi_1, \ldots, \Pi_k$ be the sequence of partitions occurring in a path to a leaf node in the search tree. Π_0 is the initial equitable partition obtained by refining the unit partition. A vertex u_0 in Π_0 is fixed, and after refinement, Π_1 is obtained. Then u_1 in Π_1 is fixed, and so on, until the discrete partition Π_k is reached. We assume throughout that discrete cells are deleted as they are encountered, so that $\Pi_0, \Pi_1, \ldots, \Pi_{k-1}$ do not have any discrete cells. Only Π_k has discrete cells. Vertices $u_0, u_1, \ldots, u_{k-1}$ have all been fixed in succession. As outlined above, the program *Stabilise*(Π) always selects u in the *first* cell C of Π. The reason for this is that the first cell is very easy to find. The program does not have to do any work to select it. Other strategies are possible. We could select u in the smallest cell, or largest cell, or some other specific choice. This would perhaps make a smaller or larger search tree, but requires more work to select u. So let C_i denote the first cell of Π_i. Then $u_i \in C_i$, for $i = 0, 1, \ldots, k-1$. Let $\pi \in \mathrm{Aut}(G)$, and let $v_i = u_i^\pi$, for $i = 0, \ldots, k-1$. Then $\Pi_0^\pi = \Pi_0$. Fixing v_0 instead of u_0 gives the partition Π_1^π, by Lemma 3.10. Since $u_1 \in C_1$, we know that $v_1 \in C_1^\pi$, the first cell of Π_1^π. Fixing v_1 in Π_1^π gives the partition Π_2^π, and so on. Thus, fixing $u_0^\pi, u_1^\pi, \ldots, u_{k-1}^\pi$ gives a sequence $\Pi_0^\pi, \Pi_1^\pi, \ldots, \Pi_k^\pi$ of partitions, such that the discrete partition Π_k^π is equivalent to Π_k.

A sequence $u_0, u_1, \ldots, u_{k-1}$ of points fixed in order to produce a discrete partition Π_k is termed a *basis* for the ordering of V defined by Π_k. The search tree constructed by *Stabilise* can be defined as the set of all sequences $(u_0, u_1, \ldots, u_j)$ of points fixed in succession by the algorithm, with adjacencies determined by *descent*: the sequence $(u_0, \ldots, u_j, u_{j+1})$ decends

from $(u_0, \ldots, u_j)$. Each descent corresponds to the vertex fixed, in this case, u_{j+1}. The tree is rooted at the empty sequence (), corresponding to no points fixed. Each sequence $(u_0, \ldots, u_j)$ corresponds to the equitable partition Π_{j+1} constructed by fixing these points in succession. The root node corresponds to Π_0. Each leaf node corresponds to a discrete partition. The path from the root to a leaf node defines the basis for that ordering of V. Fig. 4 shows a graph whose search tree is illustrated in Fig. 7. Each node of the search tree contains the corresponding partition. In this example, the discrete cells of each partition have not been deleted. Each descent is labelled by the vertex fixed at that point in the algorithm.

The remarks of the preceding paragraph show that if $\pi \in \mathrm{Aut}(G)$, then π induces an automorphism of the search tree. We know that $\mathrm{Aut}(G)$ fixes Π_0. The algorithm selects u_0 in Π_0 and fixes it to get Π_1. The branch of the search tree descending from Π_0 on the edge labelled u_0 will be searched by *Stabilise*(Π_1). If u_0^π in Π_0 is fixed instead, to get Π_1^π, then the branch descending on the edge labelled u_0^π will be isomorphic to the branch just searched, by Lemma 3.10. So having fixed u_0, we need fix no other points u_0^π for any $\pi \in \mathrm{Aut}(G)$. Write $\Gamma_0 = \mathrm{Aut}(G)$. Let Γ_1 denote the stabiliser of u_0 in Γ_0. Then Γ_1 fixes Π_1. In general, let Γ_i denote the subgroup of Γ_{i-1} obtained by stabilising u_{i-1}. Then Γ_i fixes Π_i. The algorithm selects some $u_i \in C_i \in \Pi_i$ and fixes it. Let $\pi \in \Gamma_i$. The branch of the search tree descending from Π_i on the edge labelled u_i will be isomorphic to the branch descending on the edge labelled u_i^π. So having chosen $u_i \in C_i$, we need choose no other points u_i^π for any $\pi \in \Gamma_i$. That is, we need to fix *exactly one point from each orbit* of Γ_i on C_i in order to find the unique minimum adjacency matrix of G.

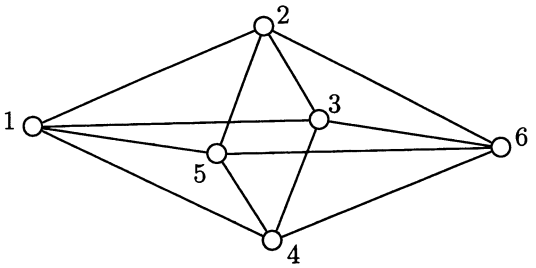


Figure 4 The graph of the octahedron

Initially the algorithm does not know $\mathrm{Aut}(G)$ or any of the stabiliser subgroups Γ_i. However as the program executes it will occasionally find a leaf node whose ordering of V is equivalent to the best ordering. By Lemma

3.4 this means that an automorphism of G has been discovered. The program maintains a data structure representing $\Gamma_0, \Gamma_1, \Gamma_2, \ldots$. Each time an automorphism π is discovered, the data structure is updated. This enables the program to avoid searching most equivalent branches of the tree. In the next section we describe the data structures used to represent the stabiliser subgroups.

4 THE SCHREIER-SIMS ALGORITHM

Given a set of permutations of a set V, we can make a *permutation diagram* for them. It is a directed graph in which the vertices are the elements of V. The edges are labelled by the permutations. If π is a permutation that maps u to $v = u^\pi$, then there is an edge from u to v labelled π. Fig. 5 shows a permutation diagram for $\pi = (1,6,5,2,4,3)$ and

$\tau = (1,4,3)(2,5)(6)$. These diagrams are the basis of many algorithms for groups.

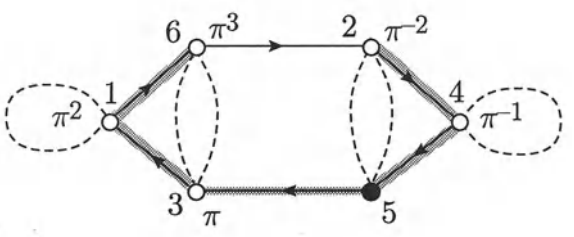


Figure 5 A permutation diagram

Let Γ denote the group generated by π and τ. Suppose that we are given π and τ and want to find Γ and a sequence of stabiliser subgroups. Γ consists of all products that can be formed from π, τ and their inverses. It is very easy to construct the permutation diagram for π and τ. Notice that each connected component of the permutation diagram represents an orbit of Γ. In the example above there is only one orbit. Suppose that we start at vertex 1 in the graph of Fig. 5 and follow the edges labelled $\pi\pi\tau\pi\tau\pi^{-1}$. We come to vertices 1,6,2,5,3,6,1, in that order. We conclude that the product $\pi\pi\tau\pi\tau\pi^{-1}$ maps 1 to 1, that is, it is an element of the stabiliser of 1 in Γ. Thus we have the following important observation.

4.1 Every walk in the permutation diagram corresponds to a product of the generators and/or their inverses. Given any starting point $u \in V$, every product of the generators and/or their inverses corresponds to a walk in the permutation diagram starting at u. If the walk ends at point $v \in V$, then u is mapped to v by this product.

Thus the stabiliser Γ_u of a point u corresponds to all walks that start and end at u. The walks that start at u and end at v correspond to all elements of Γ mapping u to v, that is, a right coset of Γ_u (see 3.5). Let us choose a spanning tree T of the graph, and pick any point u as the *root node* of T. Fig. 5 shows a spanning tree T in grey rooted at point 5. Now T contains a unique path from u to every point in V. These paths define a representative element of Γ mapping u to v, for every v. This gives a second important observation.

4.2 Every spanning tree T of a connected component X of the permutation diagram defines a decomposition of Γ into right cosets of Γ_u, where u is any point in X.

The spanning tree in Fig. 5 is rooted at point 5. The cosets of Γ_5, the stabiliser of point 5, are then $\Gamma_5\pi$, $\Gamma_5\pi^2$, $\Gamma_5\pi^3$, $\Gamma_5\pi^{-1}$, and $\Gamma_5\pi^{-2}$, as determined by T. If we could construct generators for Γ_u, then together with the coset representatives, this would completely determine Γ.

Let X denote a connected component of the permutation diagram with spanning tree T rooted at u. For every vertex v in X let T_v denote the word in the generators corresponding to the path in T from u to v. If vw is any edge not in T, then $T + vw$ contains a unique cycle, the *fundamental cycle* C_{vw} of vw with respect to T. Suppose that vw corresponds to a generator π. Then $T_v \pi T_w^{-1}$ is a walk in X that starts at u, travels to v, follows the edge vw to w, then travels along T_w^{-1} back to u. That is, it is a walk that starts at u, travels along a path in T to the fundamental cycle of vw, follows the cycle around, and then returns to u. See Fig. 6. In Fig. 5, for example, if the edge $(6,3)$ corresponding to the generator τ is used, the fundamental cycle is $(3,1,6)$ and the walk $T_6 \tau T_3^{-1}$ gives the sequence $(5,3,1,6,3,5)$ of vertices, which corresponds to the product $\pi\pi\pi\tau\pi^{-1}$ of the generators. Any edge vw not in T is called a *chord* of T.

It is a theorem of graph theory that every closed walk in a graph can be decomposed into fundamental cycles. See Bollobas [6] for a proof. We

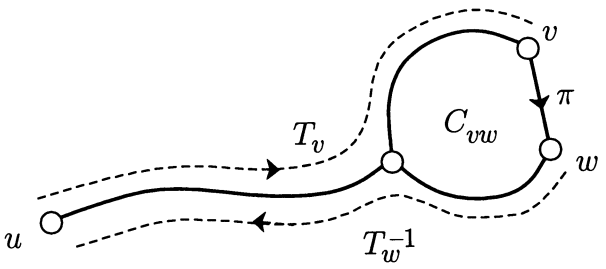


Figure 6 A fundamental cycle C_{vw}

sketch an outline of the proof here. Every cycle C in X must contain at least one edge not in T. Suppose that it contains edges $v_1w_1, v_2w_2, \ldots, v_kw_k$ not in T. If $k = 1$ then C is the fundamental cycle $C_{v_1w_1}$. $T_{v_1}v_1w_1T_{w_1}^{-1}$ is a walk that starts at u, travels to C, travels around C and then returns to u. If $k > 1$ we prove by induction that $T_{v_1}v_1w_1T_{w_1}^{-1} \ldots T_{v_k}v_kw_kT_{w_k}^{-1}$ is a closed walk that starts at u, travels to C, travels around C and returns to u. To complete the proof it is only necessary to show that every closed walk can be decomposed into cycles.

Since a walk that starts and ends at u is an element of the stabiliser Γ_u, we have another important observation. This result is of fundamental importance to many algorithms for groups.

4.3 Let T be a spanning tree of the permutation diagram, with root node u. Then Γ_u is generated by the products of the generators given by the walks $T_vvwT_w^{-1}$, where vw is any chord of T.

A consequence of 4.3 is that we now have a method of constructing a representation of any permutation group Γ for which generators are given. It is called the Schreier-Sims algorithm [7,8].

We store a permutation of V as an array of integers. The number of points in V will only be known when the program executes. Therefore we store pointers to permutations and allocate the arrays dynamically with the appropriate length. When storing a group, we need an array of coset representatives, that is, an array of permutations. Since we don't know the length of the array at compile-time, we store a pointer to an array of pointers. The generators of a group are stored as a linked list of permutations.

```
PermPtr = ^Perm
Perm = array[1..n] of integer
CosetPtr = ^CosetReps
CosetReps = array[1..n] of PermPtr
GenPtr = ^Gen
Gen = record
        GenPerm: PermPtr
        NextGen: GenPtr
    end
```

The data structure used to represent a permutation group Γ contains a linked list of generators, an orbit Orb(u), for some point $u \in V$, and a representative for each coset of Γ_u. It is also convenient to store the inverses of some of the coset representatives. The orbit is stored on an array, which is allocated dynamically as a PermPtr. The number of points in the orbit is stored as *NPts*. The data structure also contains the stabiliser Γ_u. Since Γ_u is also a group, we have a recursive data structure.

```
GroupPtr = ^Group
Group = record
        Generators: GenPtr   { linked list of generators }
        u: integer
        Orbit: PermPtr   { the orbit of u }
        NPts: integer   { number of points in Orb(u) }
        Cosets: CosetPtr   { coset reps of Γ_u }
        Inverses: CosetPtr
        Γ_u: GroupPtr   { stabiliser of u }
    end
```

Orbit^[k] is the k^{th} point in the orbit of u, for $k = 1, 2, \ldots,$ *NPts*. The first point in the orbit is *Orbit*^[1] $= u$. The coset representative for $v =$*Orbit*^[k] is *Cosets*^[v]. If v is a point not in the orbit of u, then *Cosets*^[v] $=$ *nil*. Given this representation of a group Γ acting on $V = \{1, 2, \ldots, n\}$, and any permutation γ on V we can easily determine whether $\gamma \in \Gamma$, as follows.

```
GroupElt(γ: PermPtr; Γ: GroupPtr): Boolean
{returns true if γ ∈ Γ}
var π: PermPtr
Begin
```

```
        for k := 1 to n do if γ^[k] ≠ k then goto 1
        return(true)   {γ ∈ Γ since γ is the identity}
    1: with Γ^ do
        begin
            v := γ^[u]
            if Cosets^[v] =nil then return(false)      {v ∉ Orb(u)}
            π := MultiplyPerm(γ, Inverses^[v])
                                        {π maps u to u}
            return(GroupElt(Γ_u, π))
        end
    End  {GroupElt}
```

The program first checks whether γ is the identity. We assume that the number of points, n, is a global variable. If γ is not the identity, $v = u^\gamma$ is found, and the program checks whether $v \in \mathrm{Orb}(u)$. If not, then $\gamma \notin \Gamma$. Otherwise let γ_v denote the coset representative stored for the point v ($\gamma_v =$ Cosets$^\wedge[v]$). The product $\pi = \gamma\gamma_v^{-1}$ is computed. Clearly π fixes u, so that $\gamma \in \Gamma$ if and only if $\pi \in \Gamma_u$. Therefore the program makes a recursive call. A function *MultiplyPerm* is needed in order to compute the product $\gamma\gamma_v^{-1}$. This assumes that the inverses of all coset representatives have been stored. If this is not the case, an additional statement is needed:

```
if Inverses^[v] = nil then Inverses^[v] := InversePerm(Cosets^[v])
```

where *InversePerm* is the following function.

```
InversePerm(γ: PermPtr): PermPtr
var τ: PermPtr
begin
    τ := NewPerm(n)   {allocate a new perm on n points}
    for k := 1 to n do τ^[γ^[k]] := k
    return(τ)
end   {InversePerm}
```

We are now in a position to give the code for the Schreier-Sims algorithm for constructing a representation of a permutation group Γ from its generators. The algorithm is based on a breadth first search to construct the permutation diagram for the generators of Γ. It builds a breadth-first spanning tree T which is used to define a set of coset representatives for the stabiliser

subgroup Γ_u. The chords of the spanning tree are used to construct generators for Γ_u. The main procedure *AddGen* takes an existing group Γ which is already stored. The first time it is called, Γ will have been intiallized to the identity group on n points. This is a *Group* record whose arrays have been allocated to have length n, but with no generators, and no point u selected. In an identity group the arrays *Cosets* and *Inverses* are allocated and every entry is intiallized to *nil*, and the stabiliser subgroup Γ_u is set to *nil*. The program takes a permutation γ which is a generator of Γ, but not yet represented in the data structure. Thus, $\gamma \notin \Gamma$ when the procedure is called. It updates the data structure so that upon completion γ is now recognized as a generator of Γ.

```
AddGen(γ: PermPtr; Γ: GroupPtr)
{add γ to the data structure representing Γ}
var π: PermPtr
Begin with Γ^ do begin
    if Γ_u = nil then begin
        {initialize Γ_u to the identity group on n points}
        Γ_u := IdentityGroup(n)
        {find a point u moved by γ}
        u := 1
        while γ^[u] = u do u := u + 1
    end
    add γ to the linked list Generators
    M := NPts   {current number of points in Orb(u)}
    k := 1
    while k ≤ M do begin
        v := Orbit^[k]   {k^th point in the orbit}
        w := γ^[v]
        if Cosets^[w] = nil then begin   {w ∉ Orb(u)}
            NPts := NPts + 1
            Orbit^[NPts] := w   {add w to Orb(u)}
            Cosets^[w] := MultiplyPerm(Cosets^[v], γ)
        end
        else begin   {w ∈ Orb(u)}
            if Inverses^[w] = nil then
                Inverses^[w] := InversePerm(Cosets^[w])
            π := GenMultiply(Cosets^[v], γ, Inverses^[w])
```

```
            if not GroupElt(π, Γ_u) then AddGen(π, Γ_u)
            else Dispose(π)   {π is not needed}
        end
        k := k + 1
    end
    {if γ has extended Orb(u), apply all generators to all new points}
    while k ≤ NPts do begin
        v := Orbit^[k]   {k^th point in the orbit}
        for each generator τ ∈ Generators do begin
            w := τ^[v]
            if Cosets^[w] = nil then begin   {w ∉ Orb(u)}
                NPts := NPts + 1
                Orbit^[NPts] := w   {add w to Orb(u)}
                Cosets^[w] := MultiplyPerm(Cosets^[v], τ)
            end
            else begin   {w ∈ Orb(u)}
                if Inverses^[w] = nil then
                    Inverses^[w] := InversePerm(Cosets^[w])
                π := GenMultiply(Cosets^[v], τ, Inverses^[w])
                if not GroupElt(π, Γ_u) then AddGen(π, Γ_u)
                else Dispose(π)   {π is not needed}
            end
            k := k + 1
        end
    end
end end   {AddGen}
```

The procedure first computes $w = v^\gamma$ for all $v \in \mathrm{Orb}(u)$. Let $\gamma_v = \mathrm{Cosets}^\wedge[v]$ be the coset representative for v. If $w \notin \mathrm{Orb}(u)$, then w is added to the orbit. $\gamma_v\gamma$ maps u to w, so it becomes γ_w, the coset representative for w. It is computed by *MultiplyPerm*. If $w \in \mathrm{Orb}(u)$, then $\pi = \gamma_v\gamma\gamma_w^{-1}$ maps u to u. Therefore it is a generator for the stabiliser Γ_u. *GenMultiply* is a procedure that performs this multiplication. This is more efficient than calling *MultiplyPerm* twice.

```
GenMultiply(α, β, γ: PermPtr): PermPtr
{compute αβγ}
var τ: PermPtr
```

```
begin
    τ := NewPerm(n)   {allocate a new perm on n points}
    for k := 1 to n do τ^[k] := γ^[β^[α^[k]]]
    return(τ)
end   {GenMultiply}
```

The program calls *GroupElt* to check if π is known to be in Γ_u. If so it is discarded. The storage used by it is reclaimed by *Dispose*, an operating system call. If π is not currently known to be in Γ_u then *AddGen* is called recursively. When it returns, the data structure representing Γ_u will have been modified to account for π. Once γ has been applied to all $v \in \mathrm{Orb}(u)$, the program applies all generators of Γ to all new points in the orbit. This builds the permutation diagram for the orbit containing u. When the program terminates, a coset representative has been stored for each $v \in \mathrm{Orb}(u)$, and the stabiliser Γ_u is up to date. Therefore the data structure for Γ is also up to date.

The sequence of points fixed in order to construct the tower of stabilisers is called the *basis* of the group. See Butler and Cannon [7] and Butler and Lam [8] for further information. The program *AddGen* constructs the permutation diagram using a breadth-first search. A number of variations on this algorithm are possible. See Kirk [17], Butler and Cannon [7] and Butler and Lam [8]. If there are k generators and n points in the orbit of u, the number of coset representatives stored will be n. The number of steps required to multiply two permutations of degree n is n. Therefore the number of steps required to build the diagram is proportional to kn^2, for one level in the recursion. The depth of the recursion will depend on the group Γ. The symmetric group requires depth n. The alternating group requires depth $n-1$. All other groups require much less, since any group which is 6-transitive or more must be either an alternating or symmetric group. The number of generators can always be taken to be at most $\log_2 |\Gamma|$, since each new generator must expand the tower by at least doubling the size of at least one group. Since $|\Gamma| \leq n!$, we can use Stirling's formula $n! \approx n^n e^{-n}\sqrt{2\pi n}$ to get a polynomial bound on the complexity of *AddGen*. However the only groups that reach the bound are the alternating and symmetric groups. These can often be detected and handled as special cases to give a much better complexity. A number of variants are also possible for the *AddGen* procedure such as randomized methods for building the group, or storing the coset representatives as words in the generators, rather than permutations.

5 GRAPH ISOMORPHISM

The procedure *AddGen* is called each time an automorphism is discovered in the search tree created by *Stabilise*. Notice that the automorphism group Γ is stored as a set of coset representatives plus a stabiliser subgroup. Together these create a tower of stabilisers. Each time an automorphism is discovered the effect is to expand the entire tower. Therefore the orbits will be updated for each group in the tower.

In section 3 we had a sequence of partitions $\Pi_0, \Pi_1, \ldots, \Pi_k$ and a sequence of vertices $u_0, u_1, \ldots, u_{k-1}$ such that u_i in Π_i was stabilised in order to obtain Π_{i+1}. A sequence of groups $\Gamma_0, \Gamma_1, \ldots, \Gamma_k$ is associated with the partitions. $\Gamma_0 = \mathrm{Aut}(G)$. Γ_{i+1} is the stabiliser of u_i in Γ_i, where $i = 0, 1, \ldots, k-1$. Γ_i fixes Π_i. Γ_k is the identity group. So the basis $u_0, u_1, \ldots, u_{k-1}$ of the discrete partition Π_k is also a basis for the group Γ_0. Associated with each node of the search tree is a partition Π_i and a group Γ_i. Associated with each descent from Π_i to Π_{i+1} is the vertex u_i. u_i is contained in the first cell C_i of Π_i. In order to select a vertex from each orbit of Γ_i acting on C_i we must store the orbits of Γ_i on C_i. They are stored in an array referenced by the pointer *CellOrbits*. We collect together in one data structure the various objects associated with a partition.

```
SearchNodePtr = ^SearchNode
SearchNode = record
    Partition: CellPtr   {a partition Π_i}
    FixedPt: Integer   {the point u_i fixed to get Π_{i+1}}
    ItsGroup: GroupPtr   {the group Γ_i}
    CellOrbits: PermPtr   {the orbits of Γ_i on C_i}
    Depth: Integer   {current depth of the search tree}
    NFixed: Integer   {the number of fixed points on the array F}
    OnBestPath: Boolean   {whether the node is on the best path}
    NextSearchNode: SearchNodePtr   {the next search node, descending}
  end
```

The Cell Orbits.

The orbits of Γ_i on C_i, the first cell of Π_i, are stored using the *union-find* data structure (see Aho, Hopcroft and Ullmann [1], Weiss [27]). *CellOrbits* points to an integer array of length $n = |V|$. Each orbit has a representative vertex. Two vertices are in the same orbit if and only if they have the

same representative. The value of *CellOrbits*^[v] is a pointer toward the representative, which in turn is marked by a negative value. We initiallize *CellOrbits*^[v] $= -1$, for all $v \in C_i$. The entries for $v \notin C_i$ are not used. The value of -1 indicates that each v is an orbit representative, and thus forms an orbit by itself. This is the situation initially when Γ_i is the identity group. When a generator γ of Γ_i is found, there will be a call to *AddGen*(γ, Γ_i). *AddGen* must update the known orbits of Γ_i on C_i. Thus it will contain a call to the following procedure, which can be placed immediately after the statement adding γ to the list of generators.

```
UpdateOrbits(γ: PermPtr, C: CellPtr)
{update the orbits of the stabiliser on cell C}
Begin with C^ do begin
    for k := FirstPt to LastPt do begin
        u := V[k]   {k^th vertex of the graph}
        v := γ^[u]
        uRep := OrbitRep(u)   {orbit representative for u}
        vRep := OrbitRep(v)   {orbit representative for v}
        if uRep ≠ vRep then Merge(uRep, vRep)
    end
end End   {UpdateOrbits}
```

In order to have access to the partition we must change the calling parameters of *AddGen* to include the current searchnode, which in turn contains the group.

AddGen(γ: PermPtr; S: SearchNodePtr)

The procedure *UpdateOrbits* computes $v = u^\gamma$ for all $u \in C$ and merges the orbits of u and v if they are different. Orbits are merged by reassigning the *CellOrbits* pointer for one of u and v. For efficiency, we merge the smaller orbit onto the larger (see [1]). The size of an orbit is given by the negative value stored in *CellOrbits* for the orbit representatives.

```
Merge(uRep, vRep: Integer)
{merge the orbits of uRep and vRep, which are orbit representatives}
Begin
    uSize := −CellOrbits^[uRep]
    vSize := −CellOrbits^[vRep]
    if uSize < vSize then begin
```

```
            CellOrbits^[uRep] := vRep
            w := vRep   {the new orbit representative}
        end
        else begin
            CellOrbits^[vRep] := uRep
            w := uRep   {the new orbit representative}
        end
        CellOrbits^[w] := -(uSize + vSize)
    End  {Merge}
```

The function *OrbitRep* follows the pointers *CellOrbits*[v] to the orbit representative.

```
    OrbitRep(v: Integer): Integer
    {follow the pointers to the orbit representative}
    var w: Integer
    Begin
        if CellOrbits^[v] < 0 then return(v)
        w := OrbitRep(CellOrbits^[v])
        CellOrbits^[v] := w   {path compression}
        return(w)
    End  {OrbitRep}
```

These procedures require access to the *CellOrbits* array. It can either be passed as a parameter, or else a global variable can be used to store a pointer to the current *CellOrbits* array.

Comparing Orderings.

As presented in section 3, the procedure *Stabilise* first refines Π, then selects in turn each $u \in C \in \Pi$, creates a partition Π_u in which u is fixed, and calls itself recursively. During the refinement of Π_u, any discrete cells are deleted. The vertices in discrete cells are placed on an array F of fixed points. At least one vertex, u, is moved to the array F during each refinement. The *Refine* procedure will also set the variable *NFixed*, the number of points currently on the array F. In order to have access to this variable, we must alter the calling parameters for *Refine* to include the current searchnode, which contains the partition to be refined.

Refine(S: SearchNodePtr)

The first time a discrete partition is obtained, the ordering of F is saved to an array B, the *best* ordering so far. B is a candidate for the canonical ordering of V. The program needs to know whether the ordering B has been intiallized yet. We use a global boolean variable *B_exists* to indicate this condition. It is initially set to *false*. In order to detect whether a refined partition is discrete, the *Refine* procedure can set a global boolean variable *isDiscrete*. The initial search node created containing the unit partition (before refinement) is saved as a global variable, called *TopSearchNode*. The *GroupPtr* that it contains points to Aut(G). When an automorphism is discovered by the program, it is added to the *TopSearchNode*. A refined version of the procedure *Stabilise* is now presented.

```
Stabilise(S: SearchNodePtr)  {refined version}
{S is the current search node, containing a partition Π. Refine Π,
      then select u in the first cell of Π in all inequivalent ways}
Begin
    m := S^.NFixed  {save the number of points currently fixed}
    S^.OnBestPath := false
    Refine(S)  {refine the partition in this search node}
    Result := CompareOrders(m + 1, S^.NFixed)
    if isDiscrete then begin
        if B_exists then begin
            case Result of
            equal: begin  {an automorphism has been found}
                γ := NewPerm(n)  {allocate a new PermPtr}
                for k := 1 to n do γ^[F[k]] := B[k]
                AddGen(γ, TopSearchNode)
                AutoFound := true
            end
            better: begin
                copy F[1..n] to B[1..n]
                set all OnBestPath values to true
            end
            worse: {ignore}
            end  {case}
        end
        else begin
            copy F[1..n] to B[1..n]
```

```
                B_exists := true
                set all OnBestPath values to true
            end
            goto 1
        end     {if isDiscrete}
        {otherwise the partition is not discrete}
        case Result of
            equal: {ignore}
            better: B_exists := false
            worse: goto 1
        end  {case}
        with S^ do begin
            Π := Partition
            if NextSearchNode=nil then create and initiallize
                NextSearchNode
            S_u := NextSearchNode
        end
        C := first cell of Π
        u := C^.FirstPt
        repeat  {until all inequivalent choices u have been made}
            S_u^.FixedPt := u
            make a copy Π_u of Π in which C is split into {u} and
                C - {u}
            S_u^.Partition := Π_u
            Stabilise(S_u)
            if AutoFound then ...
            dispose(Π_u)
            mark CellOrbits^[u] to indicate that u has been fixed
            change the base of Aut(G) if necessary
            select next inequivalent u ∈ C to stabilise
        until no u was found
1:      {reset degrees to 0 before returning}
        for k := m + 1 to S^.NFixed do Degree^[F[k]] := 0
        S^.FixedPt := 0   {mark this search node inactive}
    End {Stabilise}
```

The Global Variables.

We collect together here a brief summary of the global variables used by the program, and their purpose.

A: array, the adjacency matrix
V: array, the current ordering of the vertices
B: array, the best ordering of the vertices found so far
F: array, the array of points fixed by refinement
Graph[u]: linked list, the vertices adjacent to u
isDiscrete: Boolean, whether refinement produced a discrete partition
B_exists: Boolean, whether the array B has a value
AutoFound: Boolean, whether an automorphism was found
TopSearchNode: SearchNodePtr, the initial search node
LastBaseChange: SearchNodePtr, see below
BasisOK: Integer, the depth to which the bases agree

When an Automorphism is Discovered.

There are several subtle aspects of the algorithm which are best illustrated by an example. Let G be the graph of Fig. 4. See also the search tree of Fig. 7. The initial unit partition is

$\Pi_0 = \{1, 2, 3, 4, 5, 6\}$

The program will move the points of discrete cells to the array F. However we show the discrete cells included with each partition in this example. The cells of an ordered partition are shown separated by vertical bars.

Stabilise(Π_0) is called, Π_0 is refined
$\Pi_0 = \{1, 2, 3, 4, 5, 6\}$ is equitable
fix 1 :⇒ $\Pi_1 = \{1 \mid 2, 3, 4, 5, 6\}$
Stabilise(Π_1) is called, Π_1 is refined
$\Pi_1 = \{1 \mid 6 \mid 2, 3, 4, 5\}$ is equitable, $F = (1, 6)$
fix 2 :⇒ $\Pi_2 = \{1 \mid 6 \mid 2 \mid 3, 4, 5\}$
Stabilise(Π_2) is called, Π_2 is refined
$\Pi_2 = \{1 \mid 6 \mid 2 \mid 4 \mid 3, 5\}$ is equitable, $F = (1, 6, 2, 4)$
fix 3 :⇒ $\Pi_3 = \{1 \mid 6 \mid 2 \mid 4 \mid 3 \mid 5\}$
Stabilise(Π_3) is called, Π_3 is refined

Π_3 is discrete, $B = (1,6,2,4,3,5)$ is assigned
Backtrack to Π_2
fix 5 :⇒ $\Pi_3 = \{1 \mid 6 \mid 2 \mid 4 \mid 5 \mid 3\}$
Stabilise(Π_3) is called, Π_3 is refined
Π_3 is discrete, $F = (1,6,2,4,5,3)$
An automorphism $\gamma = (3,5)$ is discovered.
AddGen is called.
The orbits of each Γ_i on the first cell of Π_i are updated.
The orbits are indicated as follows.
$\Pi_0 = \{(1),(2),(3,5),(4),(6)\}$
$\Pi_1 = \{1 \mid 6 \mid (2),(3,5),(4)\}$
$\Pi_2 = \{1 \mid 6 \mid 2 \mid 4 \mid (3,5)\}$
Backtrack to Π_2
In Π_2 both 3 and 5 have already been fixed
⇒ backtrack to Π_1
In Π_1, only 2 has been fixed so far
fix 3 :⇒ $\Pi_2 = \{1 \mid 6 \mid 3 \mid 2,4,5\}$
Stabilise(Π_2) is called, Π_2 is refined
$\Pi_2 = \{1 \mid 6 \mid 3 \mid 5 \mid 2,4\}$ is equitable, $F = (1,6,3,5)$
fix 2 :⇒ $\Pi_3 = \{1 \mid 6 \mid 3 \mid 5 \mid 2 \mid 4\}$
Stabilise(Π_3) is called, Π_3 is refined
Π_3 is discrete, $F = (1,6,3,5,2,4)$
An automorphism $\gamma = (2,3)(4,5)$ is discovered.
AddGen is called.
The generators of $\mathrm{Aut}(G)$ are now $(3,5)$ and $(2,3)(4,5)$
The orbits of each Γ_i on the first cell of Π_i are updated.
$\Pi_0 = \{(1),(2,3,4,5),(6)\}$
$\Pi_1 = \{1 \mid 6 \mid (2,3,4,5)\}$
$\Pi_2 = \{1 \mid 6 \mid 3 \mid 5 \mid (2,4)\}$
Backtrack to Π_2
In Π_2 points 2 and 4 are in the same orbit
⇒ backtrack to Π_1
In Π_1 points $2,3,4,5$ are in the same orbit ⇒ backtrack to Π_0
fix 2 :⇒ $\Pi_1 = \{2 \mid 1,3,4,5,6\}$
Stabilise(Π_1) is called, Π_1 is refined
$\Pi_1 = \{2 \mid 4 \mid 1,3,5,6\}$ is equitable, $F = (2,4)$

fix 1 :⇒ $\Pi_2 = \{2 \mid 4 \mid 1 \mid 3,5,6\}$

Stabilise(Π_2) is called, Π_2 is refined

$\Pi_2 = \{2 \mid 4 \mid 1 \mid 6 \mid 3,5\}$ is equitable, $F = (2,4,1,6)$

fix 3 :⇒ $\Pi_3 = \{2 \mid 4 \mid 1 \mid 6 \mid 3 \mid 5\}$

Stabilise(Π_3) is called, Π_3 is refined

Π_3 is discrete, $F = (2,4,1,6,3,5)$

An automorphism $\gamma = (1,2)(4,6)$ is discovered.

AddGen is called.

The generators of $\mathrm{Aut}(G)$ are now

$(3,5), (2,3)(4,5)$, and $(1,2)(4,6)$

The orbits of each Γ_i on the first cell of Π_i are updated.

$\Pi_0 = \{(1,2,3,4,5,6)\}$

$\Pi_1 = \{2 \mid 4 \mid (1,3,5,6)\}$

$\Pi_2 = \{2 \mid 4 \mid 1 \mid 6 \mid (3,5)\}$

Backtrack to Π_2

In Π_2 points 3 and 5 are in the same orbit

⇒ backtrack to Π_1

In Π_1 points $1,3,5,6$ are in the same orbit ⇒ backtrack to Π_0

In Π_0 points $1,2,3,4,5,6$ are in the same orbit ⇒ done.

The program visited 4 leaf nodes in the search tree, as shown in Fig. 7. Three generators for $\mathrm{Aut}(G)$ were found. It is easy to see that these generators give all of $\mathrm{Aut}(G)$, for the following reason. If the orbits of Γ_i on C_i were not stored, the search algorithm would visit every leaf node of the search tree. Consider the first time an ordering B giving the minimum adjacency matrix was found. The number of leaf nodes with an equivalent ordering is $|\mathrm{Aut}(G)|$. Since every one of these would be visited by the search, every automorphism would be found. By computing the orbits of Γ_i on C_i, the program avoids visiting leaf nodes which it knows to be equivalent to a node already visited. Therefore it will find a representative of every coset of Γ_i in Γ_{i-1}. Consequently the generators found generate all of $\mathrm{Aut}(G)$. The group Γ_0 is initially the identity group, and grows until it equals $\mathrm{Aut}(G)$. Γ_0 is always a subgroup of $\mathrm{Aut}(G)$. So each time a new generator is discovered, Γ_0 must at least double in size. Therefore the number of generators found is at most $\log_2|\mathrm{Aut}(G)|$.

In the example above, all leaf nodes of the search tree are equivalent to each other. Each one gives an automorphism of G. For larger graphs, the

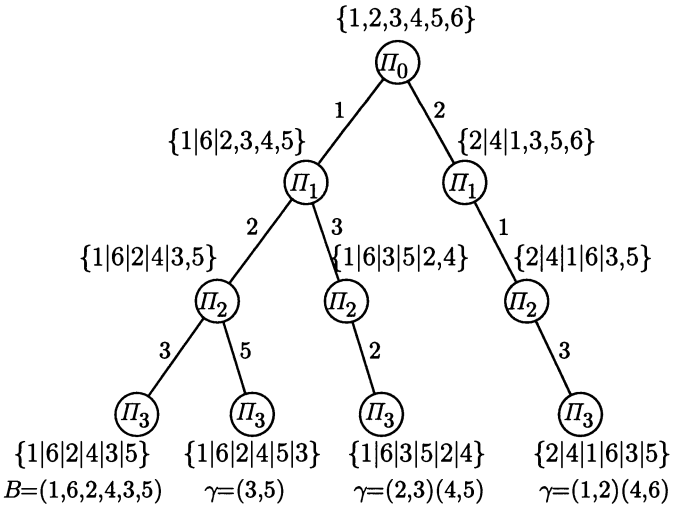


Figure 7 The search tree for the graph of Fig. 6

search tree does not have such a simple structure, even when G is highly symmetric. The ordering B can be set and reset many times. Consider the portion of a search tree shown in Fig. 8. The tree is drawn so that the leaf nodes are visited from left to right.

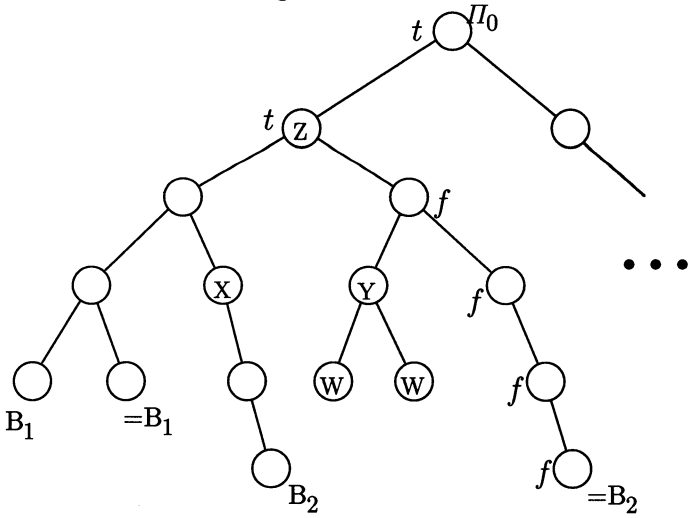


Figure 8 A portion of a search tree

The first discrete partition occurs at the node labelled B_1. The ordering B is assigned at this point. The next discrete partition gives an equivalent ordering, indicated by $=B_1$ in the diagram. At each level in the recursion the program compares the orderings B and F for the new points fixed at that level. This occurs in the statement "*CompareOrders*$(m+1, S\hat{}.\text{NFixed})$". It can occur that the new ordering F is already discovered to be better than the previous best ordering B. This is the situation at node X in the diagram. At this point the program sets the flag *B_exists* to *false*. When the next leaf node is reached, the array B is assigned, as if for the first time. This is at node B_2 in the diagram. Notice that the depth of the recursion when a leaf node is discovered can vary. The program then visits the two leaf nodes marked W and finds that the orderings there are worse than the current ordering B_2.

Notice that an automorphism can be missed! These orderings marked W could be equivalent to B_1, but the program does not notice it. In order to avoid this situation, we could store the best two orderings, or the best three, etc. However the more orderings we store, the longer it takes to do a comparison. So far as I know, *Nauty* stores two orderings, the first one found, and the best so far. The current version of *Groups & Graphs* stores only one best ordering.

We make another observation at this point. We have defined a canonical ordering as one which gives a minimum adjacency matrix over all leaf nodes of the search tree constructed by *Stabilise*. It is possible to use the shape of the search tree to further restrict the orderings considered as candidates for the canonical ordering. We could require that only leaf nodes at the minimum possible depth in the recursion are valid candidates. This would exclude the node B_2 in Fig. 8.

The program then proceeds to the node marked $=B_2$ where an ordering equivalent to B_2 is reached. An automorphism is discovered. This means that the current branch of the search tree being searched is equivalent to a branch previously searched. There is no need to continue searching the current branch. How is this detected by the program? The discrete partition at B_2 was obtained by fixing a sequence of vertices $u_0, u_1, u_2, u_3, \ldots$. The partition at $=B_2$ was obtained by fixing another sequence $v_0, v_1, v_2, v_3, \ldots$. These two sequences will usually have some initial vertices in common, in this example, $u_0 = v_0$ but $u_1 \neq v_1$. At node Z, u_1 was first fixed and the branch of the search tree containing B_1 and $=B_1$ were searched. Then v_1 was fixed and the search descended to the branch containing node $=B_2$. The automorphism γ that was discovered maps each u_i to v_i. Therefore

once γ has been added to Aut(G), the program can ascend up the search tree to node Z, without searching any remaining portion of the current branch. There are several ways to find the node Z. The method used by *Groups & Graphs* works as follows. The path in the search tree from the top to the leaf node containing B is called the best path. The search node data structure contains a boolean value *OnBestPath*. This is initiallized to *false* each time *Stabilise* is entered. Whenever B is assigned, the program executes a loop which starts at *TopSearchNode* and sets the value of *OnBestPath* to *true* for every search node. When the algorithm later descends into other branches of the search tree, some of these will be changed to *false*. This is indicated in Fig. 8 by the letters t and f beside the nodes. The t's were assigned when node B_1 was visited. When an automorphism is discovered, a global flag *AutoFound* is set to *true*. So long as *AutoFound* remains *true* and *OnBestPath* is *false*, the program ascends the search tree. In the example, it will ascend to node Z before another branch is entered. The statement in the program that does this needs to be inserted after the recursive call to *Stabilise*.

```
if AutoFound then begin
    if not S^.OnBestPath then goto 1
    AutoFound := false
end
```

When *Stabilise* is descending the search tree the values *Degree*[v] are not re-initiallized after each refinement. This is because the degree is a cell-invariant of the partition. Before returning to the point where *Stabilise* was called from, it must re-initiallize the degrees, *only* for the vertices which were fixed by the refinement at that level. The reason for this is as follows. At a leaf node, every vertex currently in the partition became fixed. Before returning, the degrees of the vertices just fixed will all be reset to zero. When the calling program in turn is ready to return, it will reset to zero the degrees of the vertices which were fixed at that level, and so on up the tree. This selective re-initiallization saves an enormous amount of execution time.

It is also possible to input to the program known automorphisms of the graph before beginning the search. The group generated by the known automorphisms can prune the search tree significantly. Butler and Lam [8] have developed a very general algorithm that restricts the search to those portions of the search tree known to be equivalent under the action of a specified group.

Change of Basis.

There is a correspondence between the basis of the automorphism group, and the basis of the partitions created by refinement. Suppose that $u_0, u_1, \ldots, u_{i-1}$ have been fixed to create a partition Π_i. A vertex $u_i \in C_i$ is to be selected for stabilisation. We need to know the orbits of Γ_i on C_i. *Whenever a vertex u is to be selected for stabilisation, the basis of the automorphism group and the basis of the partition must be the same.* Consider again the search tree of Fig. 8. When the automorphism discovered at the node $=B_1$ is added to the group, the basis used for the group will be the current basis of the partition, corresponding to the leftmost path through the search tree. Later, when at node Y vertices are selected for stabilisation, the basis will have changed. This must be reflected in the data structure storing $\mathrm{Aut}(G)$. *The basis of the group is constantly changing.*

As presented in section 4, *AddGen* selects the point u to fix as any point moved by γ. This must be changed so that the basis of the group always agrees with that of the partition.

if $S^{\wedge}$.FixedPt $\neq 0$ then ItsGroup$^{\wedge}.u := S^{\wedge}$.FixedPt else select ItsGroup$^{\wedge}.u$ as any point moved by γ

Sometimes the basis of the group will be longer than the basis of the partition. In order to accomodate this possibility, the *FixedPt* of the search node will be set to zero to indicate an inactive search node. This occurs at the end of *Stabilise*.

Before selecting u for stabilisation, the program must check whether the basis of $\mathrm{Aut}(G)$ equals the current sequence of fixed points. The program executes very many base changes. In order to avoid comparing the sequences of stabilised points, we store some additional information. Let $U = u_0, u_1, \ldots$ denote the basis of $\mathrm{Aut}(G)$, and let $U' = u'_0, u'_1, \ldots$ denote the basis of the partition. The first time an automorphism is discovered, *AddGen* will construct $\Gamma_0, \Gamma_1, \ldots$ with the current basis U', so that the bases agree. When the program acsends the search tree up to a partition Π_i, the bases will still agree up to depth i. If the program now selects a new u_i to stabilise and descends to Π_{i+1} the bases will disagree at this point. The program will continue to descend the tree until a discrete partition is reached. When it then selects another point to stabilise, the bases will still be in agreement up to depth $i-1$. Thus we store a global variable *BasisOK* which contains the depth up to which the bases are in agreement, and a global variable

LastBaseChange which points to the searchnode whose depth equals *BasisOK*. So when the program is descending the search tree, the value of *BasisOK* doesn't change. When the program is ascending, its value can decrease. We add the following statements right after "*if AutoFound then* ...".

```
if S^.Depth > BasisOK then ChangeBase(S^.Depth)
BasisOK := S^.Depth
LastBaseChange := S
```

The base change algorithm used by *Groups & Graphs* is very straightforward. It simply deletes the generators and coset representatives from the groups at depth greater than *BasisOK* and rebuilds the tower of stabiliser subgroups from this depth. It also re-initiallizes the *CellOrbits*. An alternative way of changing the basis without deleting the existing information is to use the Butler-Sims base change algorithm [8]. It works by converting one basis into another through a sequence of transpositions of consecutive fixed points. My experience with graph isomorphism suggests that it is faster to simply delete the existing data structures and rebuild them. This may not be true for all implementations of the Butler-Sims algorithm.

```
ChangeBase(d: Integer)
{change the basis of the automorphism group, called from depth d}
var     Γ: GroupPtr
        S: SearchNodePtr
Begin
    S := LastBaseChange
    Γ := S^.ItsGroup
    if Γ = nil then return   {no group at this depth of recursion}
    if Γ^.Γ_u = nil then return   {Γ = identity group}
    {first reset the cell orbits from depth BasisOK}
    S := S^.NextSearchNode
    while S ≠ nil do begin
        {search nodes below depth d do not correspond to partitions/}
        if S^.Depth ≤ d then begin
            C := first cell of S^.Partition
            for i := C^.FirstPt to C^.LastPt do
                S^.CellOrbits^[V[i]] := -1
        end
```

```
            S := S^.NextSearchNode
        end
        save the generators from Γ
        delete coset representatives from Γ
        delete generators and coset representatives for all groups
            in Γ_u tower
        for each generator γ of Γ do AddGen(γ, LastBaseChange)
    End  {ChangeBase}
```

Marking the Cell Orbits.

When a point $u_i \in C_i$ has been stabilised in the search node containing Π_i, the orbit of Γ_i containing u_i must be marked to indicate that it is not necessary to stabilise any more points from that orbit. If this orbit later merges with other orbits when an automorphism is discovered, the resulting orbit must also be marked to indicate that a point from it has already been fixed. One way to do this is to utilise the *CellOrbits* array. Currently the value stored in *CellOrbits*$[v]$ has the following property.

$$\begin{aligned} &\mathit{CellOrbits}[v] > 0, \text{ if } v \text{ is not the representative of Orb}(v) \\ &\mathit{CellOrbits}[v] = -|\mathit{Orb}(v)|, \text{ if } v \text{ is the representative of Orb}(v) \end{aligned}$$

If v is an orbit representative, we can use the value stored in *CellOrbits*$[v]$ to mark an orbit. If n is the (global) number of points in the graph G, we can set

$$\begin{aligned} &\mathit{CellOrbits}[v] = -|\mathit{Orb}(v)|, \text{ if no point from Orb}(v) \text{ has been fixed} \\ &\mathit{CellOrbits}[v] = -|\mathit{Orb}(v)| - n, \text{ if a point from Orb}(v) \text{ has been fixed} \end{aligned}$$

The only effect this has is to require a slight modification to the procedure which merges orbits when an automorphism is added to the group. It also has the advantage of not requiring another array to mark the orbits, since another array would have to be re-initiallized every time the basis is changed.

6 AFTERWORD

Many modifications to these ideas and methods are possible. It is my experience that there are many people who are interested in writing isomorphism programs, but who are somewhat daunted by the difficulty and the lack of information on specific programming techniques. I have endeavored to provide the results of my experience programming graph isomorphism, in order to highlight the main ideas involved and to point out some of the difficulties, subtleties, and unexplored possibilities.

REFERENCES

[1] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison Wesley, Toronto, 1974.

[2] L. Babai, P. Erdös, S.M. Selkow, Random graph isomorphism, *SIAM J. of Computing* 9 (1980), 628–635.

[3] Laszlo Babai and Eugene Luks, Canonical labeling of graphs,*Proc. 15th Annual ACM Symp. on Theory of Computing*, SIGACT, 1983.

[4] J.M. Bennett and J.J. Edwards, A graph isomorphism algorithm using pseudoinverses, preprint, Computer Science Department, University of Sydney.

[5] N.L. Biggs and A.T. White, *Permutation Groups and Combinatorial Structures*, London Math. Soc. Lecture Notes #33, Cambridge Univ. Press, Cambridge, 1979.

[6] Bela Bollobas,*Graph Theory, an Introductory Course*, Springer-Verlag, New York, 1979.

[7] Gregory Butler and John Cannon, Computing in permutation and matrix groups I: normal closure, commutator subgroups, series, *Mathematics of Computation* 39 (1982), 663–670.

[8] G. Butler and C.W.H. Lam, A general backtrack algorithm for the isomorphism problem of combinatorial objects, *J. Symbolic Computation* 1 (1985), 363–381.

[9] Derek Corneil and Mark Goldberg, A non-factorial algorithm for canonical numbering of a graph, Technical Report #160/82, 1982, Department of Computer Science, University of Toronto.

[10] D.G. Corneil and C.C. Gotlieb, An efficient algorithm for graph isomorphism, *JACM* 17 (1970) 51–64.

[11] D.G. Corneil and D.G. Kirkpatrick, A theoretical analysis of various heuristics for the graph isomorphism problem, *SIAM J. Computing* 9 (1982), 281–297.

[12] M. Furst, J.E. Hopcroft, and E. Luks, A subexponential algorithm for trivalent graph isomorphism, 1980, Tech. Rept. 80–426, Dept. of Computer Science, Cornell University.

[13] R.M. Karp, Probabilisitic analyisis of a canonical numberiung algorithm for graphs, *Proc. Symp. Pure Math.* 34 (1979), 365–378.

[14] Zvi Galil, Christoff M. Hoffmann, Eugene M. Luks, Claus P. Schnorr, and Andreas Weber, An $O(n^3 \log n)$ deterministic and $O(n^3)$ probabilistic isomorphism test for trivalent graphs, *Proc. 23rd IEEE Symp. on the Foundations of Comp. Sci.*, N.Y., 1982, pp. 118–125.

[15] S.J. Gismondi and E.R. Swart, A polynomial-time procedure for resolving the graph isomorphism problem, Department of Computing and Information Science, UIniversity of Guelph, Ontario, preprint.

[16] Christoff Hoffmann, *Group Theoretic Algorithms and Graph Isomorphism*, Lecture Notes in Computer Science #136, Springer-Verlag, New York, 1982.

[17] Andrew Kirk, Efficiency considerations in the canonical labelling of graphs, Technical report TR-CS-85-05, Computer Science Department, Australian National University (1985).

[18] William Kocay, Groups & Graphs, a Macintosh application for graph theory, *J. Combinatorial Mathematics and Combinatorial Computing* 3 (1988), 195–206.

[19] J.S. Leon, An algorithm for computing the automorphism group of a Hadamard matrix, *J. Combinatorial Theory* 27A (1979) 289–306.

[20] Eugene M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, *Proc. 21st IEEE Symp. on the Foundations of Comp. Sci.*, Rochester, N.Y., 1980, 42–49.

[21] Rudi Mathon, Sample graphs for isomorphism testing, *Conressus Numerantium* 21 (1978) 499–517.

[22] Brendan McKay, *Topics in Computational Graph Theory*, Ph.D. Thesis, University of Melbourne, 1980.

[23] Brendan McKay, *Nauty User's Guide* (Version 1.5), Computer Science Department, Australian National University.

[24] Brendan McKay, Practical graph isomorphism, *Conressus Numerantium* 30 (1981), 45–87.

[25] Brendan McKay, *Backtrack Programming and the Graph Isomorphism Problem*, M.Sc. Thesis, University of Melbourne, 1976.

[26] R.C. Read and D.G. Corneil, The graph isomorphism disease, *J. Graph Theory* 1 (1977), 339–363.

[27] Mark Allen Weiss, *Data Structures and Algorithm Analysis*, Benjamin Cummings, Redwood City, California, 1992.

[28] Helmut Wielandt, *Finite Permutation Groups*, Academic Press, New York, 1964.

7

THE NONEXISTENCE OF 4–(12,6,6) DESIGNS

Brendan D. McKay
***Stanisław P. Radziszowski**

Department of Computer Science,
Australian National University,
Canberra, ACT 0200, Australia
bdm@cs.anu.edu.au
**Department of Computer Science*
Rochester Institute of Technology
Rochester, NY 14623, USA
spr@cs.rit.edu

ABSTRACT

With the help of computer algorithms we prove that there are no $4-(12,6,6)$ designs, thereby answering the last open existence question in design theory for at most 12 points. We also enumerate three families of related designs, namely the 10977 simple $3-(10,4,3)$ designs, the 67 simple $4-(11,5,3)$ designs, and the 23 simple $5-(12,6,3)$ designs. Finally, we complete the census of all possible partitions of 6-sets on 12 points into $5-(12,6,\lambda)$ designs and of 5-sets on 11 points into $4-(11,5,\lambda)$ designs.

1 INTRODUCTION

A $t-(v,k,\lambda)$ *design* is a pair $(X,\mathcal{D})$ where X is a v-element set of points and $\mathcal{D}$ is a multiset of k-element subsets of X, called *blocks*, such that for all $T \subseteq X$, $|T| = t$, $|\{K \in \mathcal{D} : T \subseteq K\}| = \lambda$. A design is called *simple* if it has no repeated blocks. For $v \leq 12$, $4-(12,6,6)$ was the only set of parameters for which design existence was unsettled before this work. The recent paper by Kreher, de Caen, Hobart, Kramer and Radziszowski [9] contains much information about the properties of this and related designs, in particular it notes that $4-(12,6,\lambda)$ designs exist for the arithmetically feasible adjacent λ's, namely 4 and 8, with an unknown case at $\lambda = 6$. Our result that no $4-(12,6,6)$ design exists is the first known example where such a hole of nonexistence appears as λ is varied

W. D. Wallis (ed.), Computational and Constructive Design Theory, 177–188.
© 1996 *Kluwer Academic Publishers.*

with the other parameters held fixed. The nonexistence of $4-(12,6,6)$ designs was firmly established using three independent methods, many important steps of which were performed in duplicate.

The family of simple $t-(t+7,t+1,\lambda)$ designs is perhaps one of the most investigated parameter situations in design theory [1], [2], [4], [5], [7], [9], [13], with particular attention given to the well known Witt designs with $\lambda = 1$. For $\lambda = 2$ all such simple designs were described in [13]. The census of this family for $\lambda = 3$ is completed here. The study of the latter is included in this work since all the simple $t-(t+7,t+1,3)$ designs for $t \geq 4$ are required by one of our proofs of the nonexistence of $4-(12,6,6)$ designs. Table 1 below summarizes the number of nonisomorphic simple $t-(t+7,t+1,\lambda)$ designs for all t and $\lambda \leq 3$ (for $\lambda \geq 4$ such designs are complements of those with $\lambda \leq 3$). This table completes Table I in [13], the new entries being those with $\lambda = 3$ and $3 \leq t \leq 5$.

t	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$
1	1	3	6
2	1	13	332
3	1	27	10977
4	1	1	67
5	1	1	23

Table 1. The number of nonisomorphic simple $t-(t+7,t+1,\lambda)$ designs

Having completed the enumeration of all simple $t-(t+7,t+1,\lambda)$ designs, we could not resist the temptation to answer the remaining open questions concerning their resolvabilities. This problem was studied originally by Kramer and Mesner in [7], who showed that there are at most two disjoint Steiner systems S(5,6,12). Other possible resolvabilities within this family were discussed in [13]. In Section 4 we report our computation completing the census of all possible resolvabilities for interesting t, namely all possible partitions of 6-sets on 12 points into $5-(12,6,\lambda)$ designs and of 5-sets on 11 points into $4-(11,5,\lambda)$ designs.

2 THREE TOWERS OF DESIGNS

For $8 \leq v \leq 12$, let D_v, A_v and B_v denote any p-$(v, s, 6)$, q-$(v, s, 3)$ and p-$(v, q, 4)$ design, respectively, where $p = v - 8$, $q = v - 7$ and $s = v - 6$. Table 2 gives the number of blocks in these designs in all cases. All such designs are known to exist [3] except D_{12}, i.e. 4-$(12, 6, 6)$. The counts of simple designs A_v appear in the last column of Table 1, and the main goal of this paper is to disprove the existence of D_{12}. Note that the point derived design from D_v is a design D_{v-1}, and similarly for A_v and B_v. In the sequel let $\overline{D}_{12}$ denote the blockwise complement of D_{12}. For all designs we assume a fixed set of points $X = \{0, 1, \ldots, 11\}$, or its subset of proper order, and also we will identify designs with their collections of blocks.

	$v = 8$	$v = 9$	$v = 10$	$v = 11$	$v = 12$
D_v	6	18	45	99	198
A_v	12	36	90	198	396
B_v	4	18	60	165	396

Table 2. The number of blocks in D_v, A_v and B_v designs

The following theorem contains a condensation of some results obtained in [9], which establish relationships between our three towers of designs.

Theorem 1 [9]
(a) *Any* 4-$(12, 6, 6)$ *design must be simple.*
(b) *For any* D_{12}, $\overline{D}_{12}$ *is also a* 4-$(12, 6, 6)$ *design.*
(c) $D_{12} \cup \overline{D}_{12}$ *is a simple* A_{12} *design.*
(d) $D_{12} \cap \overline{D}_{12} = \emptyset$; *equivalently,* D_{12} *has no disjoint blocks.*
(e) *Each 5-set is covered by one or two blocks of* D_{12}.
(f) *In* A_{12}, *if* S *is a block then* $\overline{S}$ *is also a block.*
(g) *Each 7-set contains exactly* 3 *blocks of* A_{12}.
(h) *The 5-sets covered by one block of* D_{12} *form a* B_{12}.
(i) *The 5-sets covered by two blocks of* D_{12} *form a* B_{12}.
(j) *Each block of* D_{12} *intersects* $0, 8, 50, 80, 55, 4$ *other blocks in* $0, 1, \ldots, 5$ *points, respectively.*

Some additional properties of A_v designs, formulated in Theorem 2 below, were used in the computations.

Theorem 2 *Let A_{12} be any simple $5-(12,6,3)$ design. Then*
(a) *No k-ly derived design of A_{12} ($k = 0,1,2,3$) can have a $(7-k)$-set containing more than 3 blocks.*
(b) *If A_{10} is a doubly derived design from A_{12}, then any 5-set of A_{10} contains 1, 2, or 3 blocks.*

Proof. Since we consider only simple designs, properties (a) and (b) follow immediately from Theorem 1(g). ∎

3 ALGORITHMS

The success of our study was considerably facilitated by the use of efficient programs for the canonical labelling of designs. The program **nauty** written by the first author [11] can perform rapid canonical labelling of arbitrary graphs, and so can be applied to the bipartite point-block incidence graphs of designs. However, this approach is not very efficient, as the graphs are quite large ($v+b$ vertices) and labellings are treated as permutations of the points and blocks together. In fact, it suffices to treat labellings of the points only, as an ordering of the points implies an ordering of the blocks apart from the interchange of equal blocks.

Accordingly, a version of **nauty** was prepared that treats designs $(X, \mathcal{D})$ as hypergraphs with vertex set X and edge set $\mathcal{D}$. Both vertices and edges can be coloured, the latter to enable direct processing of the partitionings we will describe in Section 4. Automorphisms and labellings appear as permutations of X only. Only about 5% of the code for **nauty** needed to be replaced, but the result was an improvement of about one order of magnitude in processing efficiency.

Typical execution times for computing automorphism groups and canonical labellings are 0.5 seconds for $3-(10,4,2)$ designs, 1.6 seconds for $3-(10,4,3)$ designs, 20 seconds for $4-(11,5,3)$ designs and 520 seconds for $5-(12,6,3)$ designs. These times are in fact unnecessarily long, as the program is tuned not for completed designs but for partial designs induced by some subset of the points. In this case the program is usually very much faster, for example 0.011 seconds for six points of a $3-(10,4,3)$ design. All the computer times given in this paper refer to a Sun Microsystems Sparcstation One computer (approximately 12 mips).

There are a very large number of designs D_{10}. This lead us to avoid the most natural approach of enumerating D_{12}'s by straightforward extensions $D_i \rightarrow D_{i+1}$, for $i = 8, \ldots, 11$. Instead we took two alternative approaches, one similar to the algorithms described by Ivanov [6], and the other one using design extensions $A_i \rightarrow A_{i+1}$, for $i = 8, \ldots, 11$ and the relationship between the A_v-tower and D_v-tower of designs.

Let S be a t-(v, k, λ) design with b blocks. If one sees S as a $v \times b$ (0,1)-matrix M with $M[i, j] = 1$ if and only if block j contains point i, then the core of our two approaches lies in finding a labelling in which row-by-row or column-by-column, respectively, construction of M is efficient, and limits the combinatorial explosion in the number of intermediate configurations. In our experience both methods may work well after a proper combinatorial study of the design S is performed. In the case of D_v and A_v designs such a detailed study has been done in [9]. The bulk of the current work consisted of the choice of properties to be implemented, the choice of appropriate methods for the different computational paths, and finally implementation, tuning of programs and verification of results.

ALG1. Row-by-row construction.
Let $S[n]$ stand for the configuration of partial blocks of S corresponding to the first n rows of matrix M. The set of feasible $(n+1)$-th rows are the 0-1 solutions of a set of linear equations that express the basic design properties. In addition, there may be further equations or inequalities derived from theory such as that in Theorem 1. If $S[n]$ has many equal columns, there can be many solutions which only differ by the interchange of those columns. Such equivalent solutions can be avoided by replacing the 0-1 variables for each set of equal columns by an integer variable equal to their sum.

Overall isomorph rejection can be handled by several different methods. One is to simply generate all feasible extensions and reject isomorphs using **nauty**. The other employs **nauty** in a quite different way. Suppose $S[n+1]$ is constructed by adjoining row $n+1$ to $S[n]$. Application of **nauty** provides the orbits of the automorphism group of $S[n+1]$ and also a canonical labelling of the rows. $S[n+1]$ is then rejected if row $n+1$ is not in the same orbit as the row whose canonical label is "1". Isomorph rejection is then carried out only within the matrices derived from each particular $S[n]$. It is not hard to show [12] that each isomorphism class is represented exactly once in the output.

ALG2. Column-by-column construction.
By using a suitable labelling of the points, this approach is reduced to con-

secutive extension of $(s-1)$-designs S_{s-1} to s-designs S_s, for $s = 1, \ldots, t$. All extensions of a single $(s-1)$-design can be obtained by solving an appropriate system of integer equations (possibly also inequalities, if known). Each design extension adds $|S_s| - |S_{s-1}|$ columns to the matrix M.

4 CONSTRUCTIVE RESULTS

Using both methods (ALG1) and (ALG2) from Section 3, we have obtained the following enumeration results. The computations will be described in more detail in Section 5.

Theorem 3

(a) *There are exactly 10977 nonisomorphic simple* $3-(10,4,3)$ *designs. Of these, 1685 satisfy condition (b) of Theorem 2, and 248 of the latter extend to a simple* $4-(11,5,3)$ *design.*

(b) *There are exactly 67 nonisomorphic simple* $4-(11,5,3)$ *designs. Of these, 28 are rigid, 6 have groups of order 2, 14 of order 3, 6 of order 4, 11 of order 8, and one each of orders 12 and 24. None of these designs is point transitive.*

(c) *There are exactly 23 nonisomorphic simple* $5-(12,6,3)$ *designs. Of these, none is rigid, 3 have groups of order 2, 2 of order 3, 2 of order 9, 4 of order 12, 2 of order 16, 3 of order 36, 2 of order 48, and one each of orders 6, 24, 32, 144 and 288. Exactly seven of these designs are point transitive.*

Observe that by Alltop's extension theorem every A_{11} extends uniquely to an A_{12}, hence given all A_{12}'s one can easily recover all A_{11}'s. Several A_{12} designs have been described in [9], hence here we only note that we found 10 more, including one with the largest automorphism group of order 288. The latter design is described in full in the sequel as a part of a partition of all 6-sets on 12 points into 5-designs. The interested reader is encouraged to request electronic copies of some or all designs from Theorem 3 from the authors.

Some particularly beautiful configurations of sets can be obtained by partitioning all k-subsets of a v-set into designs. Kramer and Mesner [7] proved that there are no three mutually disjoint Steiner systems $S(5,6,12)$. The computations in [13] showed that there is a unique simple $5-(12,6,2)$ design, which is formed by two disjoint copies of $S(5,6,12)$. As observed in [13], this implies that the only possible partitions of all 6-sets on 12 points into nonresolvable

$5-(12,6,\lambda)$ designs are of type (1+1+5) or (1+3+3), where the type lists the λ's of the participating resolvent designs. In particular, the latter implies that there is no resolvable simple $5-(12,6,3)$ design. A very detailed study of type (1+1+5) partition can be found in [7]. Before the current work no partition of type (1+3+3) was known.

Please note that all the remarks of the preceding paragraph are equally valid for partitioning the 5-sets on 11 points into $4-(11,5,\lambda)$ designs.

By analyzing the designs from Theorem 3(b) and 3(c) we have been able to construct all such nonisomorphic partitions on 11 and 12 points, and we summarize them in the next theorem. By an automorphism of a type (1+3+3) partition into designs we understand a permutation of points which is an automorphism of the Steiner resolvent, and is an automorphism of two other resolvents or is an isomorphism between them. The resolvability claims below are consequences of Theorem 3, the computations, and the remarks above.

Theorem 4

(a) *There are exactly 7 nonisomorphic partitions of all 5-sets on 11 points into* $4-(11,5,\lambda)$ *designs of type* $(1+3+3)$. *Of these, two are rigid, and the other 5 have 3 automorphisms. In no case is there an automorphism exchanging two resolvents. Nine nonisomorphic* $4-(11,5,3)$ *designs are resolvents of such partitions; equivalently, 58 of the 67 simple* $4-(11,5,4)$ *designs are not resolvable.*

(b) *There are exactly 5 nonisomorphic partitions of all 6-sets on 12 points into* $5-(12,6,\lambda)$ *designs of type* (1+3+3). *Of these, two have 9 automorphisms, and the other 3 have 36 automorphisms and are point transitive. In no case is there an automorphism exchanging two resolvents. Seven nonisomorphic* $5-(12,6,3)$ *designs are resolvents of such partitions; equivalently, 16 of the 23 simple* $5-(12,6,4)$ *designs are not resolvable.*

Similarly to Theorem 3, every partition from (a) extends uniquely to a partition from (b). Also, given all partitions from (b), one can easily recover all partitions from (a).

Now we will give a full description of a partition of type $(1+3+3)$. We will call the resolvents R_1, R_2 and R_3, respectively.

Define four permutations of $\{0,1,\ldots,11\}$:

$$g_1 = (1\ 3\ 5)(2\ 11\ 10)(6\ 8\ 9)$$

$$
\begin{aligned}
g_2 &= (0\ 2\ 7\ 1\ 4\ 8)(3\ 10\ 9\ 5\ 11\ 6) \\
g_3 &= (4\ 5\ 11\ 8)(6\ 10\ 7\ 9) \\
g_4 &= (3\ 5)(4\ 10)(7\ 9)
\end{aligned}
$$

Resolvent R_1 is a $5-(12,6,1)$ design. Its automorphism group is $G_1 = \langle g_1, g_2, g_3 \rangle$, which is a 5-transitive representation of the Matthieu group M_{12} (order 95040). The action on the blocks is transitive. Resolvent R_2 is a $5-(12,6,3)$ design. Its automorphism group is $G_2 = \langle g_1, g_2, g_4 \rangle$, which acts transitively and has order 288 (the largest for any $5-(12,6,3)$ design). There are seven orbits of blocks. Resolvent R_3 is a $5-(12,6,3)$ design. Its automorphism group is $G_3 = \langle g_1, g_2 \rangle$, which acts transitively and has order 36. There are eleven orbits of blocks.

In Table 3, we give the lengths and representative members of the actions of group G_i on the blocks of R_i, for $i = 1, 2, 3$. Since $G_3 = G_1 \cap G_2$ and the three resolvents are nonisomorphic, the full automorphism group of the partition is G_3.

5 COMPUTATIONS

The nonexistence of D_{12} was established by three independent computations, and all the design constructions have been obtained at least twice each, with different implementations.

The following were the major computations for this work: sequences (C1) and (C2) used the point extension algorithm (ALG1), sequence (C3) used the standard design extension algorithm (ALG2), and (C4) used both.

(C1) $D_{12}[6] \rightarrow D_{12}[7] \rightarrow D_{12}[8] \rightarrow D_{12}[9] \rightarrow D_{12}[10]$. Properties (a), (b), (c), (e), (j) of Theorem 1 were enforced for D_{12}, and property (e) for $\overline{D}_{12}$. The unique $D_{12}[6]$ containing a full block was used as a starting point. In addition the required number of partial blocks of each size were encoded. It is worth observing that some of these properties are logically redundant, but experiments showed that their explicit inclusion dramatically decreased the execution time. The numbers of partial designs obtained at each level were 1, 13, 28794, 14084, and 3, respectively. None of the three designs $D_{12}[10]$ extended to 11 points.

Resolvent	*Orbit length*	*Representative*
R_1	132	$\{5,6,7,8,9,10\}$
R_2	12	$\{3,4,5,6,8,10\}$
	12	$\{3,4,5,7,9,10\}$
	18	$\{3,5,6,7,8,9\}$
	48	$\{3,5,7,8,10,11\}$
	144	$\{4,5,7,8,10,11\}$
	144	$\{5,7,8,9,10,11\}$
	18	$\{6,7,8,9,10,11\}$
R_3	36	$\{3,4,5,7,8,10\}$
	36	$\{3,4,5,8,9,10\}$
	36	$\{3,5,7,8,9,10\}$
	36	$\{3,5,8,9,10,11\}$
	36	$\{4,5,6,9,10,11\}$
	36	$\{4,5,7,8,9,11\}$
	36	$\{4,5,7,9,10,11\}$
	36	$\{4,7,8,9,10,11\}$
	36	$\{5,6,7,8,9,11\}$
	36	$\{5,6,7,8,10,11\}$
	36	$\{5,6,7,9,10,11\}$

Table 3. A partition of all 6-sets on 12 points into three $5-(12,6,\lambda)$ designs

(C2) $A_{12}[6] \to A_{12}[7] \to \cdots \to A_{12}[12]$. Properties (f) and (g) of Theorem 1, and the pattern of blocks sizes within each 7-set were enforced. The starting points were the two possible configurations of six points, one covering a block, the other not. The numbers of partial designs obtained at each level were 2, 1, 6, 232, 2424, 67 and 23, respectively. The value 2424 is reduced to 1685 if property (b) of Theorem 2 is enforced as well. It was checked that the generated A_{12}'s do not split into $D_{12} \cup \overline{D}_{12}$.

(C3) $A_8 \to A_9 \to A_{10}^* \to A_{11} \to A_{12}$. The designs A_9 agreed with the results in [5]. Only special A_{10}'s were produced, namely those satisfying Theorem 2(b). The number of designs obtained at each level was 6, 332, 1685, 67 and 23, respectively. Crosschecks with intermediate results of sequences (C1) and (C2) were done. It was checked twice that D_{12} cannot be obtained: firstly by splitting special A_{10}'s into candidates for partial systems $D_{12}[10] \cup \overline{D}_{12}[10]$ and showing that none can be completed, and secondly by verifying that A_{12}'s do not split into $D_{12} \cup \overline{D}_{12}$.

(C4) $A_9 \rightarrow A_{10}$. Dramatic improvement of the programs after (C3) had been finished enabled a full census of the 10977 simple $3-(10,4,3)$ designs A_{10} to be completed in about 100 hours by method (ALG2). In slightly more time, it was repeated by method (ALG1) with identical results.

Finally, the special designs A_{10}^* needed by (C3) were extracted from the 10977 and checked against those found before.

In addition, several small testing programs were implemented to crosscheck other intermediate results of different sequences. These programs were based on obvious relationships between designs, and the properties listed in Theorems 1 and 2. For example, 248 out of the 1685 special A_{10}'s produced in sequence (C3) extended to 67 A_{11}'s. After rejecting isomorphs amongst the 11×67 point derived designs, we recovered the same set of 248 A_{10}'s. Note that each of the sequences (C1), (C2) and (C3) shows the nonexistence of D_{12} designs. Thus we have:

Theorem 5 $4-(12,6,6)$ *designs do not exist.*

The partitions of Theorem 4 were obtained as follows. First we produced all simple $4-(11,5,4)$ designs by taking the complements of A_{11}'s. Then using an equation solver we found all possible embeddings of the Steiner system $S(4,5,11)$ as a subdesign of each $4-(11,5,4)$. Each such embedding gave an instance of a type (1+3+3) partition. The partitions of Theorem 4(b) were obtained twice: the same way as those of Theorem 4(a) and independently by applying Alltop's extension theorem to those from Theorem 4(a). As usual, all these computations were performed independently by each of two authors, using distinct programs. The computer time needed to obtain all partitions was negligible.

The total computer time used for all computations was just about 1000 cpu hours, including all repetitions and verification. A single disproof of the existence along path (C3) could be obtained in about 50 cpu hours and along path (C2) in about twice that long. Our computations were completed quickly by using a network of about 120 Sun computers simultaneously.

6 THE NEXT CHALLENGE

To the best of our knowledge, as reported in [3], the only remaining open existence question in design theory for $v \leq 13$ is that for $4-(13,6,6)$ designs (286 blocks). For other arithmetically feasible λ's with the same parameters such designs exist; namely, there are known constructions of simple $4-(13,6,12)$ [8] and simple $4-(13,6,18)$ designs. The latter are actually the simple $5-(13,6,4)$ designs found in [10].

The only design on 14 points whose existence is in question is a $5-(14,7,6)$ design [3]. Note that if a $5-(14,7,6)$ design exists, its derived design is a $4-(13,6,6)$ design, mentioned above. Hence any construction of a simple $5-(14,7,6)$ design or proof of the nonexistence of $4-(13,6,6)$ designs would answer all remaining existence questions for $v \leq 14$.

Observe that, using the notation from Section 2, the known $5-(13,6,4)$ and $6-(14,7,4)$ designs [10] can be seen to be B_{13} and B_{14}, respectively. We feel that one can expect a relationship between the unknown tower of designs $t-(t+9,t+2,6)$ and $t-(t+8,t+1,4)$ $(= B_{t+8})$, to be similar to the relationship between the D_v and A_v towers studied here.

REFERENCES

[1] T. Beth, D. Jungnickel and H. Lenz, *Design Theory*, Cambridge University Press (1986).

[2] A. E. Brouwer, The t-designs with $v < 18$, *Stichting Math. Centrum* ZN 76/77, Amsterdam, August 1977.

[3] Y. M. Chee, C. J. Colbourn and D. L. Kreher, Simple t-Designs with $v \leq 30$, *Ars Combinatoria* 29 (1990), 193–261.

[4] P. B. Gibbons, *Computing Techniques for the Construction and Analysis of Block Designs* (Ph.D. thesis), Department of Computer Science, University of Toronto (1976).

[5] J. J. Harms, C. J. Colbourn and A. V. Ivanov, A Census of (9,3,3) Block Designs without Repeated Blocks, *Congressus Num.*57 (1987), 147–170.

[6] A. V. Ivanov, Constructive Enumeration of Incidence Systems, *Ann. Discrete Math.* 26 (1985), 227–246.

[7] E. S. Kramer and D. M. Mesner, Intersections Among Steiner Systems, *J. Combinatorial Theory* 16A (1974), 273–285.

[8] E. S. Kramer and D. M. Mesner, t-Designs on Hypergraphs, *Discrete Math* 15 (1976), 263–296.

[9] D. L. Kreher, D. de Caen, S. A. Hobart, E. S. Kramer and S. P. Radziszowski, The Parameters $4-(12,6,6)$ and Related t-Designs, *Austral. J. Combin.* 7 (1993), 3–20.

[10] D. L. Kreher and S. P. Radziszowski, The Existence of Simple $6-(14,7,4)$ Designs, *J. Combinatorial Theory* 43A (1986), 237–243.

[11] B. D. McKay, nauty Users' Guide (Version 1.5), *Technical Report* TR-CS-90-02, Computer Science Department, Australian National University, 1990.

[12] B. D. McKay, Isomorph-free exhaustive generation, to appear.

[13] S. P. Radziszowski, Enumeration of All Simple $t-(t+7,t+1,2)$ Designs, *J. Combin. Math. Combin. Comput.* 12 (1992), 175–178.

8

BIBDS WITH $k = 6$ AND $\lambda = 1$

W. H. Mills

Institute for Defense Analyses,
Princeton, New Jersey

ABSTRACT

Balanced incomplete block designs, or BIBDs, have been studied for many years. In a famous 115 page paper Haim Hanani completely settled the existence question for BIBDs with block size less than 6. In the same paper he settled this question for block size 6 and $\lambda > 1$. However when the block size is 6 and $\lambda = 1$ the whole question becomes much more difficult. In the present paper we discuss the work that has been done on this. There are, at present, 55 values of v for which the existence of such a design is in doubt. We will show how the remaining values of v are handled.

1 INTRODUCTION

Let v, k, and λ be positive integers. A balanced incomplete block design, or BIBD, consists of a v-element set S and a collection $B_1, B_2, \ldots, B_b$ of k-element subsets called blocks, such that every pair of distinct elements of S appears in exactly λ blocks. We call such a design a (v, k, λ)–BIBD. For what values of the parameters does such a BIBD exist? This is one of the central questions of combinatorial design theory.

H. Hanani [12] has found necessary and sufficient conditions for the existence of a BIBD when $k \leq 5$, as well as when $k = 6$, $\lambda > 1$. This leaves the case $k = 6$, $\lambda = 1$.

W. D. Wallis (ed.), Computational and Constructive Design Theory, 189–226.
© 1996 *Kluwer Academic Publishers.*

Simple counting arguments tell us that each element of S appears in exactly r blocks where

$$r = \frac{(v-1)\lambda}{k-1},$$

and that the total number b of blocks is exactly

$$b = \frac{v(v-1)\lambda}{k(k-1)}.$$

Thus we have the divisibility conditions $(k-1)|(v-1)\lambda$ and $k|v(v-1)\lambda$.

For our case $k = 6$, $\lambda = 1$ these conditions are equivalent to $v \equiv 1$ or $6 \pmod{15}$.

There are a few additional negative results. Some of them come from Fisher's inequality which states that $b \geq v$ if $v > k$. (See Hall [6].) Thus there are no BIBDs with $\lambda = 1$ and $k < v < k^2 - k + 1$. In particular for $k = 6$, $\lambda = 1$ the cases $v = 16$ and 21 are impossible.

A $(k^2, k, 1)$–BIBD is a finite affine or Euclidean plane of order k. It is well known (and easy to show) that such a plane can be extended to a finite projective plane of order k. By the Bruck-Ryser theorem [7] if such a projective plane exists with $k \equiv 1$ or $2 \pmod 4$, then k is the sum of two squares. (See Hall [6].) In particular a (36,6,1)–BIBD cannot exist.

We will say that a positive integer v is admissible if $v \equiv 1$ or $6 \pmod{15}$ and $v \neq 16$, 21, or 36.

R. M. Wilson [31] has shown that for fixed k and λ a (v, k, λ)–BIBD exists for all sufficiently large v that satisfy the divisibility conditions. However his methods are not entirely constructive and apply only to extremely large values of v.

In his large 1975 paper Hanani [6] stated that "at the present state of knowledge" it was almost hopeless to find reasonable necessary and sufficient conditions for the existence of BIBDs with $k = 6$ and $\lambda = 1$. However there have been many developments in the last 20 years and now it is possible to state the following result (See [2]).

Theorem 1. *A $(v, 6, 1)$–BIBD exists for all admissible v with 55 possible exceptions. These 55 possible exceptions are listed in Table 1.*

Table 1
The 55 values of v for which
the existence of a $(v, 6, 1)$ BIBD is in doubt.

46	51	61	81	141	166	196	201	226	231	256
261	276	286	291	316	321	346	351	376	406	411
436	441	466	471	496	501	526	561	591	616	646
651	676	706	741	766	771	796	801	916	946	1096
1221	1246	1251	1396	1456	1486	1521	1611	1671	1851	2031

In this paper we will discuss the known constructions for $k = 6$ and $\lambda = 1$, and we will outline the proof of Theorem 1. We must consider the admissible values of v that are not listed in Table 1.

We note first that for $v = 1$, and for $v = 6$, $(v, 6, 1)$–BIBDs exist and are trivial. In the first case we have no blocks and in the second one block. We have already observed that these BIBDs do not exist for $v = 16$ and $v = 21$ by Fisher's inequality. The (31,6,1)–BIBD is a finite projective plane of order 5, so of course it exists, and it is denoted by PG(2,5). The (36,6,1)–BIBD does not exist by the Bruck-Ryser Theorem as noted above.

The next three admissible values of v are listed in Table 1, namely $v = 46$, $v = 51$, and $v = 61$. The first two of these are famous unsolved problems, and it begins to look as though they may not exist. At any rate a great deal of effort has gone into looking for them without success.

2 AUTOMORPHISM GROUPS AND BASE BLOCKS

Next we come to a number of computer constructed designs. In each case we give a permutation group A on our basic set S, and a set of base blocks. This set of base blocks has two properties: (i) the set of distinct images of these blocks under the permutation group is the required BIBD, and (ii) no one of these base blocks is the image of another one under a permutation in our group A. Thus A becomes an automorphism group of our BIBD. These designs are in scattered places in the literature. We gather them all together here.

We start with a detailed discussion of the case $v = 171$, which will illustrate some of the ideas used in the computer searches. Here $b = 969 = 57 \cdot 17$. We must first choose a putative automorphism group A. If this group is too small, then there are too many cases to be examined. If it is too large, then no design has that automorphism group. It is natural to try a transitive permutation group of order 171. Then if p is a fixed element of S, we can identify a permutation α of the group with the element $\alpha(p)$, so that we can think of the set S as the set of elements of our group. It can be shown that if this group is abelian, then each base block generates exactly 171 blocks. (See [21].) Since b is not a multiple of 171 we must use a nonabelian group. We choose the one with no elements of order 9 because it is slightly easier to work with. We now take our basic set S to consist of the elements of the group generated by an element β of order 19, and two elements α and γ of order 3, such that γ commutes with α and β and $\alpha^{-1}\beta\alpha = \beta^7$. We note that 7 is a cube root of unity modulo 19.

Thus an arbitrary element of S has the form $\alpha^x\beta^y\gamma^z$ and we will write it as (x, y, z) for convenience. For any fixed element ξ of S we let L_ξ denote the left multiplication by ξ. Our automorphism group consists of all the mappings L_ξ.

The condition that each pair of elements occurs in a block is equivalent to the condition that each nonidentity element of S occurs as the quotient $\mu^{-1}\nu$ of two elements μ, ν in one of the base blocks. Then if we have the correct number of blocks each pair must occur exactly once.

Each of our base blocks generates 57 or 171 blocks. Let n_1 be the number of base blocks that generate 171 blocks, and let n_3 be the number of base blocks that generate 57 blocks. We must have $171n_1 + 57n_3 = 969$ or $3n_1 + n_3 = 17$. In order to keep the number of base blocks to a minimum we try $n_1 = 5$ and $n_3 = 2$. We will arrange the base blocks so that the first five generate 171 blocks each, and the last two generate 57 blocks each.

It is easy to see that if a base blocks generates exactly 57 blocks it must be of the form

$$\mu \quad \omega\mu \quad \omega^2\mu \quad \nu \quad \omega\nu \quad \omega^2\nu,$$

where ω is an element of order 3 that is not in the center of our group.

Now let the elements of the j-th base block be (x_{ij}, y_{ij}, z_{ij}), $1 \leq i \leq 6$. We begin with the x_{ij}. The last two base blocks must be of the form

$$(0,*,*) \quad (1,*,*) \quad (2,*,*) \quad (0,*,*) \quad (1,*,*) \quad (2,*,*),$$

where the asterisks denote numbers to be determined later. By a counting argument we see that the first five base blocks contain exactly 26 pairs of the form $\{(d, e, f), (d, g, h)\}$ with the ordered pair (e, f) different from the ordered pair (g, h). Using this fact and the possible symmetries it can be shown that there are 64 possible choices, up to symmetries, for the first coordinates of the first five base blocks. We choose one arbitrarily:

$$\begin{array}{cccccc}
(0,*,*) & (0,*,*) & (0,*,*) & (0,*,*) & (1,*,*) & (2,*,*) \\
(0,*,*) & (0,*,*) & (0,*,*) & (0,*,*) & (1,*,*) & (2,*,*) \\
(0,*,*) & (0,*,*) & (0,*,*) & (1,*,*) & (1,*,*) & (1,*,*) \\
(0,*,*) & (0,*,*) & (0,*,*) & (1,*,*) & (1,*,*) & (2,*,*) \\
(0,*,*) & (0,*,*) & (0,*,*) & (1,*,*) & (2,*,*) & (2,*,*).
\end{array}$$

So far we have made a number of arbitrary choices. If this particular set of choices does not work we go back and try another one. Fortunately the choices we made work.

Now we go and tackle the second coordinates. Here we use a simple backtrack algorithm, making use of any symmetries we can find to reduce the computer time needed. It was necessary to try many possible choices for the second coordinates before we found one that could be extended to a complete BIBD, namely

$$\begin{array}{cccccc}
(0,0,*) & (0,0,*) & (0,1,*) & (0,3,*) & (1,0,*) & (2,2,*) \\
(0,0,*) & (0,2,*) & (0,6,*) & (0,10,*) & (1,1,*) & (2,1,*) \\
(0,0,*) & (0,1,*) & (0,5,*) & (1,4,*) & (1,10,*) & (1,18,*) \\
(0,0,*) & (0,2,*) & (0,12,*) & (1,4,*) & (1,10,*) & (2,7,*) \\
(0,0,*) & (0,3,*) & (0,12,*) & (1,5,*) & (2,6,*) & (2,17,*) \\
(0,0,*) & (1,0,*) & (2,0,*) & (0,12,*) & (1,12,*) & (2,12,*) \\
(0,4,*) & (1,4,*) & (2,4,*) & (0,18,*) & (1,18,*) & (2,18,*).
\end{array}$$

Once we have got this far another backtrack algorithm gives us the third coordinates and we obtain our set of seven base blocks:

$$\begin{array}{cccccc}
(0,0,0) & (0,0,1) & (0,1,0) & (0,3,2) & (1,0,0) & (2,2,1) \\
(0,0,0) & (0,2,0) & (0,6,2) & (0,10,0) & (1,1,0) & (2,1,1) \\
(0,0,0) & (0,1,1) & (0,5,1) & (1,4,0) & (1,10,1) & (1,18,0) \\
(0,0,0) & (0,2,1) & (0,12,2) & (1,4,2) & (1,10,2) & (2,7,1) \\
(0,0,0) & (0,3,0) & (0,12,1) & (1,5,1) & (2,6,1) & (2,17,0) \\
(0,0,0) & (1,0,1) & (2,0,2) & (0,12,0) & (1,12,1) & (2,12,2) \\
(0,4,0) & (1,4,1) & (2,4,2) & (0,18,1) & (1,18,2) & (2,18,0).
\end{array}$$

Having found our design we write it in the more compact form given in [2]. We now take our set S to be $\mathrm{Z}(3)\times\mathrm{Z}(57)$, the direct product of the cyclic groups

of order 3 and 57. We take our automorphism group A to be the 171 element group generated by the permutations σ and τ where $\sigma(g,h) = (g, h+1)$ and $\tau(g,h) = (g+1, 7h)$. The seven base blocks are

$$\begin{array}{cccccc}
(0,0) & (0,19) & (0,39) & (0,41) & (1,14) & (2,38) \\
(0,0) & (0,21) & (0,44) & (0,48) & (1,26) & (2,11) \\
(0,0) & (0,1) & (0,43) & (2,8) & (2,15) & (2,44) \\
(0,0) & (0,40) & (0,50) & (1,11) & (2,25) & (2,34) \\
(0,0) & (0,3) & (0,31) & (1,23) & (1,43) & (2,36) \\
(0,0) & (0,12) & (1,0) & (1,27) & (2,0) & (2,18) \\
(0,37) & (0,42) & (1,9) & (1,31) & (2,6) & (2,46).
\end{array}$$

The actual mapping used to obtain this form is

$$(x,y,z) \to (2x, 19(2x+z) + 39 \cdot 11^x y).$$

The last two of these base blocks are fixed by τ. It is a straightforward matter to check that the 969 blocks obtained from these seven base blocks form a (171,6,1)–BIBD.

We now proceed to list the remaining designs of this type. Except for the designs of Denniston and Hanani, these are all computer generated designs. We omit detailed discussions of these designs and content ourselves with giving the base blocks and the corresponding automorphism group. It is a routine calculation to check that the distinct images of the given base blocks form the required BIBD. As before b will denote the total number of blocks in our BIBD.

$v = 66$ (Denniston [9]). Here $b = 143$. We take our set S to consist of a point at infinity X and the 65 ordered pairs (g,h) where g is an element of GF(13) and h is either an element of GF(3) or one of two additional infinite elements Y, Z. Our group A is the 39 element nonabelian group generated by σ and τ where $\sigma X = \tau X = X$, $\sigma(g,h) = (g+1, h)$, $\tau(g,h) = (3g, h+1)$, where $Y+1$ and $Z+1$ are defined to be Y and Z respectively. Our base blocks are

$$\begin{array}{ccccccc}
X & (0,0) & (0,1) & (0,2) & (0,Y) & (0,Z) \\
(3,0) & (10,0) & (5,2) & (6,2) & (0,Y) & (2,Y) \\
(4,0) & (6,0) & (7,1) & (11,2) & (0,Y) & (1,Z) \\
(11,1) & (3,2) & (9,2) & (12,2) & (0,Z) & (1,Z) \\
(2,Y) & (3\cdot 2, Y) & (3^2 \cdot 2, Y) & (4,Z) & (3\cdot 4, Z) & (3^2 \cdot 4, Z).
\end{array}$$

The first and last of these base blocks are fixed by τ so they generate 13 blocks each. The other 3 base blocks generate 39 blocks each. We notice that if S' denotes the 40 element subset consisting of X and the 39 ordered pairs (g,h) with h in GF(3), then every block of our design meets S' in four points or not

at all. This set of 40 points is a "maximal arc". If we delete the elements not in S' from our blocks and delete the blocks that do not intersect S', we obtain a (40,4,1)–BIBD. Indeed Denniston obtained this (66,6,1)–BIBD by starting with a known (40,4,1)–BIBD and extending it.

$v = 76$ [22]. Here $b = 190$. We take our set S to consist of the 76 ordered pairs (g, h) where g is an element of GF(19) and h is either an element of GF(3) or an infinite element Y. The group A is the 57 element nonabelian group generated by σ and τ where $\sigma(g, h) = (g + 1, h)$, $\tau(g, h) = (7g, h + 1)$, where $Y + 1$ is defined to be Y as before. Our base blocks are

$$\begin{array}{cccccc}
(0,0) & (1,0) & (3,0) & (14,0) & (10,2) & (0,Y) \\
(0,0) & (7,0) & (0,1) & (2,2) & (1,Y) & (3,Y) \\
(3,0) & (7\cdot 3,1) & (7^2\cdot 3,2) & (1,Y) & (7,Y) & (7^2,Y) \\
(12,0) & (7\cdot 12,1) & (7^2\cdot 12,2) & (3,Y) & (7\cdot 3,Y) & (7^2\cdot 3,Y) \\
(7,0) & (7\cdot 7,1) & (7^2\cdot 7,2) & (16,0) & (7\cdot 16,1) & (7^2\cdot 16,2) \\
(13,0) & (7\cdot 13,1) & (7^2\cdot 13,2) & (17,0) & (7\cdot 17,1) & (7^2\cdot 17,2).
\end{array}$$

The last four of these base blocks are fixed by τ so they generate 19 blocks each. The first 2 base blocks generate 57 blocks each.

The next case is $v = 81$ which is open and so listed in Table 1.

In the designs given in the rest of this section our basic set S will be either an abelian group or an abelian group with a single infinite element X adjoined. If h is a fixed element of this abelian group, we let σ_h denote the permutation given by $\sigma_h(g) = g + h$ for all elements g in this group, and $\sigma_h(X) = X$ if there is an infinite element X. When the group is a direct product addition is componentwise. Let G^+ denote the group of all permutations σ_h. Thus the order of G^+ is either v or $v - 1$. If τ is a permutation of S, then $G^+(\tau)$ will denote the permutation group generated by τ and the elements of G^+.

$v = 91$ [17]. Here $b = 273$. I wrote a computer program to look for a (61,6,1)–BIBD with a cyclic automorphism group of order 61, using a very simple backtrack algorithm. It was not successful. However with a few minor modifications this program found the (91,6,1)–BIBD given here. It was a surprise that this BIBD had an automorphism of order 3. The elements are the integers modulo 91. The group A is the 273 element nonabelian group $G^+(\tau)$ where $\tau(h) = 16h$. There is a single base block, namely

$$1 \quad 2 \quad 4 \quad 8 \quad -4\cdot 13 \quad 16\cdot 13.$$

$v = 96$ [20]. Here $b = 304$. We take our set S to be Z(48)×Z(2). The group A is the 48 element cyclic group generated by σ where $\sigma(g,h) = (g+1,h)$. Our base blocks are the eight blocks

$(0,h)$	$(8,h)$	$(16,h)$	$(24,h)$	$(32,h)$	$(40,h)$	$h = 0,1$
$(0,0)$	$(1,0)$	$(3,0)$	$(13,0)$	$(28,0)$	$(0,1)$	
$(0,0)$	$(4,0)$	$(11,0)$	$(17,1)$	$(36,1)$	$(38,1)$	
$(0,0)$	$(5,0)$	$(19,0)$	$(1,1)$	$(24,1)$	$(42,1)$	
$(0,0)$	$(9,0)$	$(26,0)$	$(4,1)$	$(7,1)$	$(40,1)$	
$(0,0)$	$(6,0)$	$(8,1)$	$(9,1)$	$(18,1)$	$(22,1)$	
$(0,0)$	$(18,0)$	$(11,1)$	$(28,1)$	$(33,1)$	$(39,1)$.	

Each of the first two of these base blocks generates 8 blocks, and each of the other six base blocks generates 48 blocks.

$v = 106$ [18]. Here $b = 371$. We take our set S to be Z(53)×Z(2). The group A is the cyclic group of order 53 generated by σ where $\sigma(g,h) = (g+1,h)$. Our base blocks are the seven blocks

$(0,0)$	$(1,0)$	$(3,0)$	$(11,0)$	$(38,0)$	$(0,1)$
$(0,0)$	$(5,0)$	$(19,0)$	$(25,0)$	$(36,1)$	$(39,1)$
$(0,0)$	$(7,0)$	$(29,0)$	$(8,1)$	$(16,1)$	$(48,1)$
$(0,0)$	$(9,0)$	$(21,0)$	$(12,1)$	$(13,1)$	$(27,1)$
$(0,0)$	$(13,0)$	$(30,0)$	$(23,1)$	$(35,1)$	$(51,1)$
$(0,0)$	$(4,0)$	$(28,1)$	$(30,1)$	$(37,1)$	$(47,1)$
$(0,0)$	$(2,1)$	$(7,1)$	$(25,1)$	$(29,1)$	$(49,1)$.

Each of these seven base blocks generates exactly 53 blocks.

$v = 111$ [20]. Here $b = 407$. We take our set S to be GF(37)×GF(3). Our automorphism group is generated by σ and τ, where $\sigma(g,h) = (g+1,h)$ and $\tau(g,h) = (10g, h+1)$. The base blocks are

$(0,0)$	$(1,0)$	$(3,0)$	$(7,0)$	$(17,0)$	$(0,1)$
$(0,0)$	$(5,0)$	$(19,1)$	$(28,1)$	$(10,2)$	$(30,2)$
$(26,0)$	$(10\cdot 26,1)$	$(10^2\cdot 26,2)$	$(34,0)$	$(10\cdot 34,1)$	$(10^2\cdot 34,2)$
$(5,0)$	$(10\cdot 5,1)$	$(10^2\cdot 5,2)$	$(33,0)$	$(10\cdot 33,1)$	$(10^2\cdot 33,2)$
$(13,0)$	$(10\cdot 13,1)$	$(10^2\cdot 13,2)$	$(24,0)$	$(10\cdot 24,1)$	$(10^2\cdot 24,2)$
$(10,0)$	$(10\cdot 10,1)$	$(10^2\cdot 10,2)$	$(23,0)$	$(10\cdot 23,1)$	$(10^2\cdot 23,2)$
$(9,0)$	$(10\cdot 9,1)$	$(10^2\cdot 9,2)$	$(27,0)$	$(10\cdot 27,1)$	$(10^2\cdot 27,2)$.

Each of the first two base blocks generates 111 blocks. Each of the last five base blocks is fixed by τ and generates 37 base blocks.

$v = 121$ [17]. Here $b = 484$. We take our set S to be GF(121). We set $A = G^+$. We note that the elements of GF(121) can be written in the form $x + yi$ where

x and y are elements of GF(11) and $i^2 = -1$. We set $X_1 = 1 + 3i$, $Y_1 = 7 + 7i$, $X_2 = X_1^2 = 3 + 6i$, and $Y_2 = Y_1^2 = -i$. Our four base blocks are

$$0 \quad -X_h \cdot i^n \quad 2X_h \cdot i^n \quad -Y_h \cdot i^n \quad 2Y_h \cdot i^n \quad (4X_h + 6Y_h) \cdot i^n,$$

$h = 1, 2$; $n = 0, 1$. Each one generates exactly 121 blocks.

This design was originally found by a computer run. Upon analysis it was found to have the curious form described above.

$v = 126$ (Hanani [12]). Here $b = 525$. I have changed Hanani's notation. Here we let our set S be GF(5)×GF(25) and an infinite point X. Let δ be one of the square roots of 3 in GF(25), and let $\omega = 4\delta + 2$. Then δ is a primitive 8-*th* root of unity and ω is a primitive cube root of unity in GF(25). We set $A = G^+$, which has order 125. Here we have 5 base blocks. They are

$$X \quad (0,0) \quad (1,0) \quad (2,0) \quad (3,0) \quad (4,0)$$

and

$$(0, \delta^n) \quad (0, -\delta^n) \quad (2^{n^2}, \delta^n \omega) \quad (2^{n^2}, -\delta^n \omega) \quad (4 \cdot 2^{n^2}, \delta^n \omega^2) \quad (4 \cdot 2^{n^2}, -\delta^n \omega^2),$$

n = 0,1,2,3. The first of these base blocks generates 25 blocks and each of the other four base blocks generates 125 blocks, giving us the correct total.

There is a geometric construction of a (126,6,1)–BIBD, which uses a certain subset of GF(625). This was done first by Bose [5]. (See Stanton and Gryte [27].) Indeed this method gives a $(q^3 + 1, q + 1, 1)$–BIBD for any prime power q. McKay and Stanton [15] have shown that the BIBD obtained in this way is not isomorphic to Hanani's design given above.

$v = 136$ [21]. Here $b = 612$. Our set S consists of Z(45) × Z(3) and an infinite point X. Our automorphism group has order 135 and is generated by σ and τ, where $\sigma(X) = \tau(X) = X$, $\sigma(g, h) = (g + 1, h)$ and $\tau(g, h) = (16g, h + 1)$. There are 6 base blocks. They are

X	$(0,0)$	$(9,0)$	$(18,0)$	$(27,0)$	$(36,0)$
$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(16,1)$	$(16^2,2)$
$(0,0)$	$(3,0)$	$(15,0)$	$(35,0)$	$(6,2)$	$(10,2)$
$(0,0)$	$(22,0)$	$(11,1)$	$(30,1)$	$(1,2)$	$(18,2)$
$(0,0)$	$(5,0)$	$(18,1)$	$(41,1)$	$(13,2)$	$(42,2)$
$(0,0)$	$(11,0)$	$(17,0)$	$(4,2)$	$(5,2)$	$(28,2)$.

The first of these base blocks generates 27 blocks, the second 45 blocks, and each of the last four generates 135 blocks.

Since 141 is listed as unknown in Table 1, we have now proved Theorem 1 for all $v < 150$. There are 6 more values of v that are handled in a similar way. We give them next.

$v = 216$ [23]. Here $b = 1548$. Our set S consists of Z(215) and an infinite point X. We set $A = G^+(\tau)$ where $\tau(X) = X$ and $\tau(h) = 6h$ for h in Z(215). We note that 6 is a cube root of unity modulo 215, so that τ has order 3, and A has order 645. There are four base blocks:

$$\begin{array}{cccccc} X & 0 & 43 & 86 & 129 & 172 \\ 65 & 6\cdot 65 & 6^2\cdot 65 & 130 & 6\cdot 130 & 6^2\cdot 130 \\ 0 & 3 & 122 & 135 & 139 & 141 \\ 0 & 7 & 48 & 95 & 106 & 159. \end{array}$$

The first of these base blocks is fixed by both σ^{43} and τ so that it generates 43 blocks. The second is fixed by τ and generates 215 blocks. Each of the last two generates 645 blocks, giving us a total of 1548 blocks.

$v = 246$ (Greig [10]. Here $b = 2009$. Our set S consists of an infinite point X and the 245 elements of GF(5)×GF(49). We let $\omega = 3 + i$, where i is a square root of -1 in GF(49). Then ω is a primitive element of GF(49). We set $A = G^+$. There are nine base blocks:

$$X \quad (0,0) \quad (1,0) \quad (2,0) \quad (3,0) \quad (4,0),$$

the four base blocks

$$(0,\omega^a) \quad (0,-\omega^a) \quad (1,\omega^{4+a}) \quad (1,-\omega^{4+a}) \quad (4,\omega^{17+a}) \quad (4,-\omega^{17+a}),$$

where $a = 0, 3, 12, 15$, and the four base blocks

$$(0,\omega^{22+a}) \quad (0,-\omega^{22+a}) \quad (2,\omega^{11+a}) \quad (2,-\omega^{11+a}) \quad (3,\omega^{18+a}) \quad (3,-\omega^{18+a}),$$

where again $a = 0, 3, 12, 15$. The first of these base blocks generates 49 blocks, and each of the other eight generates 245 blocks.

$v = 301$ [23]. Here $b = 3010$. Our set S is Z(301). We set $A = G^+(\tau)$ where $\tau(h) = 135h$. We note that 135 is a cube root of unity modulo 301, so that A has order 903. There are four base blocks:

$$\begin{array}{cccccc} 24 & 135\cdot 24 & 135^2\cdot 24 & 81 & 135\cdot 81 & 135^2\cdot 81 \\ 0 & 2 & 43 & 46 & 47 & 130 \\ 0 & 5 & 20 & 139 & 174 & 216 \\ 0 & 69 & 88 & 120 & 268 & 276. \end{array}$$

The first of these base blocks is fixed by τ and generates 301 blocks. Each of the other three generates 903 blocks, giving us a total of 3010 blocks.

$v = 306$. Here $b = 3111$. This design was constructed by Hanani, and it appears in an unpublished addendum to [12]. The set S consists of GF(5)×GF(61) and an infinite point X. We set $A = G^+$. There are 11 base blocks:

$$X \quad (0,0) \quad (1,0) \quad (2,0) \quad (3,0) \quad (4,0)$$

and the 10 base blocks

$$(0,8^n) \quad (0,-8^n) \quad (2^{n^2},16\cdot 8^n) \quad (2^{n^2},-16\cdot 8^n)$$
$$(4\cdot 2^{n^2},19\cdot 8^n) \quad (4\cdot 2^{n^2},-19\cdot 8^n),$$

$0 \le n < 10$. The first of these base blocks generates 61 blocks and each of the other ten base blocks generates 305 blocks, giving us the correct total.

$v = 486$ (Greig [10]). Here $b = 7857$. Our set S consists of an infinite point X and the 485 elements of GF(5)×GF(97). We set $A = G^+$. There are 17 base blocks:

$$X \quad (0,0) \quad (1,0) \quad (2,0) \quad (3,0) \quad (4,0),$$

the 16 base blocks

$$(0,5^{3n}) \quad (0,-5^{3n}) \quad (2^{n^2},5^{3n+5}) \quad (2^{n^2},-5^{3n+5}$$
$$(4\cdot 2^{n^2},5^{3n+19}) \quad (4\cdot 2^{n^2},-5^{3n+19}),$$

$0 \le n < 16$. The first of these base blocks generates 97 blocks, and each of the other 16 generates 485 blocks.

$v = 511$ [23]. Here $b = 8687$. Our set S is Z(511). We set $A = G^+(\tau)$ where $\tau(h) = 373h$. We note that 373 is a cube root of unity modulo 511, and A has order 1533. There are seven base blocks:

132	$373\cdot 132$	$373^2\cdot 132$	149	$373\cdot 149$	$373^2\cdot 149$
24	$373\cdot 24$	$373^2\cdot 24$	126	$373\cdot 126$	$373^2\cdot 126$
0	1	73	77	148	222
0	2	152	293	296	459
0	181	225	239	259	295
0	86	171	255	261	368
0	99	120	125	231	443.

The first two of these base blocks are fixed by τ and generate 511 blocks each. Each of the other five generates 1533 blocks.

Success in the cases $v = 216, 301, 511$ was possible only because we were able to use a automorphism group with order larger than v.

In the next two sections we will discuss a number of other well known types of combinatorial designs that we will need to complete the induction proof of Theorem 1.

3 TRANSVERSAL DESIGNS AND INCOMPLETE TRANSVERSAL DESIGNS

A group divisible design GDD$[a^\alpha b^\beta \cdots d^\delta; K; \lambda]$ is a triple (V, G, B) where

(i) V is a finite set, and K is a collection of positive integers;

(ii) G is a family of disjoint subsets of V, called groups, whose union is V;

(iii) exactly α of the groups have size a, exactly β of the groups have size b,..., exactly δ of the groups have size d, and there are no other groups;

(iv) B is a family of subsets of V, called blocks, and the size of every block in B is an element of the set K;

(v) each block contains at most one element from any given group; and

(vi) every pair of distinct elements of V belongs to exactly one of the groups or to exactly λ of the blocks.

When $\lambda = 1$ and K consists of a single integer k we will write GDD$[a^\alpha \cdots d^\delta; k]$.

A transversal design TD(k, n) is a GDD$[n^k; k]$. Such a design is equivalent to a set of $k - 2$ mutually orthogonal Latin squares of order n. We shall need the following three known results:

Theorem 2. (MacNeish [14]). *Let n be written as the product of prime powers, say $n = q_1 q_2 \cdots q_t$, where the q_i are powers of distinct primes. Let $q = \min\{q_i\}$. Then a TD$(q + 1, n)$ exists.*

Theorem 3. *Let n be an integer greater than 4, $n \notin \{6, 10, 14, 18, 22, 34, 42\}$. Then a $TD(6, n)$ exists.*

Theorem 4. *If $n > 62$ then a $TD(7, n)$ exists. Furthermore a $TD(7, 12)$ exists.*

A proof of Theorem 2 can be found in Hanani [12], Hall [6], or Beth, Jungnickel, and Lenz [4].

Theorem 3 is the result of work by a number of authors. Most of the TD$(6, v)$s can be found in Beth, Jungnickel, and Lenz [4]. The remaining ones can be found in Abel [1], Abel and Todorov [3], Colbourn [8], Mills [19], and Roberts [26].

For Theorem 4 the result for $n > 62$ is due to Hanani [12], and the result for $n = 12$ is due to Johnson, Dulmage, and Mendelsohn [13].

An incomplete transversal design TD$(k, n)-$TD(k, h) is defined to be a quadruple (X, Y, G, B) such that

(i) X is a kn-element set;

(ii) G is a family of disjoint n-element subsets $G_1, G_2, \ldots G_k$, called groups, whose union is X;

(iii) Y is a subset of X, called the hole, satisfying $|Y \cap G_i| = h$ for all i;

(iv) B is a family of k-elements subsets of X, called blocks, such that each block intersects each group in exactly one point,

(v) no block contains more than one element of Y; and

(vi) every pair of distinct elements of distinct groups, not both in Y, belongs to exactly one block.

Such a design is equivalent to a set of mutually orthogonal Latin squares of order n with a common hole of size $h \times h$. For our constructions we need incomplete transversal designs TD$(6, n)$ – TD$(6, h)$ for 72 ordered pairs (n, h). These are given in Table 2. We make use of 8 different constructions, which we denote by S, T, U, V, W, X, Y, Z. These 8 constructions are based on ideas of Wilson [30]. The second column of Table 2 gives us the construction, and the last column, the Par column, gives us the parameters involved, which differ

Table 2
Incomplete transversal designs
$TD(6,n) - TD(6,h)$

n,h	C	Par
15,3	R	
16,3	R	
28,3	R	
29,4	R	
68,3	V	9,2
72,2	U	9,7,7
96,11	T	8,8
99,14	W	13,1
103,18	X	7,1,6
103,3	U	13,7,9
137,12	T	11,5
140,15	U	16,7,13
141,16	T	8,13
143,18	W	17,13
146,21	Y	11,5,3
156,6	U	19,7,17
159,4	U	13,11,12
164,14	U	19,7,17
167,17	U	19,7,17
168,18	U	19,7,17
168,3	S	11,15
170,15	W	23,1
171,16	W	23,1
172,22	Z	13,7,3

n,h	C	Par
175,25	T	7,0
176,21	W	23,1
178,28	Z	13,7,5
183,28	Z	13,12,5
188,3	V	25,10
197,12	T	16,5
216,11	T	19,7
225,20	U	27,7,16
229,24	U	27,7,16
231,26	U	27,7,16
248,3	S	7,35
267,22	U	31,7,28
276,31	T	8,28
279,14	W	37,13
292,7	S	19,15
293,48	X	19,17,16
293,8	S	19,15
305,20	U	37,7,26
316,16	S	25,12
318,18	S	25,12
321,16	T	20,1
364,9	U	47,7,26
371,21	U	47,7,21
381,21	U	47,7,31

n,h	C	Par
387,12	S	25,15
387,7	S	19,20
388,28	U	47,7,31
396,41	U	47,7,26
402,12	S	13,30
419,14	S	27,15
432,12	S	12,35
435,15	U	53,7,49
441,16	S	17,25
483,3	S	32,15
494,9	T	54,8
500,15	U	61,7,58
501,16	T	31,5
591,16	S	25,23
594,49	U	71,7,48
682,7	S	27,25
693,58	U	81,7,68
707,17	S	23,30
711,21	S	23,30
819,9	S	27,30
973,63	T	15,28
981,16	T	61,5
1032,17	S	29,35
1168,63	U	139,7,132

from construction to construction. In all cases the necessary transversal designs exist by Theorems 2, 3, and 4.

An R in the Par column means that the incomplete transversal design is one of four special designs. The $TD(6,15) - TD(6,3)$ was constructed by Colbourn (unpublished) and the $TD(6,16) - TD(6,3)$ appears in the appendix of Brouwer and van Rees [6]. The $TD(6,28) - TD(6,3)$ and the $TD(6,29) - TD(6,4)$ were found by R. C. Mullin. The former appears in [24].

Constructions S and T. Suppose that $n = tm+a$ where t and m, are positive integers and a is a nonnegative integer such that $a \leq t$, and the transversal designs TD$(7,t)$, TD$(6,m)$ and TD$(6,m+1)$ exist. Then we have

(i) The incomplete transversal design TD$(6,n)$ – TD$(6,a)$ exists.

(ii) If a TD$(6,a)$ exists then the incomplete transversal design TD$(6,n)$ – TD$(6,t)$ exists.

When (i) holds this is construction S. For the 19 pairs (n,h) for which we need this construction the values of t and m are given, in that order, in the Par column of Table 2, and of course $a = h$. When (ii) holds this is construction T. For the 12 pairs (n,h) for which we need this construction the values of m and a are given, in that order, in the Par column of Table 2, and of course $t = h$.

Constructions S and T are special cases of Proposition 3.4 of Brouwer and van Rees [6].

Construction U. Suppose that $n = tm + a + h$ where t, m, a, and h, are positive integers such that $t \geq a$, $t \geq h$, and the transversal designs TD(6,a), TD(6,m), TD(6,m+1), TD(6,m+2), and TD$(8,t)$ exist. Then the incomplete transversal design TD$(6,n)$ – TD$(6,h)$ exists. This is construction U. For the 23 pairs (n,h) for which we need this construction the values of t, m, and a are given, in that order, in the Par column of Table 2.

The proof of Construction U is similar to the proof of Construction S. It is obtained by making the obvious modification to the proof of Theorem 2.4 of Wilson [30]. See also Brouwer and van Rees [6].

Constructions V and W. Suppose that $n = 7t + a + b$ where t, a, and b are nonnegative integers such that $t > a$, and the transversal design TD$(b+7,t)$ exists. Then

(i) if a TD$(6,b+7)$ exists then a TD$(6,n)$ – TD$(6,a)$ exists.

(ii) if a TD$(6,a)$ exists, then a TD$(6,n)$ – TD$(6,b+7)$ exists.

When (i) holds we have Construction V. Here the parameters given in Table 2 are t and b in that order. When (ii) holds we have construction W. Here the parameters given are t and a in that order, and $b = h - 7$.

Constructions V and W can be obtained by making minor modifications to the proof of Theorem 2.2 of Wilson [30].

In Constructions V and W the integer 7 can be replaced by any integer m such that the transversal designs TD$(6, m)$, TD$(6, m+1)$, and TD$(6, m+2)$ exist. However we do not need these constructions for $m \neq 7$.

Constructions X, Y, and Z. (Brouwer and van Rees [6]) Suppose that $n = 12t + a + 3b$ where t, a, and b are positive integers such that $t \geq a$, $t \geq b$, and the transversal design TD$(6, a)$ exists, then

(i) if a TD$(8, t)$ exists then a TD$(6, n)$ − TD$(6, 3b)$ exists,

(ii) if a TD$(b+7, t)$ exists and $t > a$, then a TD$(6, n)$ − TD$(6, 12+3b)$ exists, and

(iii) if a TD$(b+7, t)$ exists, then a TD$(6, n)$ − TD$(6, 13+3b)$ exists.

When (i) holds we have Construction X, when (ii) holds we have Construction Y, and when (iii) holds we have Construction Z. In all three cases the parameters given in Table 2 are t, a, and b in that order.

To make these last three constructions work we need the incomplete transversal designs TD$(6, 15)$ − TD$(6, 3)$ and TD$(6, 16)$ − TD$(6, 3)$, which are listed in Table 2.

It is easily checked that for each of the last 68 pairs n, h listed in Table 2 the parameter satisfy the required conditions. Thus we have the following result:

Theorem 5. *The incomplete transversal design TD$(6, n)$ − TD$(6, h)$ exists for each of the 72 ordered pairs n, h listed in Table 2.*

4 PAIRWISE BALANCED DESIGNS

It is convenient to generalize the definition of a BIBD by allowing the blocks to have different sizes. We call the result a pairwise balanced design or PBD. More precisely let K be a set of positive integers. Then a (v, K, λ)–PBD is a pair (S, B) where

(i) S is a v-element set;

(ii) B is a collection of subsets of S called blocks,

(iii) the number of elements in any of the blocks in B belongs to K, and

(iv) every pair of distinct elements of S appears in exactly one block.

We shall need a number of PBDs in which all blocks except one have size 6. Thus a $(v, \{6, h^*\}, 1)$–PBD is a PBD in which there is one block of size h and the remaining blocks all have size 6. When $h > 6$ the block of size h is sometimes called the large block. We allow the possibilities $h = 0$, 1, and 6, in which cases our PBD becomes a BIBD.

A parallel class of blocks is a set of disjoint blocks, whose union is the basic set S. A BIBD is called resolvable if its blocks can be broken up into disjoint parallel classes. In this case we say we have an RBIBD.

5 CONSTRUCTIONS

We need constructions for the 77 PBDs listed in Table 3 and the 837 BIBDs listed in Table 4. In this section we describe these constructions. Theorem 6 and Constructions D through N are due, in essence, to Hanani [12] and Wilson [29]. Construction O is due, in essence, to Mullin, Hoffman, and Lindner [25]. Construction P is due to A. E. Brouwer (unpublished). A generalization can be found in Mullin, Hoffman, and Lindner [25]. Construction Q is due to Zhu and Miao [16].

Construction A. An A in the C column of Table 4 means that this BIBD was discussed in §1 or constructed in §2.

Construction B. A B in the C column of Table 3 means that this PBD is given in [2].

Construction D. This construction is based on $(w, 5, 1)$–RBIBDs. Suppose we have such an RBIBD. This RBIBD has h parallel classes, where $h = (w-1)/4$. We adjoin h additional elements x_i, $1 \leq i \leq h$, to our RBIBD. We adjoin x_i to all the blocks of the ith parallel class and add a new block $\{x_1, x_2, \ldots, x_h\}$.

Table 3
Pairwise balanced designs
$(v, \{6, h^*\}, 1)$–PBDs

v, h	C	Par
81,16	D	65
106,21	D	85
141,16	B	
156,31	F	31
171,21	Q	31,6,3
176,26	Q	31,6,2
181,31	L	31,1
186,31	L	31,0
201,16	B	
256,51	D	205
306,61	D	245
331,66	F	66
331,31	F	66
336,31	B	
356,71	D	285
366,66	L	66,6
366,31	M	66,6,6
381,31	F	76
391,66	L	66,1
391,31	M	66,1,6
396,66	L	66,0
406,81	D	325
406,16	E	81
421,66	B	
436,16	Q	76,6,4
451,76	L	76,1

v, h	C	Par
456,76	L	76,0
456,31	F	91
506,101	D	405
516,31	M	91,6,6
576,96	L	96,0
606,31	F	121
621,21	Q	106,6,3
636,106	L	106,0
636,31	M	111,6,6
636,21	E	106
681,136	F	136
711,106	B	
711,21	E	106
721,31	M	121,1,6
726,91	B	
726,31	M	126,6,6
736,111	B	
811,136	L	136,1
816,136	L	136,0
831,76	B	
876,66	O	81,16,13,1
891,136	B	
936,91	O	81,16,13,1
951,51	Q	171,21,15
976,66	M	66,1,15
981,51	Q	186,31,27

v, h	C	Par
991,81	M	81,16,15
991,16	E	81
996,171	L	171,6
996,31	M	171,6,6
996,21	E	171
1011,21	Q	171,6,3
1021,171	L	171,1
1021,21	E	171
1036,106	Q	186,31,16
1036,21	E	106
1126,201	L	201,16
1126,16	E	201
1131,76	B	
1161,106	B	
1161,21	E	106
1171,66	M	66,1,18
1176,121	B	
1191,136	B	
1276,171	O	81,16,17,6
1276,91	O	81,16,17,6
1341,106	O	81,16,19,1
1341,21	E	106
1366,261	O	81,16,17,6
1461,106	B	
1491,136	B	

This gives us a $(w + h, \{6, h^*\}, 1)$–PBD. For these designs the parameter w is listed in the Par column. □

Inspecting Table 3 we see that we need $(w, 5, 1)$–RBIBDs for $w = 65$, 85, 205, 245, 285, 325, and 405. In 1991 Zhu, Du, and Zhang [33] published a list of 113 values of w such that for any w not on this list, $w \equiv 5 \pmod{20}$, such a $(w, 5, 1)$–RBIBD is known to exist. Of the six values of w for which we need

such a design only 245 and 285 are on this list. However M. Greig [10] has shown that if q is a prime such that $q \equiv 1 \pmod{10}$ and $60 < q < 2000$, then a $(4q + 1, 5, 1)$–RBIBD exists. This take care of 245 and 285.

Construction E. These are designs constructed from a $(v, \{6, w^*\}, 1)$–PBD and a $(w, 6, 1)$-BIBD or a $(w, \{6, f^*\}, 1)$–PBD by replacing the block of size w in the former by the blocks of the latter. In this case the parameter w is given in the Par column.

We use Construction E 21 times. Each time the two designs that we need are given in Tables 3 and 4.

Next we prove a Theorem that includes our next 6 constructions.

Theorem 6. *Suppose that a $(w, K, 1)$–PBD exists, which we will call $\mathcal{P}$, and that there a $(5k + 1, 6, 1)$–BIBD for all $k \in K$. Then there exists a $(5w + 1, 6, 1)$–BIBD. Furthermore if B is a block of size k in $\mathcal{P}$, then there exists a $(5w + 1, \{6, (5k + 1)^*\}, 1)$–PBD. Finally if all blocks in $\mathcal{P}$ have size 6, then a $(5w + 1, \{6, w^*\}$–PBD exists.*

Proof. We take the $(w, K, 1)$–PBD and we replace each point α of this design by a five point set P_α containing α and four new points. This is called inflating the point α by a factor of 5. Now we adjoin one additional point x. This gives us a $5k + 1$ point set which contains all the points of the original PBD. Let $Q_\alpha = P_\alpha \cup \{x\}$. For each block B of size k of our original PBD we consider the $5k+1$ point set consisting of the distinct points in the k sets Q_b, $b \in B$. On this set we construct $(5k+1, 6, 1)$–BIBD in such a way that each of the Q_b, $b \in B$, is a block. We now have a collection of blocks in which each Q_b occurs a number of times. We simply remove duplications and we have our $(5w+1, 6, 1)$–BIBD. If B is a block of $\mathcal{P}$ that has size k, we remove all the blocks that are obtained from this particular block and replace them with a single block of size $5k + 1$, to obtain a $(5w + 1, \{6, \{6, (5k + 1)^*\}, 1)$–PBD. Finally if all the blocks of $\mathcal{P}$ have size 6 we can choose our $(31, 6, 1)$–BIBDs in such a way that all blocks of $\mathcal{P}$ occur in our $(5w+1, 6, 1)$–BIBD. Removing these blocks and replacing them be a single block of size w, we obtain a $(5w + 1, \{6, w^*\}$–PBD. □

Construction F. Suppose that a $(w, 6, 1)$–BIBD exists. Then a $(5w+1, 6, 1)$–BIBD, a $(5w + 1, \{6, 31^*\}, 1)$–PBD, and a $(5w + 1, \{6, w^*\}, 1)$–PBD all exist. Here w is given in the Par column.

Construction F is obtained by applying Theorem 6 to the $(w, 6, 1)$–BIBD. □

Table 4
Constructions of $(v, 6, 1)$–BIBDs

v	C	Par
1	A	
6	A	
31	A	
66	A	
76	A	
91	A	
96	A	
106	A	
111	A	
121	A	
126	A	
136	A	
151	N	7
156	F	31
171	A	
181	L	31,1
186	L	31,0
211	N	9
216	A	
241	N	13
246	A	
271	N	9
301	A	
306	A	
331	F	66
336	E	31
361	N	$3+i$
366	L	66,6
381	F	76
391	L	66,1
396	L	66,0
421	Q	81,16,13
426	L	76,6
451	L	76,1
456	F	91

v	C	Par
481	F	96
486	A	
511	A	
516	L	91,6
531	F	106
541	L	91,1
546	L	91,0
556	F	111
571	L	96,1
576	L	96,0
586	Q	106,21,10
601	Q	106,21,7
606	F	121
621	Q	106,21,3
631	F	126
636	L	106,0
661	L	111,1
666	L	111,0
681	F	136
691	N	60
696	L	121,6
711	E	106
721	L	121,1
726	L	121,0
736	E	111
751	L	126,1
756	F	151
781	F	156
786	L	136,6
811	L	136,1
816	L	136,0
826	Q	141,16,4
831	E	76
841	Q	141,16,1
846	H	13,13

v	C	Par
856	F	171
861	Q	156,31,15
871	Q	156,31,13
876	L	151,6
886	Q	156,31,10
891	E	136
901	L	151,1
906	F	181
921	L	171,21
931	F	186
936	L	156,0
951	O	81,16,13,1
961	L	186,31
966	O	81,16,13,1
976	M	66,1,15
981	O	81,16,13,1
991	Q	171,21,7
996	L	171,6
1006	Q	171,21,4
1011	Q	171,21,3
1021	L	171,1
1026	L	171,0
1036	Q	176,26,4
1041	Q	186,31,15
1051	Q	176,26,1
1056	F	211
1066	Q	186,31,10
1071	Q	181,31,3
1081	F	216
1086	L	181,0
1101	Q	186,31,3
1111	K	17,13
1116	L	186,0
1126	M	106,21,13
1131	E	76

Table 4 (continued)

v	C	Par
1141	Q	201,16,13
1146	M	66,6,19
1156	P	9,3,1
1161	E	106
1171	M	66,1,18
1176	E	121
1186	Q	201,16,4
1191	E	136
1201	O	81,16,17,6
1206	F	241
1216	O	81,16,17,6
1231	F	246
1236	H	19,13
1261	L	211,1
1266	L	211,0
1276	O	81,16,17,6
1281	K	17,15
1291	L	216,1
1296	L	216,0
1306	M	106,6,13
1311	O	81,16,19,1
1321	O	81,16,17,6
1326	M	66,6,22
1336	Q	256,51,40
1341	O	81,16,19,1
1351	M	76,1,18
1356	F	271
1366	M	66,1,21
1371	M	111,6,13
1381	Q	256,51,31
1386	O	81,16,19,1
1401	Q	256,51,27
1411	Q	256,51,25
1416	L	241,6
1426	H	19,15
1431	M	66,1,22
1441	L	241,1
1446	L	241,0
1461	E	106
1471	L	246,1
1476	L	246,0
1491	E	136
1501	K	23,13
1506	F	301
1516	P	12,3,1
1531	F	306
1536	K	17,18
1546	Q	306,61,58
1551	M	106,21,18
1561	M	66,1,24
1566	M	126,6,13
1576	M	76,1,21
1581	M	111,6,15
1591	O	81,16,23,6
1596	L	271,6
1606	O	81,16,23,6
1621	L	271,1
1626	H	25,13
1636	M	106,21,19
1641	Q	306,61,39
1651	M	76,1,22
1656	F	331
1666	O	81,16,23,6
1681	F	336
1686	Q	306,61,30
1696	M	136,6,13
1701	O	81,16,25,1
1711	G	19,0,18
1716	G	19,1,18
1726	Q	331,66,52
1731	K	23,15
1741	G	19,6,18
1746	O	81,16,25,1
1756	H	27,13
1761	O	81,16,25,1
1771	Q	306,61,13
1776	G	19,13,18
1786	G	19,15,18
1791	M	91,6,21
1801	G	19,18,18
1806	H	19,19
1816	Q	356,71,64
1821	Q	356,71,63
1831	F	366
1836	L	306,0
1846	O	81,16,27,1
1861	L	336,31
1866	L	366,66
1876	H	25,15
1881	Q	356,71,51
1891	K	29,13
1896	M	111,6,18
1906	F	381
1911	Q	331,31,15
1921	Q	331,31,13
1926	O	81,16,25,1
1936	O	81,16,27,1
1941	Q	336,31,15
1951	M	66,1,30
1956	F	391
1966	O	81,16,27,1
1971	M	171,21,13
1981	F	396
1986	L	331,0
1996	M	96,1,21
2001	M	111,6,19
2011	L	336,1
2016	H	31,13

Table 4 (continued)

v	C	Par
2026	H	27,15
2041	L	366,31
2046	L	396,66
2056	O	81,16,27,1
2061	M	106,21,24
2071	O	81,16,29,6
2076	K	23,18
2086	O	81,16,27,1
2091	M	96,1,22
2101	J	23,18
2106	F	421
2116	O	81,16,27,1
2121	O	81,16,31,1
2131	F	426
2136	J	23,18
2146	J	23,18
2151	M	171,6,13
2161	J	23,18
2166	J	23,18
2176	J	23,18
2181	K	29,15
2191	K	23,19
2196	L	366,0
2206	M	106,6,22
2211	M	66,1,34
2221	O	81,16,29,6
2226	M	66,6,37
2236	Q	381,31,10
2241	Q	421,66,57
2266	I	25,18
2271	I	25,18
2286	I	25,18
2296	Q	391,31,10
2301	M	91,6,27
2311	M	106,1,22
2331	Q	391,31,3

v	C	Par
2376	H	25,19
2386	Q	451,76,64
2391	Q	456,76,69
2401	Q	421,66,25
2406	H	37,13
2416	Q	406,16,4
2421	K	23,21
2446	I	27,18
2451	I	27,18
2466	I	27,18
2476	I	27,18
2481	J	23,21
2491	J	23,21
2511	J	23,21
2566	H	27,19
2571	M	106,21,30
2581	F	516
2586	M	66,6,43
2596	Q	436,16,4
2601	Q	506,101,87
2611	Q	436,16,1
2616	K	29,18
2641	I	25,21
2646	I	25,21
2661	Q	456,31,15
2671	K	41,13
2676	J	29,18
2686	J	29,18
2706	J	29,18
2751	H	25,22
2761	K	29,19
2766	K	23,24
2776	H	37,15
2781	F	556
2806	I	31,18
2811	I	31,18

v	C	Par
2826	I	31,18
2851	I	31,18
2871	M	171,21,19
2971	H	27,22
2976	L	576,96
2986	Q	516,31,22
2991	Q	576,96,93
3016	Q	516,31,16
3021	Q	516,31,15
3036	L	511,6
3046	O	81,16,41,6
3051	K	29,21
3061	K	47,13
3081	K	41,15
3126	H	25,25
3136	J	29,21
3141	J	29,21
3151	J	29,21
3156	J	29,21
3166	J	29,21
3171	J	29,21
3181	J	29,21
3186	H	49,13
3196	K	29,22
3201	O	141,16,25,1
3211	O	81,16,41,6
3216	L	541,6
3226	H	43,15
3231	M	96,1,34
3276	I	31,21
3291	O	81,16,49,1
3301	I	31,21
3411	I	37,18
3471	M	171,6,21
3496	O	141,16,27,1
3516	H	37,19

Table 4 (continued)

v	C	Par
3526	O	141,16,27,1
3531	K	47,15
3541	O	81,16,47,6
3546	J	29,24
3556	J	29,24
3561	Q	606,31,15
3571	J	29,24
3576	J	29,24
3586	J	29,24
3591	J	29,24
3601	J	29,24
3606	J	29,24
3616	J	29,24
3621	L	621,21
3631	K	29,25
3636	L	606,0
3646	H	27,27
3651	Q	681,136,87
3661	L	636,31
3666	M	66,6,61
3676	H	49,15
3681	F	736
3691	O	81,16,47,6
3696	K	41,18
3706	M	106,6,37
3711	L	636,21
3736	I	31,24
3741	O	81,16,49,1
3756	I	31,24
3766	J	41,18
3771	M	171,21,25
3781	J	41,18
3801	J	41,18
3861	J	41,18
3906	I	43,18
3921	K	29,27

v	C	Par
3931	I	43,18
4071	H	37,22
4086	H	43,19
4096	O	156,31,27,1
4101	M	106,21,48
4111	O	81,16,53,6
4116	L	691,6
4126	O	81,16,53,6
4131	F	826
4141	L	691,1
4146	L	691,0
4156	F	831
4161	L	711,21
4171	L	696,1
4176	L	696,0
4186	H	31,27
4191	Q	726,91,33
4201	L	726,31
4206	F	841
4216	L	816,136
4221	Q	811,136,129
4231	F	846
4236	K	47,18
4246	Q	711,21,4
4251	M	126,1,34
4261	J	47,18
4266	L	711,0
4276	Q	721,31,10
4281	F	856
4291	M	391,66,13
4296	J	47,18
4306	J	47,18
4311	K	41,21
4321	J	47,18
4326	J	47,18
4336	J	47,18

v	C	Par
4341	J	47,18
4351	J	47,18
4356	H	67,13
4366	J	47,18
4371	J	41,21
4381	J	47,18
4386	J	47,18
4396	J	41,21
4401	J	47,18
4426	I	49,18
4431	I	49,18
4456	I	49,18
4461	J	41,21
4491	I	49,18
4711	L	786,1
4731	H	43,22
4791	O	81,16,61,1
4806	H	31,31
4816	M	136,6,37
4821	O	81,16,61,1
4831	F	966
4836	J	53,18
4846	J	53,18
4851	O	81,16,61,1
4861	J	53,18
4866	J	53,18
4876	J	53,18
4881	J	53,18
4891	J	53,18
4896	J	53,18
4906	J	53,18
4911	O	81,16,67,1
4921	J	53,18
4926	J	53,18
4936	M	151,6,34
4941	J	53,18

Table 4 (continued)

v	C	Par
4951	J	53,18
4956	J	53,18
4966	J	47,21
4971	Q	876,66,57
4981	J	53,18
4986	J	53,18
4996	H	37,27
5001	J	47,21
5011	J	53,18
5016	J	53,18
5026	H	67,15
5031	J	47,21
5041	K	53,19
5046	J	47,21
5056	J	47,21
5061	J	47,21
5071	J	47,21
5076	J	41,24
5086	J	47,21
5091	J	47,21
5101	J	41,24
5106	J	47,21
5116	J	47,21
5121	J	47,21
5131	K	41,25
5136	H	79,13
5181	F	1036
5196	I	49,21
5206	F	1041
5376	H	43,25
5391	H	49,22
5401	K	83,13
5406	J	59,18
5416	J	59,18
5421	J	59,18
5431	J	59,18

v	C	Par
5436	J	59,18
5446	J	59,18
5451	L	951,51
5461	J	59,18
5466	J	59,18
5476	H	73,15
5481	J	59,18
5506	I	61,18
5511	I	61,18
5526	I	61,18
5536	I	61,18
5541	K	41,27
5571	I	61,18
5631	J	53,21
5691	J	53,21
5716	J	53,21
5746	J	53,21
5751	J	53,21
5766	J	47,24
5776	J	53,21
5781	J	53,21
5796	H	61,19
5806	H	43,27
5811	J	53,21
5821	J	47,24
5826	J	47,24
5836	K	53,22
5841	Q	976,66,3
5851	J	47,24
5856	J	47,24
5866	L	991,16
5871	L	996,21
5896	I	49,24
5901	Q	996,31,15
5916	I	49,24
5926	H	79,15

v	C	Par
5931	I	49,24
5941	I	49,24
5961	L	1011,21
6021	L	1021,21
6046	I	67,18
6076	I	67,18
6081	F	1216
6111	I	67,18
6171	O	81,16,79,1
6196	Q	1036,21,4
6226	J	59,21
6231	K	83,15
6256	J	41,30
6261	J	59,21
6286	J	59,21
6291	H	37,34
6306	H	97,13
6316	J	59,21
6321	J	59,21
6346	J	59,21
6351	J	59,21
6366	H	67,19
6376	J	59,21
6381	J	59,21
6391	J	53,24
6396	K	71,18
6421	I	61,21
6426	I	61,21
6441	J	59,21
6466	J	71,18
6486	I	61,21
6711	H	61,22
6801	L	1276,171
6826	M	451,76,18
6846	I	73,18
6856	F	1371

Table 4 (continued)

v	C	Par
6861	L	1161,21
6886	O	81,16,89,6
6891	L	1366,261
6916	M	456,76,18
6921	O	201,16,37,1
6936	H	73,19
6946	O	81,16,89,6
6951	O	201,16,37,1
6961	K	107,13
6966	L	1161,0
6976	K	41,34
6981	O	176,26,41,6
6991	O	81,16,89,6
6996	L	1171,6
7006	F	1401
7011	Q	1171,66,3
7021	L	1171,1
7026	L	1171,0
7051	I	67,21
7056	I	67,21
7071	O	171,21,43,1
7081	I	67,21
7086	H	109,13
7096	O	81,16,89,6
7311	H	43,34
7371	H	67,22
7401	I	79,18
7516	J	41,36
7546	J	83,18
7551	J	71,21
7576	J	83,18
7581	K	101,15
7596	H	49,31
7606	J	83,18
7611	J	71,21
7626	H	61,25

v	C	Par
7636	J	71,21
7641	J	83,18
7651	J	83,18
7656	J	83,18
7681	I	73,21
7686	I	73,21
7701	J	71,21
7711	I	73,21
7716	I	73,21
7726	H	103,15
7941	L	1341,21
7956	H	43,37
7981	J	53,30
7986	L	1336,6
8011	L	1336,1
8016	K	89,18
8031	H	73,22
8056	I	67,24
8061	J	53,30
8076	I	67,24
8086	J	89,18
8091	I	67,24
8101	I	67,24
8121	J	89,18
8181	J	89,18
8206	F	1641
8236	H	61,27
8241	O	81,16,103,1
8266	M	106,21,97
8271	O	81,16,109,1
8316	I	79,21
8331	H	49,34
8356	O	81,16,107,6
8376	I	79,21
8436	I	79,21
8461	I	79,21

v	C	Par
8491	I	79,21
8496	I	79,21
8521	I	79,21
8526	I	79,21
8551	I	79,21
8556	I	79,21
8571	I	79,21
8581	I	79,21
8586	I	79,21
8611	J	71,24
8616	J	71,24
8641	J	71,24
8646	J	71,24
8671	J	71,24
8691	H	79,22
8701	J	71,24
8706	J	71,24
8716	M	136,6,67
8721	K	83,21
8746	I	97,18
8751	I	97,18
8776	I	97,18
8781	J	83,21
8811	I	97,18
8961	J	83,21
8986	J	83,21
9016	J	83,21
9021	I	97,18
9046	H	67,27
9051	J	83,21
9076	H	121,15
9081	J	83,21
9171	I	61,30
9196	J	101,18
9231	F	1846
9291	I	103,18

Table 4 (continued)

v	C	Par
9316	I	103,18
9351	I	103,18
9411	J	89,21
9436	J	89,21
9466	J	89,21
9471	J	101,18
9496	I	79,24
9501	J	89,21
9526	H	127,15
9531	I	79,24
9561	I	103,18
9621	O	81,16,121,1
9646	J	89,21
9676	J	89,21
9681	J	89,21
9706	J	107,18
9711	J	89,21
9736	J	107,18
9741	I	79,24
9766	J	107,18
9771	J	89,21
9796	K	89,22
9801	J	107,18
9826	I	109,18
9831	I	109,18
9856	I	109,18
9891	I	109,18
9951	M	891,136,13
9976	J	47,42
10006	J	47,42
10011	J	107,18
10036	J	83,24
10041	J	47,42
10066	J	83,24
10071	I	67,30
10086	I	109,18

v	C	Par
10096	J	83,24
10101	I	109,18
10131	J	83,24
10246	J	113,18
10536	I	97,21
10561	I	97,21
10591	J	113,18
10596	H	163,13
10621	J	113,18
10626	J	113,18
10651	J	113,18
10656	J	113,18
10671	H	97,22
10681	J	113,18
10686	J	113,18
10696	J	101,21
10701	J	113,18
10711	J	113,18
10716	J	113,18
10726	J	113,18
10731	J	101,21
10741	K	113,19
10746	J	89,24
10756	J	101,21
10761	J	101,21
10771	J	89,24
10776	J	101,21
10786	J	101,21
10791	J	101,21
10801	J	89,24
10806	J	89,24
10831	I	103,21
10836	I	103,21
10851	J	101,21
10861	I	103,21
10866	I	103,21

v	C	Par
10876	J	101,21
11331	I	121,18
11356	I	121,18
11361	J	125,18
11386	I	121,18
11391	H	67,34
11416	J	107,21
11481	J	107,21
11721	I	127,18
11991	H	109,22
12046	J	113,21
12076	J	113,21
12081	I	79,30
12106	J	113,21
12111	J	113,21
12141	M	981,51,13
12201	J	113,21
12226	H	163,15
12256	J	113,21
12261	J	113,21
12286	J	113,21
12291	J	113,21
12316	J	113,21
12321	J	131,18
12336	K	137,18
12346	J	131,18
12351	J	113,21
12376	I	103,24
12381	J	113,21
12396	I	103,24
12406	J	137,18
12411	I	103,24
12421	I	103,24
12441	J	137,18
12501	J	137,18
12526	I	139,18

Table 4 (continued)

v	C	Par
12556	I	139,18
12561	J	83,30
12591	I	139,18
12651	M	576,1,22
12676	H	169,15
12766	M	936,91,15
13371	J	125,21
13396	J	125,21
13626	I	151,18
13651	I	151,18
13996	J	149,18
14001	J	131,21
14026	J	131,21
14031	I	151,18
14056	I	151,18
14061	J	131,21
14086	I	151,18
14091	J	149,18
14116	J	131,21
14151	I	157,18
14361	J	131,21
14386	J	131,21
14416	J	137,21
14421	I	157,18
14446	M	1126,16,13
14451	J	137,21
14476	H	193,15
14481	J	137,21

v	C	Par
14506	J	137,21
14536	I	121,24
14541	J	137,21
14566	J	137,21
14571	I	157,18
14871	I	139,21
15471	I	103,30
15496	M	1276,91,13
15531	J	125,24
15616	J	167,18
15646	J	173,18
15651	I	169,18
15676	I	169,18
15681	I	103,30
15706	I	169,18
15711	J	167,18
15736	J	149,21
15801	I	169,18
15916	J	149,21
16056	H	247,13
16116	I	151,21
16131	I	151,21
16141	I	151,21
17391	I	193,18
17406	I	193,18
17431	I	193,18
18826	J	179,21
18981	J	179,21

v	C	Par
19041	J	179,21
19281	I	211,18
19936	I	211,18
20056	J	83,48
20061	I	163,24
20086	I	223,18
20121	I	163,24
20301	J	191,21
20326	H	271,15
21921	J	233,18
22066	J	239,18
22096	J	239,18
22126	J	233,18
22131	I	241,18
22191	J	239,18
23301	I	127,36
23326	J	251,18
23476	H	313,15
26926	I	289,18
27106	I	289,18
27231	I	289,18
27406	I	256,21
27526	H	367,15
27676	I	307,18
28126	I	307,18
28131	J	263,21
28221	I	307,18
28426	H	379,15

If we remove some of the points of a transversal design, both from the design itself and from the groups and blocks that contain them, we obtain a special type of group divisible design known as a truncated transversal design. Regarding the groups of this design as additional blocks we obtain a PBD. We apply Theorem 6 to this type of PBD to obtain our next few constructions.

Construction G (The basic MOLS construction). Let m, t, and ρ be integers such that

(i) $0 \leq t < m$,

(ii) a TD$(\rho + 1, m)$ exists,

(iii) $(5w + 1, 6, 1)$–BIBDs exist for $w = t$, m, ρ, and $\rho + 1$.

Then a $(5m\rho + 5t + 1, 6, 1)$–BIBD exists.

Here m, t, and ρ are given, in that order, in the Par column of Table 4.

Construction G is obtained from the TD$(\rho + 1, m)$ by removing $m - t$ points from one of the groups and applying Theorem 6. □

We will see later that this construction is powerful enough to reduce the proof of our main theorem to the consideration of a finite number of cases.

Construction H. Let m and ρ be integers such that

(i) a TD(ρ, m) exists,

(ii) $(5w + 1, 6, 1)$–BIBDs exist for $w = m$ and ρ.

Then a $(5m\rho + 1, 6, 1)$–BIBD exists. Here m and ρ are given, in that order, in the Par column of Table 4.

Construction H is obtained from directly from the TD(ρ, m) be applying Theorem 6. It is a $t = 0$ variation of Construction G. □

Construction I. Let m, t, and ρ be integers such that

(i) a TD$(\rho + t, m)$ exists,

(ii) $(5w + 1, 6, 1)$–BIBDs exist for $w = m$, ρ, $\rho + 1$, and $\rho + t$.

Then a $(5m\rho + 5t + 1, 6, 1)$–BIBD exists. Here m and ρ are given, in that order, in the Par column of Table 4, and $t = (v - 5m\rho - 1)/5$.

We start with the TD$(\rho + t, m)$ and select one of the blocks, which is called the spike block. We select t of the groups and from each of these t groups we remove all the points not in the spike block. This gives us a $(m\rho + t, \{m, \rho, \rho + 1, \rho + t\}, 1)$–PBD, and we apply Theorem 6. □

Construction J. Let m, t, and ρ be integers such that

(i) $0 \leq t \leq m$,

(ii) a TD$(\rho + 1, m)$ exists,

(iii) $(5w + 1, 6, 1)$–BIBDs exist for $w = t + 1$, $m + 1$, ρ, and $\rho + 1$.

Then a $(5m\rho + 5t + 6, 6, 1)$–BIBD exists. Here m and ρ are given, in that order, in the Par column of Table 4, and $t = (v - 5m\rho - 6)/5$.

We start with the TD$(\rho+1, m)$, and remove $m-t$ point from one of the groups. To the resulting truncated transversal design we adjoin one extra point X, sometimes called the point at infinity, and add this extra point to each of the groups. We now use Theorem 6. □

Construction K. Let m and ρ be integers such that

(i) a TD(ρ, m) exists,

(ii) $(5w + 1, 6, 1)$–BIBDs exist for $w = m + 1$ and ρ.

Then a $(5m\rho + 6, 6, 1$–BIBD exists. Here m and ρ are given in the Par column of Table 4.

We start with the TD(ρ, m) and we adjoin the point at infinity X, and add this extra point to each of the groups. We now apply Theorem 6. This construction is a $t = 0$ variation of Construction J. □

Constructions J and K are analogs of Constructions G and H.

Construction L. Suppose there exist integer w and f such that there exists a TD$(6, w-f)$ and a $(w, \{6, f^*\}, 1)$–PBD. Then there exists $(6w-5f, \{6, w^*\}, 1)$–PBD. Furthermore if there is a $(w, 6, 1)$–BIBD, then there exists a $(6w - 5f, 6, 1)$–BIBD. Here the values of w and f are given, in that order, in the Par columns of Tables 3 and 4.

Here we start with our TD$(6, w - f)$ and adjoin f special elements. We let F denote the set of these special elements, and let $G_1, G_2, \ldots, G_6$ be the six groups of our TD. For $i < 6$ we construct a $(w, \{6, f^*\}, 1)$–PBD on the set $G_i \cup F$ in such a way that F is the hole. We take our blocks to be

(I): the blocks, other that F, of these 5 PBDs,

(II): the blocks of the TD, and

(III): the one large block $G_6 \cup F$.

We have now constructed a $(6w - 5f, \{6, w^*\}, 1)$–PBD. If there is a $(w, 6, 1)$–BIBD then we can replace our large block by the blocks of this $(w, 6, 1)$–BIBD to obtain a $(6w - 5f, 6, 1)$–BIBD. □

Each time we use Construction L, the PBDs and the smaller BIBDs that we need are to be found in Tables 3 and 4.

Construction M. Suppose there exist positive integers w, f, m and x such that

(i) $w = 5m + f$,

(ii) a $\mathrm{TD}(6, m)$ exists,

(iii) a $(w, \{6, f^*\}, 1)$–PBD exists, and

(iv) a $(5x + 1, 6, 1)$–BIBD exists.

Then a $(5mx + f, \{6, w^*\}, 1)$–PBD exists. Furthermore if a $(w, 6, 1)$–BIBD exists, then a $(5mx + f, 6, 1)$–BIBD and a $(5mx + f, \{6, (5x + 1)^*\}, 1)$–PBD exist. Here w, f, and x are given, in that order, in the Par columns of Tables 3 and 4, and $m = (w - f)/5$.

We start with a $(5x + 1, 6, 1)$–BIBD. We call this BIBD $\mathcal{A}$. Let α_0 be a fixed point of $\mathcal{A}$. We replace α_0 by an f element set F containing α_0 and $f - 1$ new points. For each $\alpha \in \mathcal{A}$, $\alpha \neq \alpha_0$, we replace α by an m-element set G_α containing α and $m - 1$ new points. Thus we now have a $(5mx + f)$-element set. We now consider the blocks of $\mathcal{A}$. For each block B that does not contain α_0 we take the m^2 blocks of the $\mathrm{TD}(6, m)$ which has groups G_β, $\beta \in B$. We can do this in such a way that B is one of the blocks of our TD. Let C_0 be a fixed block of $\mathcal{A}$ that contains α_0. We take the block of size w to consist of the points of F and the points of the five subsets G_β, $\beta \in C_0$. For each block C of $\mathcal{A}$, $C \neq C_0$, that does contains α_0, we construct a $(w, \{6, f^*\}, 1)$–PBD on the set consisting of the points of F and the points of the 5 subsets G_γ, $\gamma \in C$, $\gamma \neq \alpha_0$, in such a way that F is the block of size f and C is one of the blocks of size 6. We now have constructed a $(5mx + f, \{6, w^*\}, 1)$–PBD.

Now suppose that a $(w, 6, 1)$–BIBD exists. We replace the block of size w by a $(w, 6, 1)$–BIBD on the same points to obtain a $(5mx + f, 6, 1)$–BIBD. We can do this in such a way that C_0 is one of the blocks of this $(w, 6, 1)$–BIBD. Thus our new design contains all the points and blocks of $\mathcal{A}$. We remove all these blocks and replace them with a single block of size $5x + 1$, to obtain our $(5mx + f, \{6, (5x + 1)^*\}, 1)$–PBD. □

Each time we use Construction M the other PBDs and BIBDs that we need are available in Tables 3 and 4.

Construction N. Let v be a prime power such that $v \equiv 1 \pmod{30}$. We write $v = 30t + 1$. Set $S = \mathrm{GF}(v)$, let g be a primitive element of S, let ω be a primitive cube root of unity in this field, and let H be the multiplicative subgroup of S generated by g^5. Wilson [28] observed that if there exists an element c in S such that

$$\omega - 1, c(\omega - 1), c - 1, c - \omega, c - \omega^2$$

is a system of representatives for the cosets modulo H, then a $(v, 6, 1)$–BIBD can be constructed. To do this we take $G = G^+$, and we let our base blocks be

$$g^{5a} \quad g^{5a}\omega \quad g^{5a}\omega^2 \quad g^{5a}c \quad g^{5a}c\omega \quad g^{5a}c\omega^2, \qquad 0 \le a < t.$$

Thus we obtain t base blocks, each of which generates v blocks, giving us a total vt blocks, which is correct. In this case a suitable value of c is given in the Par column of Table 4. When $v = 361$ we have $c = 3 + i$. Here i is a square root of -1 in GF(361).

There is no such c for $v = 61$ or 121, but probably such a c exists for all prime powers v such that $v \equiv 1 \pmod{30}$, $v > 121$. At any rate a c exists for all such v for which we need it. Suitable values of c for some other prime power values of v can be found in [28] and [20]. □

Construction O. Suppose that there exist positive integers w, f, m, μ, ρ, and t such that

(i) $w = f + 5\rho$,

(ii) there exists a $(w, \{6, f^*\}, 1)$–PBD,

(iii) there exists a $\mathrm{TD}(\rho + 1, m)$,

(iv) $0 \le t \le m$,

(v) $\mu = 1$ or 6,

(vi) there exist $(5\rho + 1, 6, 1)$ and $(5m + \mu, 6, 1)$–BIBDs.

Set $v = (f-1)t + 5m\rho + \mu$. Then there exists a $(v, \{6, (ft - t + \mu)^*\}, 1)$–PBD. Furthermore if there is an $(ft - t + \mu, 6, 1)$–BIBD, then there exist a $(v, 6, 1)$–BIBD and a $(v, \{6, (5m + \mu)^*\}, 1)$–PBD. Here the values of w, f, m, and μ are given, in that order, in the Par columns of Tables 3 and 4, Since v is given the values of ρ and t can be calculated.

If we delete one point from the $(5\rho + 1, 6, 1)$–BIBD and from the blocks that contain it, and then take the resulting blocks of size 5 as groups, we obtain a GDD$[5^\rho; 6]$. Similarly if we remove one point from the block of size f from the $(w, \{6, f^*\}, 1)$–PBD, we obtain a GDD$[5^\rho(f-1)^1; 6]$.

We now take a TD$(\rho+1, m)$ and we remove $m - t$ points from one of its groups and we remove these points from the blocks that contained them. We call the resulting truncated transversal design $\mathcal{B}$. It has a group G_0 with t points, and groups $G_1, \ldots, G_\rho$ with m points each. We now inflate each point of G_0 by a factor of $f-1$, and every other point of $\mathcal{B}$ by a factor of 5. Let $I(\alpha)$ be the set of order $f-1$ or 5 that we get when we inflate the point α. We now adjoin a set H of μ elements to our design. We now have a v-element set. We must now describe the blocks. For each block B of $\mathcal{B}$ that does not contain an element of G_0 we construct a GDD$[5^\rho; 6]$ whose groups are the sets $I(\alpha)$, $\alpha \in B$. For each block C of $\mathcal{B}$ that does contain a point of G_0 we construct a GDD$[5^\rho(f-1)^1; 6]$ whose groups are the sets $I(\alpha)$, $\alpha \in C$. For each i, $0 \le i \le \rho$, let G_i' denote the set of the points of H and of all the $I(\alpha)$ with $\alpha \in G_i$. For each i, $1 \le i \le \rho$, we construct a $(5m+\mu, 6, 1)$–BIBD on the points of G_i'. If $\mu = 6$ we do this in such a way that H is one of the blocks, and we delete this block. We now include the large block G_0' and we have our $(v, \{6, (ft-t+\mu)^*\}, 1)$–PBD. Furthermore if there is an $(ft-t+\mu, 6, 1)$–BIBD, then we can replace the large block by the blocks of this BIBD on its points. This gives us the $(v, 6, 1)$–BIBD. Finally in this $(v, 6, 1)$–BIBD we can replace the blocks of one of the $(5m+\mu, 6, 1)$–BIBDs by a single large block to obtain a $(v, \{6, (5m + \mu)^*\}, 1)$–PBD. □

Construction P. Suppose that there exist positive integers m, t, and μ such that

(i) there exists a TD$(7, m)$,

(ii) $0 \le t \le m$,

(iii) $\mu = 1$ or 6,

(iv) there exist $(20m + \mu, 6, 1)$ and $(25t + \mu, 6, 1)$–BIBDs.

Then there exists a $(120m + 25t + \mu, 6, 1)$–BIBD. In this case m, t, and μ are listed, in that order, in the Par column of Table 4.

Here the proof is similar to the proof for Construction O. To make this work we need a GDD$[20^6; 6]$ and a GDD$[20^6 25^1; 6]$. The first of these GDDs is simply a TD(6,20) which we already know exists. A construction of the second of these GDDs was found by Wilson and it can be found, in a slightly disguised form, in Beth, Jungnickel, and Lenz [4, Chapter 8, Proposition 9.15]. Now we take our TD$(7, m)$ and delete $m - t$ points from one of the groups, say G_0. We now inflate the remaining points of G_0 by a factor of 25 and all other points of this truncated transversal design by a factor of 20. Now we proceed as we did with Construction O. □

Construction Q. Suppose that w, f, and δ are positive integers, $\delta \leq f \leq w$, such that a $(w, \{6, f^*\}, 1)$–PBD and an incomplete transversal design TD$(6, w - \delta)$ − TD$(6, f - \delta)$ both exist. Set $v = 6w - 5\delta$. Then a $(v, \{6, (6f - 5\delta)^*\}, 1)$–PBD exists. Furthermore if a $(6f - 5\delta, 6, 1)$–BIBD exists, then a $(v, 6, 1)$–BIBD exists. In this case w, f, and δ are listed, in that order, in the Par columns of Tables 3 and 4.

We start with the incomplete transversal design TD$(6, w - \delta)$ − TD$(6, f - \delta)$, and adjoin the δ points of a δ-element set D. Now we have a hole Y such that $Y \cap G$ has $f - \delta$ points for any group G of the incomplete transversal design. For each group G we consider the w-element set $G \cup D$. We construct a $(w, \{6, f^*\}, 1)$–PBD on $G \cup D$ such that the block of size f is $(G \cap Y) \cup D$. We remove the block of size f. We now add the large block $Y \cup D$ to the remaining blocks of the PBD and the blocks of the incomplete transversal design to obtain a $(v, \{6, (6f - 5\delta)^*\}, 1)$–PBD. Finally if a $(6f - 5\delta, 6, 1)$–BIBD exists, we replace our large block by the blocks of this design to obtain a $(v, 6, 1)$–BIBD. □

Construction Q is a generalization of Construction L.

6 PROOFS

We wish to prove that all the PBDs of Table 3 exist as well as $(v, 6, 1)$–BIBDs for all admissible v except the 55 values given in Table 1.

We begin by showing that the basic MOLS construction, Construction G, is powerful enough to settle all sufficiently large v in an induction proof. To use this construction we need an integer ρ such that $(5\rho+1, 6, 1)$ and $(5\rho+6, 6, 1)$–BIBDs exist. At present the smallest ρ known to have this property is 18, so we will use $\rho = 18$.

Next we need to find values of m such that a TD$(19, m)$ and a $(5m + 1, 6, 1)$–BIBD both exist. We consider the set $\mathcal{C}$ and all integers of the form $6s+1$. Any set of seven consecutive integers of $\mathcal{C}$ contains at most two integers divisible by 5, and at most one divisible by each of 7, 11, 13, or 17. Hence any set of 42 consecutive integers contains an integer m in $\mathcal{C}$ such that all prime power factors of m are greater than 18, so that $5m + 1$ is admissible and a TD$(19, m)$ exists by Theorem 2. Such values of m are close enough together to that we can prove the following result.

Theorem 7. *Let b be a integer greater than 7 such that there exists a $(v, 6, 1)$–BIBD for all admissible v in the range $5b + 1 \leq v < 95b + 71821$. Then a $(v, 6, 1)$–BIBD exists for all admissible $v \geq 5b + 1$.*

Proof. We use induction on v. Suppose that $v \geq 95b+71821$ and set $v = 5w+1$. Then $w \geq 19b + 14364$. By our above remarks there is an integer m such that $w/19 < m \leq 42 + w/19$, $5m + 1$ is admissible, and a TD(19,m) exists. Since $w - 19b \geq 14364 = 18 \cdot 19 \cdot 42$ we have $42 + w/19 \leq (w - b)/18$. Hence $18m + b \leq w < 19m$, so that we can put $w = 18m + t$, where $b \leq t < m$. Now $5t + 1 \equiv 5w + 1 = v \pmod{15}$ so that $5t + 1$ is admissible. Therefore a $(5t + 1, 6, 1)$–BIBD and a $(5m + 1, 6, 1)$–BIBD exist. We use Construction G, with $\rho = 18$, to get a $(v, 6, 1)$–BIBD, and our induction proof is complete. □

Proof of Theorem 1. Setting $b = 407$ in Theorem 7, we see that it is sufficient to prove our results for $v < 110486$. This still leaves over 14,000 values of v to be treated. Most of these can be done using Construction G, the basic MOLS construction. In Table 4 we give constructions for all admissible values of $v < 2036$ that are not listed in Table 1. We also give construction for all admissible values v in the range $2036 \leq v < 110486$ that cannot be done with Construction G. For many of these we need $(w, 6, 1)$–BIBDs and $(w, \{6, f^*\}, 1)$–PBDs for values of $w \leq v$. The ones we need can all be found in Tables 3 and

4. We proceed by induction on v, so that all necessary ingredients for our constructions are present. □

We have noted that to use Construction G we must have both $(5\rho+1, 6, 1)$ and $(5\rho + 6, 6, 1)$–BIBDs, so that there are only a few values of ρ that we can use at present. Indeed the only values of ρ used for the BIBDs missing from Table 4 are $\rho = 18$, 21, 24, 30, 36, 42, 48, 60, and 66.

7 SHORT LISTS

Many of the 55 open cases could be done if we has the smaller BIBDs on our list. In Table 5 we give a list of 19 values of v such that if we had $v, 6, 1)$–BIBDs for these values, then all the other 36 unknown BIBDs on our list could be constructed.

Table 5
The construction of $(v, 6, 1)$–BIBDs for these 19 values of v would complete the solution of the existence problem for $k = 6$, $\lambda = 1$.

46	51	61	81	141	166	196	201	226	261
286	291	321	346	351	376	466	471	496	

The case $v = 46$ has resisted many attempts at construction. If this $(46, 6, 1)$–BIBD does not exist, then the 29 values of v from Table 6 would complete the job.

Table 6
If a $(46, 6, 1)$–BIBD does not exist the construction of $(v, 6, 1)$–BIBDs for these 29 values of v would complete the solution of the existence problem for $k = 6$, $\lambda = 1$.

51	61	81	141	166	196	201	226	231	261
286	291	316	321	346	351	376	411	436	466
471	496	501	526	616	646	741	771	946	

If neither the $(46, 6, 1)$–BIBD nor the $(51, 6, 1)$–BIBD exist, then the 35 values of v from Table 7 would complete the job.

Table 7
If the $(46, 6, 1)$ and the $(51, 6, 1)$–BIBDs do not exist
the construction of $(v, 6, 1)$–BIBDs for these 35
values of v would complete the solution of
the existence problem for $k = 6$, $\lambda = 1$.

61	81	141	166	196	201	226	231	256	261	276	286
291	316	321	346	351	376	411	436	441	466	471	496
501	526	561	616	646	651	676	741	771	801	946	

REFERENCES

[1] R. J. R. Abel, Four mutually orthogonal Latin squares of orders 28 and 52, *J. Combinatorial Theory* 58A (1991), 306–309.

[2] R. J. R. Abel and W. H. Mills, Some new BIBDS with $k = 6$ and $\lambda = 1$, *J. Combin. Des.* 3 (1995), 381–392.

[3] R. J. R. Abel and D. T. Todorov, Four MOLS of orders 20, 30, 38, and 44, *J. Combinatorial Theory* 64A (1993), 144–148.

[4] T. Beth, D. Jungnickel, and H. Lenz, *Design Theory*, Bibliographisches Institut Mannheim, Wien, Zürich, 1985.

[5] R. C. Bose, On the application of finite projective geometry for deriving a certain series of balanced Kirkman arrangements, *Calcutta Math. Soc. Golden Jubilee* (1959), 341–354.

[6] A. E. Brouwer and G. H. J. van Rees, More mutually orthogonal Latin squares, *Discrete Math.* 39 (1982), 263–281.

[7] R. H. Bruck and H. J. Ryser, The nonexistence of certain finite projective planes, *Canad. J. Math.* 1 (1949), 88–93.

[8] Charles J. Colbourn, Four MOLS of order 26, *J. Combin. Math. Combin. Comput.* 17 (1995), 147–148.

[9] R. H. F. Denniston, A Steiner system with a maximal arc, *Ars Combinatoria* 9 (1980), 247–248.

[10] Malcolm Greig, unpublished.

[11] Marshall Hall, Jr., *Combinatorial Theory*, John Wiley & Sons, New York (1986).

[12] Haim Hanani, Balanced incomplete block designs and related designs, *Discrete Math.* 11 (1975), 255–369.

[13] Diane M. Johnson, A. L. Dulmage, and N. S. Mendelsohn, Orthomorphisms of groups and orthogonal Latin squares I, *Canad. J. Math.* 13 (1961), 356–372.

[14] H. F. MacNeish, Euler squares, *Ann. Math.* 23 (1922), 221–227.

[15] R. D. McKay and R. G. Stanton, Isomorphism of two large designs, *Ars Combinatoria* 6 (1978), 87–90.

[16] Y. Miao and Lie Zhu, On resolvable BIBDs with block size five *Ars Combin.* 24 (1995), 261–275.

[17] W. H. Mills, Two new block designs, *Utilitas Math.* 7 (1975), 73–75.

[18] _____ , A new block design, *Congressus Num.* 14 (1975), 461–465.

[19] _____ , Some mutually orthogonal Latin squares, *Congressus Num.* 19 (1977), 473–487.

[20] _____ , The construction of balanced incomplete block designs with $\lambda = 1$, *Congressus Num.* 20 (1977), 131–148.

[21] _____ , The construction of BIBDs using nonabelian groups, *Congressus Num.* 21 (1978), 519–526.

[22] _____ , The construction of balanced incomplete block designs, *Congressus Num.* 23 (1979), 73–86.

[23] _____ , Balanced incomplete block designs with $k = 6$ and $\lambda = 1$, *Enumeration and Design*, Academic Press, Toronto, Ont. (1984), 239–244.

[24] R. C. Mullin, Finite bases for some PBD-closed sets, *Discrete Math.* 77 (1989), 217–236.

[25] R. C. Mullin, D. G. Hoffman, and C. C. Lindner, A few more BIBD's with $k = 6$ and $\lambda = 1$, *Combinatorial Design Theory*, North-Holland Math. Stud. 149, North Holland, Amsterdam-New York, (1987), 379–384.

[26] Charles E. Roberts, Jr., Sets of mutually orthogonal Latin squares with 'like subsquares', *J. Combin. Theory* 61A (1992), 50–63.

[27] R. G. Stanton and D. G. Gryte, A family of BIBDs, *Combinatorial Structures and their Applications*, Gordon and Breach, New York (1970), 411–412.

[28] R. M. Wilson, Cyclotomy and difference families in elementary abelian groups, *J. Number Theory* 4 (1972), 17–47.

[29] _____, An existence theory for pairwise balanced designs, I: Composition theorems and morphisms, *J. Combin. Theory* 13A (1972), 220–245.

[30] _____, Concerning the number of mutually orthogonal Latin squares, *Discrete Math.* 9 (1974), 181–198.

[31] _____, An existence theory for pairwise balanced designs, III: Proof of the existence conjectures, *J. Combin. Theory* 18A (1975), 71–79.

[32] Lie Zhu, Beiliang Du, and Yin Jianxing, Some new balanced incomplete block designs with $k = 6$ and $\lambda = 1$, *Ars Combin.* 24 (1987), 167–174.

[33] Lie Zhu, Beiliang Du, and Xuebin Zhang, A few more RBIBDs with $k = 5$, $\lambda = 1$, *Discrete Math.* 97 (1991), 409–417.

9

POLYHEDRAL METHODS IN DESIGN THEORY

Lucia Moura

Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
Canada
lucia@cs.toronto.edu

ABSTRACT

This chapter is devoted to the relation between polyhedral theory and combinatorial designs. The polyhedral aspects of constructing packings, coverings and t-designs are emphasized. Classical results and algorithms in polyhedral theory are summarized, integer programming formulation of design construction problems are presented, and polyhedra associated to these formulations and related algorithms are discussed.

1 INTRODUCTION

This chapter is devoted to the relation between polyhedral theory and combinatorial designs. Our focus is on packings, coverings and t-designs. We address the problems of existence and construction of designs, and their polyhedral aspects. Polyhedra related to coding theory were studied by Zehendner[27].

Let λ, v, k and t be positive integers with $v \geq k \geq t$. A t-(v,k,λ) *design* is a pair $(V,\mathcal{B})$ where V is a v-set and $\mathcal{B}$ is a collection of k-subsets of V called blocks such that every t-subset of V is contained in exactly λ blocks of $\mathcal{B}$. A t-(v,k,λ) *packing design* and a t-(v,k,λ) *covering design* are defined by replacing the condition "in exactly λ blocks" in the above definition by "in at most λ blocks" and "in at least λ blocks", respectively. The *packing number* denoted by $D_\lambda(v,k,t)$ is the *maximum* number of blocks in a t-(v,k,λ) packing design. The *covering number* denoted by $C_\lambda(v,k,t)$ is the *minimum* number of blocks in a t-(v,k,λ) covering design. When a t-(v,k,λ) design exists, it is both a

W. D. Wallis (ed.), Computational and Constructive Design Theory, 227–254.
© 1996 *Kluwer Academic Publishers.*

maximum packing and a minimum covering design. A survey on packings and coverings is given by Mills and Mullin [15].

For clarity of exposition, we restrict ourselves to *simple* designs, although similar results are valid for designs with repeated blocks. We denote the set of all k-subsets of a set V by $\binom{V}{k}$. Let $\mathcal{D} = (V, \mathcal{B})$ be a design with parameters t, v, k, λ. The *incidence vector* $x \in \mathbb{R}^{\binom{v}{k}}$ associated to $\mathcal{D}$ and indexed by each $S \in \binom{V}{k}$ is defined by

$$x_S = \begin{cases} 1 & \text{if } S \in \mathcal{B}, \\ 0 & \text{otherwise.} \end{cases}$$

First, we establish some notation. Let $x, y \in \mathbb{R}^n$. We say that $x \leq y$ if $x_i \leq y_i$, $i = 1, \ldots, n$; we say that x is integer if x_i is integer, $i = 1, \ldots, n$. We denote the vector $(0, \ldots, 0)^T \in \mathbb{R}^n$ by $\mathbf{0}$, and the vector $(1, \ldots, 1)^T \in \mathbb{R}^n$ by $\mathbf{1}$.

Before defining the polyhedra associated with designs, we review the definition of convex hull. Let $x^1, \ldots, x^k \in \mathbb{R}^n$ and $\alpha_1, \ldots, \alpha_k$ be nonnegative real numbers with $\sum_{i=1}^k \alpha_i = 1$; then $x = \sum_{i=1}^k \alpha_i x^i$ is called a *convex combination* of $x^1, \ldots, x^k \in \mathbb{R}^n$. Given a nonempty subset $E \subseteq \mathbb{R}^n$, the *convex hull* of elements in E is the set of all convex combinations of elements of E, and is denoted by *conv(E)*.

We define the polyhedra associated to t-designs, packings and coverings, respectively, by

$$\begin{aligned} T_{t,v,k,\lambda} &= conv\{x : x \text{ is an incidence vector of a } t\text{-}(v,k,\lambda) \text{ design}\}, \\ P_{t,v,k,\lambda} &= conv\{x : x \text{ is an incidence vector of a } t\text{-}(v,k,\lambda) \text{ packing}\}, \\ C_{t,v,k,\lambda} &= conv\{x : x \text{ is an incidence vector of a } t\text{-}(v,k,\lambda) \text{ covering}\}. \end{aligned}$$

A t-(v, k, λ) design exists if and only if the polyhedron $T_{t,v,k,\lambda}$ is non empty. Moreover, the incidence vectors of the t-(v, k, λ) designs are precisely the vertices of this polyhedron. We can easily see that the packing number $D_\lambda(v, k, t) = \max\{\mathbf{1}^T x : x \in P_{t,v,k,\lambda}\}$ and the corresponding maximum solutions x that are vertices of $P_{t,v,k,\lambda}$ correspond to the maximum packings. Analogously, the covering number $C_\lambda(v, k, t) = \min\{\mathbf{1}^T x : x \in C_{t,v,k,\lambda}\}$ and the corresponding minimum solutions x that are vertices of $C_{t,v,k,\lambda}$ correspond to the minimum coverings. Since any t-design is both a packing and a covering, we observe that $P_{t,v,k,\lambda} \cap C_{t,v,k,\lambda} = T_{t,v,k,\lambda}$.

After this brief presentation of the polyhedra associated with designs, we outline the rest of the paper. We give basic results in polyhedral theory and linear

programming in Section 2. We discuss the integer programming problems associated to designs in Section 3. We survey results on the facial structure of some related problems, namely set partitioning, set packing and set covering problems, in Section 4. We outline a polyhedral based algorithm for deciding existence of t-designs and constructing designs, and give some results on the facial structure of the polyhedra of the design problems in Section 5.

2 POLYHEDRAL THEORY AND ALGORITHMS

In polyhedral combinatorics, polyhedra are usually given implicitly, and our objective is to find their explicit representations. In the situation we are interested in, a polyhedron is implicitly defined by having designs as its vertices. The goal is to determine linear inequalities describing this polyhedron, so that we can compute its vertices, i.e., find the corresponding designs.

In this section, we give some basic definitions and results concerning polyhedra, discuss polyhedral theory in the context of combinatorial optimization, and describe related algorithms. Complete treatments of polyhedral theory can be found in the survey papers by Bachem and Grötschel [2] and Pulleyblank [20], and in the book by Nemhauser and Wolsey [17].

2.1 Basic polyhedral theory and linear programming

The proofs for the results contained in this subsection can be found in [17]. Throughout this subsection, let n and m be positive integers. A *polyhedron* $P \subseteq \mathbb{R}^n$ is the set of points satisfying a finite set of linear inequalities; that is, for any matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^n$, $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is a polyhedron. A *polytope* is a bounded polyhedron, i.e., a polyhedron $P \subseteq \mathbb{R}^n$ is a polytope if and only if there exist vectors $u, v \in \mathbb{R}^n$ such that $l \leq x \leq u$ for all $x \in P$.

A *linear programming problem* is one of maximizing (minimizing) a linear function over a polyhedron. Let $c \in \mathbb{R}^n$ be the profit (cost) vector and $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be the associated polyhedron. We can write the

linear programming problem as

$$\text{(LP)} \quad \begin{cases} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b. \end{cases}$$

Minimization problems, as well as problems with different inequality signs $(=,\geq)$, can easily be brought to the form given above by simple algebraic manipulations.The study of polyhedra is fundamental for linear programming.

First, we review some basic definitions from linear algebra; then, we define the concepts of dimension, faces, and facets of a polyhedron; finally, we state classical results on minimal representation of polyhedra.

A point x is called a *linear combination* of $x^1, \ldots, x^k \in \mathbb{R}^n$ if there exist $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{R}^k$ such that $x = \sum_{i=1}^k \alpha_i x^i$. If $\alpha \geq \mathbf{0}$ then x is a *conic combination.* If $\sum_{i=1}^k \alpha_i = 1$ then x is an *affine combination.* If $\alpha \geq \mathbf{0}$ and $\sum_{i=1}^k \alpha_i = 1$ then x is a *convex combination.*

Given a nonempty subset $S \subseteq \mathbb{R}^n$, the set of all linear (conic, affine, convex) combinations of points in S is called the *linear (conic, affine, convex) hull* of elements in S and are denoted by *lin(S), cone(S), aff(S), conv(S)*, respectively. We say that S is a *cone* if $S = cone(S)$ and that it is a *convex set* if $S = conv(S)$. It is easy to prove that any polyhedron is a convex set, and that a polyhedron of the form $\{x \in \mathbb{R}^n : Ax \leq \mathbf{0}\}$ is a cone.

A set of points $x^1, \ldots, x^k \in \mathbb{R}^n$ is *linearly independent* if the unique solution $\alpha \in \mathbb{R}^k$ to $\sum_{i=1}^k \alpha_i x^i = 0$ is $\alpha_i = 0$ for $i = 1, \ldots, k$. Analogously, a set of points $x^1, \ldots, x^k \in \mathbb{R}^n$ is *affinely independent* if the unique solution $\alpha \in \mathbb{R}^k$ to $\sum_{i=1}^k \alpha_i x^i = 0$, $\sum_{i=1}^k \alpha_i = 0$ is $\alpha_i = 0$ for $i = 1, \ldots, k$. The following proposition relates both concepts.

Proposition 1 *The following statements are equivalent:*

i. The set of points $x^1, \ldots, x^k \in \mathbb{R}^n$ is affinely independent.

ii. The set of points $x^2 - x^1, \ldots, x^k - x^1 \in \mathbb{R}^n$ is linearly independent.

iii. The set of points $(x^1, -1), \ldots, (x^k, -1) \in \mathbb{R}^{n+1}$ is linearly independent.

Since affine independence of points in a polyhedron is invariant under translation, affine independence is a more suitable concept to deal with polyhedra than linear independence.

Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. It is convenient to make some distinction among inequalities that define a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Let $I = \{1, \ldots, m\}$, $I^= = \{i \in I : a^i x = b_i \text{ for all } x \in P\}$ and let $I^\leq = \{i \in I : a^i x < b_i \text{ for some } x \in P\} = I \setminus I^=$. We indicate by $(A^=, b^=)$ and $(A^\leq, b^\leq)$ the rows of (A, b) corresponding to $I^=$ and $I^\leq$, respectively, the equality and inequality representations of P with respect to (A, b).

A polyhedron $P \subseteq \mathbb{R}^n$ is of *dimension* k, denoted by $dim P = k$, if the maximum number of affinely independent points in P is $k + 1$. We say that P is *full dimensional* if $dim P = n$. The maximum number of linear independent rows of a matrix $A \in \mathbb{R}^{m \times n}$ is the *rank* of A, denoted by $rank(A)$.

Proposition 2 *If $P \subseteq \mathbb{R}^n$ then $dim P + rank(A^=, b^=) = n$.*

Let $d \in \mathbb{R}^n$ and $d_0 \in \mathbb{R}$. An inequality $d^T x \leq d_0$ is said to be *valid* for P if it is satisfied by all points of P. A subset $F \subseteq P$ is called a *face* of P if there exists a valid inequality $d^T x \leq d_0$ such that $F = P \cap \{x \in \mathbb{R}^n : d^T x = d_0\}$. The inequality is said to *represent* or to *induce* the face F. It is easy to see that P and $\emptyset$ are faces of P, and that a face of a polyhedron is itself a polyhedron. A face F of P is called *proper* if $F \neq P$ and *nontrivial* if $F \neq \emptyset$. A maximal nontrivial proper face F of P is called a *facet* of P. A face of dimension zero is called a *vertex* and a face of dimension one is called an *edge* of P. A polyhedron that has at least one vertex is said to be *pointed*. Let $P^0 = \{x \in \mathbb{R}^n : Ax \leq \mathbf{0}\}$ be the polyhedral cone associated to $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Any point $r \in P^0$ is called a *ray* of P and if r belongs to an edge of P^0 it is called an *extreme ray* of P.

The study of faces is important for linear programming. If a linear programming problem, $\max c^T x$ subject to $x \in P$, has an optimal solution, it is easy to see that the set of all optimal solutions is a face of the polyhedron P. Just note that if $c_0 = \max\{c^T x : x \in P\}$, then $c^T x \leq c_0$ is a valid inequality for P that induces the face $F = \{x \in P : c^T x = c_0\}$, which is the set of optimal solutions to the problem.

The main result concerning minimal inequality representation is summarized below.

Theorem 2.1 *Let P be a polyhedron.*

i. If P is full dimensional then it has a unique (up to scalar multiplication) minimal representation by a finite set of linear inequalities, each one representing each of the facets of P.

ii. If $dim P = n - k$ with $k > 0$ then a minimal representation of P is given by $\{x \in \mathbb{R}^n : a^i x = b_i,\ i = 1, \ldots, k,\ a^i x \leq b_i,\ i = k+1, \ldots k+t\}$, where $\{(a^i, b_i)\}_{i=1,\ldots,k}$ is a maximal set of linearly independent rows of $(A^=, b^=)$, and for $i = k+1, \ldots k+t$, $a^i x \leq b_i$ are inequalities representing each one of the facets F_i of P.

We now give two characterizations of facets that are very useful in deciding whether or not an inequality is facet inducing.

Theorem 2.2 *Let F be a nonempty proper face of $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and $(A^=, b^=)$ be the equality representation of P. Then, the following statements are equivalent:*

i. F is a facet of P.

ii. $dim F = dim P - 1$.

iii. For any pair of inequalities $d^T x \leq d_o$ and $\bar{d}^T x \leq \bar{d}_o$ that are valid for P and represent F, there exist $\alpha \in \mathbb{R}^m$ and $\gamma > 0 \in \mathbb{R}$ such that

$$d = \gamma \bar{d} + \alpha^T A^=, \quad d_o = \gamma \bar{d}_o + \alpha^T b^=.$$

By now we have seen one of the representations of polyhedra, namely the one by a minimum number of inequalities. Another way to represent a polyhedron is by means of its vertices and extreme rays, as stated in the following theorem.

Theorem 2.3 *(Minkoviski's theorem) Let P be a pointed polyhedron, and P_0 be its associated polyhedral cone. Then*

i. P has a finite set of vertices V,

ii. there is a finite set of extreme rays R of P such that $cone(R) = P_0$, and

iii. P= conv(V)+cone(R).

Remark: Most of the cases of polyhedra we will deal with are polytopes, i.e. bounded polyhedra. If P is a polytope then $P = conv(V)$, since for polytopes $cone(R) = \{\mathbf{0}\}$.

This theorem is the basis for the simplex algorithm for linear programming. The simplex method also uses the fact that if there is an optimal solution for a linear programming problem over a pointed polyhedron P, then there is an optimal solution that is a vertex, in which case the algorithm returns an optimal vertex.

2.2 Combinatorial optimization and polyhedral methods

We describe a general combinatorial optimization problem. Let $E = \{e_1, \ldots, e_n\}$ be a ground set, c_i be a weight associated with $e_i \in E$ for $i = 1, \ldots, n$, and $\mathcal{F}$ be a family of feasible subsets. We want to find $S \in \mathcal{F}$ such that $c(S) = \sum_{i:e_i \in S} c_i$ is maximized. A natural representation of an element $S \in \mathcal{F}$ is by a $0-1$ incidence vector $x^S \in \mathbb{R}^n$ defined by

$$x_i^S = \begin{cases} 1 & \text{if } e_i \in S, \\ 0 & \text{otherwise.} \end{cases}$$

This gives a polytope representation of the problem, namely the polytope $C = conv\{x^S : S \in \mathcal{F}\}$. This polytope has the nice property of its vertices being members of $\mathcal{F}$, as we have seen for the particular case of designs. Moreover, solving the combinatorial problem is equivalent to finding an optimal vertex of this polytope, which can be done via linear programming methods. The difficulty lies in the fact that we need a description of this polytope in terms of linear inequalities.

Sometimes the combinatorial optimization problem is given in the form of an integer programming problem. An integer programming problem is one of the form

$$\text{(IP)} \quad \begin{cases} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \text{ integer.} \end{cases}$$

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. The polyhedral representation of this problem is given by $P_I = conv\{x \in P : x \text{ is integer}\}$. We would like to obtain a complete representation of P_I by means of facet inducing inequalities. A useful technique for finding such inequalities is given by the *Chvátal-Gomory* cut generation. Let $\alpha \geq \mathbf{0} \in \mathbb{R}^n$. Then $(\alpha^T A)x \leq \alpha^T b$ is a valid inequality for P_I implied by $Ax \leq b$. If $\alpha^T A$ is integral then any integer solution x to the system will make $(\alpha^T A)x$ integral, so the inequality $(\alpha^T A)x \leq \lfloor \alpha^T b \rfloor$ is valid for P_I. Such an inequality is called a *Chvátal-Gomory cut* derived from $Ax \leq b$.

The following theorem was proven by Chvátal [5] for polytopes, and generalized by Schrijver [23] for polyhedra.

Theorem 2.4 *Let P and P_I be the polyhedra given by an integer programming problem as defined previously. Let $P^0 \supseteq \cdots \supseteq P^i \supseteq P^{i+1} \supseteq \cdots \supseteq P_I$ be a sequence of polyhedra recursively defined by: $P^0 = P$, P^{i+1} is the polyhedron obtained by the inequalities defining P^i plus all the Chvátal-Gomory cuts derived from these inequalities, for $i \geq 0$. Then, there is an integer t such that $P^t = P_I$, i.e., the sequence is finite.*

The main consequence of this theorem is that any facet inducing inequality can be obtained by a finite number of applications of the Chvátal-Gomory procedure. Chvátal-Gomory cuts have been used to derive and prove the validity of inequalities for polyhedra of structured problems. They can also be seen as a technique for generating valid and facet inducing inequalities.

The branch-and-cut algorithm

Now we describe an algorithm to solve general integer programming problems when some valid inequalities for their polyhedra are known. This algorithm combines well known techniques in integer programming, namely cutting-planes and branch-and-bound, and it was called *branch-and-cut* by Padberg.

Let us first describe the cutting-plane algorithm. Suppose we have a polyhedral description of a combinatorial optimization problem by means of a large number of facet inducing inequalities, so that it would be inefficient to solve the linear programming problem using all inequalities. Cutting-plane algorithms add inequalities one by one and often obtain an optimal solution before generating all of them. A cutting-plane algorithm is outlined below.

Cutting-Plane Algorithm
For any polyhedron $Q \supseteq P_I$, let $LP(Q)$ be the problem of finding $max\{c^T x : x \in Q\}$ and $x_{opt}(Q)$ be any optimal vertex solution of $LP(Q)$.
Input
P_R: a polytope containing P_I, the polyhedron for the problem.
Output
(success,x_{opt}): $x_{opt} = x_{opt}(P_I)$, or
(failure, x_T, P_T): P_T is s.t. $P_R \supseteq P_T \supseteq P_I$,

$x_T = x_{opt}(P_T)$.

$P^0 = P_R$; $k = 0$;
repeat
    Compute $x^k = x_{opt}(P^k)$;
    If $x^k \in P_I$ then "optimum was found";
    else
        Solve the separation problem:
            Find $d \in \mathbb{R}^n, d_o \in R$ such that $d^T x \leq d_o$ is valid for P_I
            and $d^T x^k > d_o$;
        If "separation was successful" then
            $P^{k+1} = P^k \cap \{x \in \mathbb{R}^n : d^T x \leq d_o\}$;
            $k = k + 1$;
until "optimum was found" or "separation was not successful"
If "optimum was found"
    then **return**(success, x^k);
    else **return**(failure,x^k, P^k).

Remark: If an integer programming formulation is used with $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and $P_I = conv\{x \in P : x \text{ is integer}\}$ then P_R may be set as P and the condition "$x^k \in P_I$" become equivalent to "x^k is integer". Moreover, if each inequality $d^T x \leq d_0$ induces a facet of P_I then the algorithm halts in a finite number of steps.

Cutting-plane algorithms alone may not be enough to efficiently solve a combinatorial optimization problem. It may happen that a complete description of the polyhedron is not available, or that the separation problem cannot be efficiently solved for a class of inequalities. So, the cutting-plane algorithm is combined with the branch-and-bound method.

Branch-and-bound is an implicit enumeration method. A binary tree is associated to the solution of the problem. The root of the tree represents the original relaxed problem. The left child and right child of a node N correspond to subproblems of the problem associated to N obtained by fixing a free variable at 0 or 1, respectively. A list $\mathcal{L}$ of current nodes to be processed is kept. The list $\mathcal{L}$ is initialized with the root node. At each step of the algorithm, a node is removed from $\mathcal{L}$ and processed. The processing of a node N starts by solving the relaxed subproblem corresponding to N. The node N is pruned, i.e., the subtree rooted at N is neglected, if one of the following cases occur:

the subproblem is infeasible, the solution for the subproblem is integral, or the solution of the subproblem is not integral but the optimal value is less than or equal to the objective value of the best solution found. In all other cases, the children of node N are added to $\mathcal{L}$. The algorithm terminates when $\mathcal{L}$ is empty, and returns the best integral solution found as the optimal solution.

The branch-and-cut algorithm differs from pure branch-and-bound in that at each node it applies a cutting-plane algorithm rather than simply solving the relaxed subproblem, and in that the descendants of a node in the tree inherit the inequalities that are added in that node by the cutting-plane algorithm.

Branch-and-cut algorithm has been largely applied to solve combinatorial optimization problems, and it has been shown in practice to be more efficient than pure cutting-planes or branch-and-bound. Successful applications of this method to large scale symmetrical travelling salesman problems are described by Padberg and Rinaldi [19] and Applegate et al [1] . The first paper decribes an experiment with 42 problems with sizes from 48 to 2,392 cities; the second one is the most recent breaktrough, solving instances with sizes ranging from 225 to 7,397 cities.

To the best of our knowledge, branch-and-cut methods had never been applied to problems in the area of combinatorial designs.

3 INTEGER PROGRAMMING FORMULATIONS FOR DESIGNS

In this section, we present integer programming formulations for problems in design theory. First, we define an important family of matrices. Let t, v and k be integers with $0 < t \leq k \leq v$, T_i denote the t-subsets of a v-set, $i = 1, \ldots, \binom{v}{t}$, and K_j denote the k-subsets of the v-set, $j = 1, \ldots, \binom{v}{k}$. Let $W^v_{t,k}$ denote the $\binom{v}{t}$ by $\binom{v}{k}$ matrix whose entries are defined by

$$\left[W^v_{t,k}\right]_{ij} = \begin{cases} 1 & \text{if } T_i \in K_j, \\ 0 & \text{otherwise.} \end{cases}$$

Several properties of these matrices are studied in [25] and [10]. It is easy to see that the t-(v,k,λ) designs correspond to the solutions $x \in \mathbb{R}^{\binom{v}{k}}$ of

$$\text{(DP)} \quad \begin{cases} W^v_{t,k}\, x = \lambda \mathbf{1}, \\ x_j \in \{0,1\},\ j = 1, \ldots, \binom{v}{k}. \end{cases}$$

The maximum packings correspond to solutions $x \in \mathbb{R}^{\binom{v}{k}}$ of

$$\text{(PDP)} \quad \begin{cases} \text{maximize} & \mathbf{1}^T x \\ \text{subject to} & W_{t,k}^v \, x \leq \lambda \mathbf{1}, \\ & x_j \in \{0,1\}, \ j = 1, \ldots, \binom{v}{k}. \end{cases}$$

Analogously, the minimum coverings correspond to solutions $x \in \mathbb{R}^{\binom{v}{k}}$ of

$$\text{(CDP)} \quad \begin{cases} \text{minimize} & \mathbf{1}^T x \\ \text{subject to} & W_{t,k}^v \, x \geq \lambda \mathbf{1}, \\ & x_j \in \{0,1\}, \ j = 1, \ldots, \binom{v}{k}. \end{cases}$$

If we are interested in designs with possibly repeated blocks, then we must replace the condition "$x_j \in \{0,1\}$" by "x_j positive integer".

Based on these integer programming formulations, we rewrite the design polytopes defined in Section 1 as

$$\begin{aligned} T_{t,v,k,\lambda} &= conv\{x \in \mathbb{R}^{\binom{v}{k}} : W_{t,k}^v \, x = \lambda \mathbf{1}, x_j \in \{0,1\}, \ j = 1, \ldots, \binom{v}{k}\}, \\ P_{t,v,k,\lambda} &= conv\{x \in \mathbb{R}^{\binom{v}{k}} : W_{t,k}^v \, x \leq \lambda \mathbf{1}, x_j \in \{0,1\}, \ j = 1, \ldots, \binom{v}{k}\}, \\ C_{t,v,k,\lambda} &= conv\{x \in \mathbb{R}^{\binom{v}{k}} : W_{t,k}^v \, x \geq \lambda \mathbf{1}, x_j \in \{0,1\}, \ j = 1, \ldots, \binom{v}{k}\}. \end{aligned}$$

For Steiner systems, i.e., for designs with $\lambda = 1$, problems (DP), (PDP) and (CDP) are special cases of set partitioning, set packing and set covering problems, respectively. The known facet inducing inequalities for these more general problems provide some facets for the design problems with $\lambda = 1$. These inequalities are surveyed on Section 4. On the other hand, very little is known about the facial structure of polyhedra corresponding to designs with $\lambda > 1$.

The following theorem relates the problems just described. It can be proven using Schönheim bounds (see [15]), but we give an alternative proof.

Theorem 3.1 *Assume that a t-(v,k,λ) design exists. Let $x^* \in \mathbb{R}^{\binom{v}{k}}$. Then the following statements are equivalent:*

i. x^* *is a solution to (DP).*

ii. x^* *is an optimal solution to (PDP).*

iii. x^* *is an optimal solution to (CDP).*

Proof Let $b = \lambda\binom{v}{t}/\binom{k}{t}$, the number of blocks in a t–(v,k,λ) design. Let x be any feasible solution to (PDP). Adding the inequalities of $W^v_{t,k}\ x \leq \lambda\mathbf{1}$, we get that $\binom{k}{t}\sum x_i \leq \binom{v}{t}\lambda$, that is, $\mathbf{1}^T x \leq \binom{v}{t}\lambda/\binom{k}{t} = b$. Analogously, for any feasible solution y of (CDP), we have that $\mathbf{1}^T y \geq b$. By the existence of a t–(v,k,λ) design, we conclude that for any optimal solution x^* of either (PDP) or (CDP) we must have $\mathbf{1}^T x^* = b$. But this equality holds if and only if $W^v_{t,k}\ x^* = \lambda\mathbf{1}$. □

Theorem 3.1 suggests that any of the three formulations may be used to find t-designs.

3.1 Designs with prescribed automorphism groups

Let V be a finite set with $|V| = v$. The set S_V of all permutations on V forms a symmetric group where composition is the group operation.

If $\pi \in S_V$ maps the design $D_1 = (V, \mathcal{B}_1)$ to the design $D_2 = (V, \mathcal{B}_2)$ then we say that D_1 and D_2 are *isomorphic*. If π fixes D_1 so that $\mathcal{B}_1 = \mathcal{B}_2$ then π is called an *automorphism* of D_1. The set of all automorphisms of a design D, denoted by $AutD$, forms a group under the operation of composition.

There are two main reasons why it is desirable to search for designs with a prescribed automorphism group. Classes of designs admitting certain automorphism groups are of special interest, such as cyclic and rotational designs. In addition, assuming that an automorphism group acts on a design allows a reduction on the size of the corresponding integer programming problem.

Let A be a permutation group acting on a set V. If $x \in V$, the *orbit* of x under A is defined by

$$A(x) = \{y \in V : \pi(x) = y \text{ for some } \pi \in A\}.$$

It is easy to see that an equivalence relation on V can be defined saying that x and y are equivalent if and only if $y \in A(x)$. The equivalent classes are the different orbits of V.

This definition is naturally extended to orbits of s-subsets of V. The orbit of an s-subset $S \in \binom{V}{s}$ is defined by

$$A(S) = \left\{ T \in \binom{V}{s} : \pi(S) = T \text{ for some } \pi \in A \right\}.$$

Let t, v and k be integers with $0 < t \leq k \leq v$, V be a set with $|V| = v$, and A be a permutation group acting on V. Let T_i, $i = 1, \ldots, \left|\binom{V}{t}/A\right|$, be representatives of the orbits of the t-subsets and K_j, $j = 1, \ldots, \left|\binom{V}{k}/A\right|$, be representatives of the orbits of the k-subsets. We define $W^A_{t,k}$ as the $\left|\binom{V}{t}/A\right|$ by $\left|\binom{V}{k}/A\right|$ matrix defined by

$$\left[W^A_{t,k}\right]_{i,j} = |\{K \in A(K_j) : T_i \subseteq K\}|.$$

Theorem 3.2 *(Kramer and Mesner [12]) Let A be a permutation group and $l = \left|\binom{V}{k}/A\right|$. The t-(v,k,λ) designs admitting an automorphism group A are the solutions $x \in \mathbb{R}^l$ of*

$$(\mathrm{DP_A}) \quad \begin{cases} W^A_{t,k}\, x = \lambda \mathbf{1}, \\ x_j \in \{0,1\}, \ j = 1, \ldots, l. \end{cases}$$

Algorithms for computing solutions of $(\mathrm{DP_A})$ are proposed in [11], [13], [22]. The following theorem shows similar formulations for packings and coverings.

Theorem 3.3 *Let A be a permutation group, $l = \left|\binom{V}{k}/A\right|$ and $c_i = |A(K_i)|$, for $i = 1, \ldots, l$. The t-(v,k,λ) packing designs admitting automorphism group A are the solutions $x \in \mathbb{R}^l$ of*

$$(\mathrm{PDP_A}) \quad \begin{cases} \text{maximize} & [c_1, \ldots, c_l]\, x \\ \text{subject to} & W^A_{t,k}\, x \leq \lambda \mathbf{1}, \\ & x_j \in \{0,1\}, \ j = 1, \ldots, l. \end{cases}$$

Analogously, the t-(v,k,λ) covering designs admitting automorphism group A are the solutions $x \in \mathbb{R}^l$ of

$$(\mathrm{CDP_A}) \quad \begin{cases} \text{minimize} & [c_1, \ldots, c_l]\, x \\ \text{subject to} & W^A_{t,k}\, x \geq \lambda \mathbf{1}, \\ & x_j \in \{0,1\}, \ j = 1, \ldots, l. \end{cases}$$

For designs with $\lambda = 1$, problems (DP_A) and (PDP_A) can be brought to the form of set partitioning and set packing problems, respectively. As we will see on Section 4, problem (DP_A) would differ from the integer programming formulation of the set partitioning problem in that $W^A_{t,k}$ is not a $0-1$ matrix. However, any column j containing an element greater than or equal to 2 can be eliminated, since for $\lambda = 1$ any solution x of (DP_A) must have $x_j = 0$. Therefore, after performing these eliminations, the problem (DP_A) becomes a special case of set partitioning. In the same way, problem (PDP_A) becomes a special case of set packing.

4 SET PARTITIONING, SET PACKING AND SET COVERING POLYHEDRA: FACETS FOR DESIGNS WITH $\lambda = 1$

As we have mentioned in Section 3, the integer programming formulations for designs with $\lambda = 1$ are special cases of set partitioning, set packing and set covering problems. In this section, we define these problems and summarize known results on the facial structure of the corresponding polyhedra.

Let $M = \{1, \ldots, m\}$, $\mathcal{S}$ be a family of subsets $S_i \subseteq M, i = 1, \ldots, n$, and c_i be a weight associated with S_i for $i = 1, \ldots, n$. A *partition* of M (relatively to $\mathcal{S}$) is a subfamily $\mathcal{P}$ of $\mathcal{S}$ such that for each $j \in M$ there exists *a unique* S in $\mathcal{P}$ with $j \in S$. The *set partitioning problem* is the problem of finding a *maximum* weight partition of M. A *packing* of M (relatively to $\mathcal{S}$) is a subfamily $\mathcal{K}$ of $\mathcal{S}$ such that for each $j \in M$ there exists *at most one* S in $\mathcal{K}$ with $j \in S$. The *set packing problem* is the problem of finding a *maximum* weight packing of M. Analogously, a *covering* of M (relatively to $\mathcal{S}$) is a subfamily $\mathcal{K}$ of $\mathcal{S}$ such that for each $j \in M$ there exists *at least one* S in $\mathcal{K}$ with $j \in S$. The *set covering problem* is the problem of finding a *minimum* weight covering of M.

Let $A \in \mathbb{R}^{m \times n}$ be the matrix whose columns are incidence vectors of members of $\mathcal{S}$, defined for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ as

$$a_{ij} = \begin{cases} 1 & \text{if } i \in S_j \\ 0 & \text{otherwise.} \end{cases}$$

The integer programming formulation for the set partitioning problem is

$$(\text{PT}) \quad \begin{cases} \text{maximize} & c^T x \\ \text{subject to} & Ax = \mathbf{1}, \\ & x_j \in \{0,1\},\ j = 1,\ldots,n, \end{cases}$$

for the set packing is

$$(\text{PK}) \quad \begin{cases} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq \mathbf{1}, \\ & x_j \in \{0,1\},\ j = 1,\ldots,n, \end{cases}$$

and for the set covering is

$$(\text{CV}) \quad \begin{cases} \text{minimize} & c^T x \\ \text{subject to} & Ax \geq \mathbf{1}, \\ & x_j \in \{0,1\},\ j = 1,\ldots,n. \end{cases}$$

These three problems were shown to be NP-hard for general matrix A (see [14]). However, for the special case of design problems, it is not known whether or not the problems lie in P or NP.

The associated polyhedra for the problems defined previously are given by

$$\begin{aligned} P_{PT} &= conv\{x \in \mathbb{R}^n : Ax = \mathbf{1}, x_j \in \{0,1\}, j = 1,\ldots,n\}, \\ P_{PK} &= conv\{x \in \mathbb{R}^n : Ax \leq \mathbf{1}, x_j \in \{0,1\}, j = 1,\ldots,n\}, \\ P_{CV} &= conv\{x \in \mathbb{R}^n : Ax \geq \mathbf{1}, x_j \in \{0,1\}, j = 1,\ldots,n\}. \end{aligned}$$

These polyhedra are polytopes since they are included in $\{x \in \mathbb{R}^n : 0 \leq x \leq 1\}$.

The most studied polytope of them is the **set packing polytope** P_{PK}. Several classes of facets were discovered in the 1970's, as we will see in next subsection. Let $G_A = (V, E)$ be the **intersection graph** of A, whose vertices correspond to the columns of A and $\{j_1, j_2\} \in E$ if and only if $a_{ij_1} = a_{ij_2} = 1$, for some $i \in \{1,\ldots,m\}$. The *vertex packing problem* on a graph $G = (V, E)$ (also known as node packing, independent set or stable set problem) is the problem of finding a maximum weight subset $U \subseteq V$ such that any $v, u \in U$ satisfy $\{v, u\} \notin E$. The associated polytope is given by $P_{VPK}(G) = conv\{x : x_{j_1} + x_{j_2} \leq 1$, for all $\{j_1, j_2\} \in E, x_j = 0$ or $1, j = 1,\ldots,n\}$. It is easy to see that the set packing problem (PK) is equivalent to the vertex packing problem in G_A and that $P_{PK} = P_{VPK}(G_A)$. Some known classes of facet inducing inequalities for $P_{VPK}(G)$ (and so for P_{PK}) are listed in next subsection. No complete description of this polytope is known for the general problem.

The **set covering polytope** P_{CV} has received more attention only recently (1980's decade). As pointed out by Cornuéjols and Sassano [8], this is probably due to the fact that the key concepts developed to the study of set packing are not immediately transferable to the set covering polytope. We need to define another graph for this problem. Let $B_A = (V, U, E)$ be a bipartite **incidence graph** of A having the vertices in U corresponding to the rows of A, the vertices in V corresponding to columns of A and for $i \in U$ and $j \in V$, we have $\{i, j\} \in E$ if and only if $a_{ij} = 1$. Given a bipartite graph $B = (V, U, E)$, a set $S \subseteq V$ is called a *cover* of U if every $u \in U$ is adjacent to at least one vertex of S. Let $P_{VCV}(B)$ denote the polytope associated to this problem. The set covering problem (CV) is equivalent to the problem of finding a minimum weight cover of U in $B_A = (V, U, E)$, and $P_{CV} = P_{VCV}(B_A)$ by definition.

The difficulty of studying the **set partitioning polytope** P_{PT} seems to be related to dimension determination. Since we do not know its dimension beforehand, it is harder to prove that a valid inequality is facet inducing. Probably this is the reason why not much is known about the facial structure of this polytope. On the other hand, the knowledge of P_{PK} and P_{CV} are helpful for the set partitioning problem. Any valid inequality for P_{PK} and for P_{CV} is also valid for P_{PT} since any partition is both a packing and a covering. Moreover, Balas and Padberg [4] show how to convert the set partitioning problem to an equivalent set packing and set covering problems.

4.1 The set packing polytope

Let $G = (V, E)$ be a graph. We will study the vertex packing polytope $P_{VPK}(G)$. Let $V' \subseteq V$ and $G' = (V', E')$ be the subgraph of G induced by V'. It is easy to see that valid inequalities for $P_{VPK}(G')$ are also valid for $P_{VPK}(G)$. The following result shows how to relate the facet inducing inequalities for them.

Theorem 4.1 *If $\sum_{j \in V'} \alpha_j x_j \leq s$ is a facet inducing inequality for $P_{VPK}(G')$, then there exist integers β_j, $0 \leq \beta_j \leq s$ such that*

$$\sum_{j \in V'} \alpha_j x_j + \sum_{j \in V \setminus V'} \beta_j x_j \leq s$$

is a facet inducing inequality for $P_{VPK}(G)$.

The proof is done by showing an algorithm (lifting procedure) to compute the coefficients β_j. This algorithm was proposed by Padberg [18] for odd holes, and generalized for any graph by Nemhauser and Trotter [16].

Now we focus on classes of valid inequalities and facet inducing ones for the vertex packing polytope. They are based on some subgraphs of G, namely cliques, odd holes, odd anti-holes, webs, anti-webs and wheels.

1. Inequalities defining $P_{VPK}(G)$. The following inequalities are trivially valid for $P_{VPK}(G)$:

$$\begin{aligned} x_i &\geq 0, && \text{for all } i \in V, \\ x_i + x_j &\leq 1, && \text{for all } \{i,j\} \in E. \end{aligned}$$

2. Clique inequalities (Fulkerson [9]). Let C be a *clique* of G, that is a maximal set $C \subseteq V$ such that the subgraph of G induced by C is complete. Then

$$\sum_{j \in C} x_j \leq 1$$

is a facet inducing inequality of $P_{VPK}(G)$.

3. Odd hole (Padberg [18]) and odd anti-hole (Nemhauser and Trotter [16]) inequalities. An *odd hole* is a circuit of odd length greater than 3 without chords. An *odd anti-hole* is a graph whose complement is an odd hole.

   If $H \subseteq V$ induces an odd hole G' of G then

$$\sum_{j \in H} x_j \leq \frac{|H| - 1}{2}$$

   is a facet inducing inequality of $P_{VPK}(G')$.

   If $A \subseteq V$ induces an odd anti-hole G' of G then

$$\sum_{j \in A} x_i \leq 2$$

   is a facet inducing inequality of $P_{VPK}(G')$.

   Figure 1 shows an odd hole and an odd anti-hole for graphs associated to packing design problems.

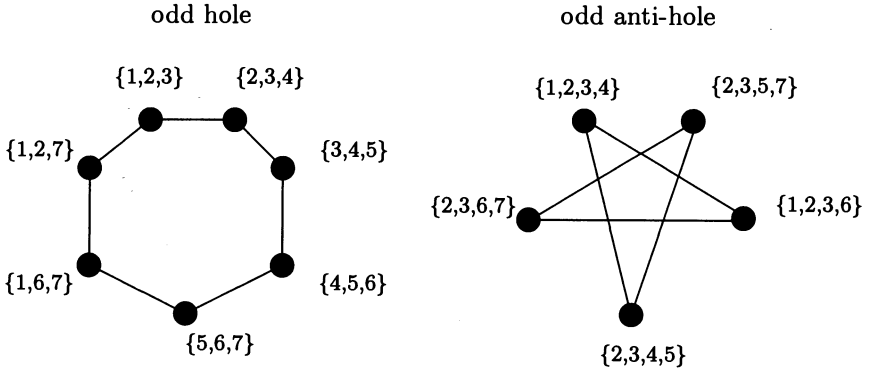


Figure 1 Examples of subgraphs of the intersection graph of $W^v_{t,k}$.

4. Web and anti-web inequalities (Trotter [24]). Webs are generalizations of cliques, holes and anti-holes. Let n and k be integers with $n \geq 2$ and $1 \leq k \leq \frac{n}{2}$. A *web* $W(n,k)$ is the graph with vertices $V_n = \{1,\ldots,n\}$ and edges defined by: $\{i,j\} \in E$ if and only if $(j-i) \text{mod } n \in \{k, k+1, ..., n-k\}$. Observe that $W(n,1)$ is a clique, $W(2s+1,s)$ is an odd hole and $W(2s+1,2)$ is an odd anti-hole. The inequality

$$\sum_{j=1}^{n} x_j \leq k$$

is facet inducing for $P_{VPK}(W(n,k))$ if and only if n and k are relatively prime.

An *anti-web* is a complement of a web, denoted by $\overline{W}(n,k)$. The inequality

$$\sum_{j=1}^{n} x_j \leq \lfloor n/k \rfloor$$

is facet inducing for $P_{VPK}(\overline{W}(n,k))$ if and only if n and k are relatively prime.

5. Wheel inequalities. An odd wheel of center v_0 is a graph with vertices $\{v_0, v_1, \ldots, v_n\}$ such that $v_1, \ldots, v_n$ forms an odd cycle and v_0 is connected

by an edge to all of the other vertices. If $W \subseteq V$ induces an odd wheel G' of G with center v_o then

$$\sum_{i \in W \setminus \{v_0\}} x_j + \frac{|W|-2}{2} x_{v_0} \leq \frac{|W|-2}{2}$$

is a facet inducing inequality for $P_{VPK}(G')$.

4.2 The set covering polytope

While the set packing polytope has been extensively studied, little is known about the set covering polytope.

Conforti, Corneil and Mahjoub [7] study the polyhedra associated to other related covering problems. They give conditions for which odd holes for the intersection graph of the associated matrix produce facet inducing inequalities for the set covering polytope.

Balas and Ng [3] study all the inequalities of the form $\alpha x \geq \alpha_0$ with $\alpha_i \in \{0, 1, 2\}$, for $i = 0, \ldots, n$. They give necessary and sufficient conditions for such inequalities to induce facets for the set covering polytope. They use Chvátal-Gomory cuts, rather than the graph theoretical approach, for generating the inequalities.

Cornuéjols and Sassano [8] and Sassano [21] show some parallel between the role of intersection graph of the set packing matrix and of the bipartite incidence graph of the set covering matrix. Some key concepts were transferred from the set packing graph to the set covering graph, and equivalent results shown. We summarize some facet inducing inequalities that were found for the set covering polytope.

1. Trivial facets. We assume that P_{CV} is full dimensional which is true if and only if there is at least two 1's per row in the $0-1$ matrix A. If this is not the case, a simple elimination of variables and rows transforms the problem in an equivalent one with this property. Then $x_j \geq 0$ defines a facet of P_{CV} if and only if after eliminating column j of A, there is at least two ones per line in A. Moreover, $x_j \leq 1$ is always facet inducing.

2. Corded cycle inequalities (Cornuéjols and Sassano [8]).

   A bipartite graph $B = (V, U, E)$ is said to be a *chorded cycle* if $|V| = |U| = n$ and there is a subset $E_H \subset E$, $|E_H| = n$ such that $H = (V, U, E_H)$ is a

cycle. The set $E_C = E \setminus E_H$ is called the *chord set* of H and each edge in E_C is said to be a *chord* of H. A cycle of a bipartite graph is said to be *odd* if its number of edges is $2k$ with k odd. A chord is said to be *odd* if it induces two odd cycles in H. Two edges $\{v_1, u_1\}$ and $\{v_2, u_2\}$ are said to be *crossing* if their nodes v_1, v_2, u_1, u_2 appear in this order on graph H. The chord set E_C of a chorded cycle $B = (V, U, E)$ is said to be *compatible* if

(a) every chord is odd, and

(b) every pair of crossing chords $\{v_1, u_1\}$ and $\{v_2, u_2\}$ are such that the remotion of u_1 together with its two incident edges from H makes the unique path in H joining v_1 and v_2 to have number of edges that is a multiple of 4.

Let $B = (V, U, E)$ be a chorded cycle with $|V| = |U| = n$ and E_C be its chord set. Then

$$\sum_{i=1}^{n} x_v \geq \frac{n+1}{2}$$

induces a facet of $P_{VCV}(B)$ if and only if the chord set E_C is compatible.

3. The (q, t)-rose inequalities (Sassano [21]). Let $p \geq q \geq 2$ be integers. A bipartite graph $R(p, q) = (V, U, E)$ is said to be a *complete q-rose of order p* if $|V| = p$, $|U| = \binom{p}{q}$ and the neighbor sets $N(u)$ for all nodes $u \in U$ are exactly all the q-subsets of V.

Let $p \geq t \geq q \geq 2$ be integers. Let $B = (V, U, E)$ be a bipartite graph with $V = \{v_1, \ldots, v_p\}$ and let $V^i = \{v_i, \ldots, v_{i+t-1}\}$ (indexes taken modulo p), for $i = 1, \ldots, p$. The graph B is said to be a *(q, t)-rose of order p* if

(a) the set U is the union of p subsets $U^1, \ldots, U^p$ such that the neighbor set of U^i is equal to V^i for $i = 1, \ldots, p$, and

(b) the subgraph induced by $V^i \cup U^i$ is a complete q-rose of order t, for each $i = 1, \ldots, p$.

Let $p \geq t \geq q \geq 2$ and $B = (V, U, E)$ be a (q, t)-rose of order p. Then the inequality

$$\sum_{v \in V} x_v \geq \left\lceil \frac{(t - q + 1)p}{t} \right\rceil$$

induces a facet of the $P_{VCV}(B)$ if and only if $p(q-1) \not\equiv 0 \bmod p$ or $t = p$.

5 DESIGNS: POLYTOPES AND ALGORITHMS

In this section, we summarize the main results concerning construction of designs seen in previous sections, and give some results on the polytopes of design problems.

The branch-and-cut method described in Section 2 can be applied to the packing design and covering design problems of Section 3. For the t-design problem, we suggest the following approach via maximum packing designs (a similar algorithm via minimum covering designs can be done as well).

Constructing t-designs via maximum packing designs.
Input: t,v,k,λ.
Output: a t–(v,k,λ)design or "nonexistence".

For any polyhedron $Q \supseteq P_{t,v,k,\lambda}$, consider the problem
$LP(Q)$: maximize $1^T x$ subject to $x \in Q$,
and let $v_{opt}(Q)$ be the objective value of any optimal solution of $LP(Q)$.
Let $b = \lambda\binom{v}{t}/\binom{k}{t}$, the number of blocks of the
desired t-design.
Apply to $LP(P_{t,v,k,\lambda})$ a branch-and-cut algorithm (Section2) with
the following modification in the stopping condition.
At each iteration, let $P_1, \ldots, P_r$ be the polytopes associated
to the nonprocessed nodes in list $\mathcal{L}$, x_{best} be the best integer
solution found, and v_{best} be its corresponding objective value.
The algorithm terminates when:
($\mathcal{L}$ is empty) or ($v_{best} = b$) or ($v_{opt}(P_i) < b$ for $i = 1, \ldots, r$).
If ($v_{best} = b$) then
return x_{best}
If ($v_{best} < b$) or ($v_{opt}(P_i) < b$ for $i = 1, \ldots, r$) then
return "nonexistence".

Remark: This algorithm differs from a simple branch-and-cut for finding maximum packing designs in that it is able to detect nonexistence of the corresponding t-designs before the maximum packing is found.

The branch-and-cut algorithm requires the knowledge of valid inequalities or facet inducing ones for the polytopes $P_{t,v,k,\lambda}$, $C_{t,v,k,\lambda}$ or $T_{t,v,k,\lambda}$. In the following, we give some results on these polytopes.

Let $e^i \in \mathbb{R}^{\binom{v}{k}}, 1 \leq i \leq \binom{v}{k}$, be defined by $(e^i)_i = 1$ and $(e^i)_j = 0$ for $j \neq i$. The following theorem gives information on the dimension of the design polytopes.

Theorem 5.1

i. If $k \leq v - t$ then $dim T_{t,v,k,\lambda} \leq \binom{v}{k} - \binom{v}{t}$.

ii. $P_{t,v,k,\lambda}$ is full dimensional.

iii. If $\lambda < \binom{v-t}{k-t}$ then $C_{t,v,k,\lambda}$ is full dimensional.

Proof

i. This is a direct consequence of the fact that $rank(W^v_{t,k}) = \binom{v}{t}$, when $k \leq v - t$ (see [10] for a proof). Since the inequalities in $W^v_{t,k}$ $x <= \lambda \mathbf{1}$ are part of the equality representation of $T_{t,v,k,\lambda}$, say $(A^=, b^=)$, we have that $rank(A^=, b^=) \geq rank(W^v_{t,k}, \lambda \mathbf{1}) = \binom{v}{t}$. Therefore, by proposition 2, $dim T_{t,v,k,\lambda} \leq \binom{v}{k} - \binom{v}{t}$.

ii. Since $\mathbf{0}, e^1, \ldots, e^{\binom{v}{k}} \in P_{t,v,k,\lambda}$ are affine independent, the result follows.

iii. Since $\lambda < \binom{v-t}{k-t}$, we have that $\mathbf{1} - e^1, \ldots, \mathbf{1} - e^{\binom{v}{k}}, \mathbf{1} \in C_{t,v,k,\lambda}$. The result follows from the affine independence of these points. □

The next proposition gives some trivial facets for the polytopes of packing and covering designs.

Proposition 3 *The inequalities $x_i \geq 0$, $i = 1, \ldots, \binom{v}{k}$, are facet inducing for $P_{t,v,k,\lambda}$. Moreover, if $\lambda < \binom{v-t}{k-t} - 1$ then they are facet inducing for $C_{t,v,k,\lambda}$.*

Proof It is enough to show $\binom{v}{k}$ affine independent points in each polytope that satisfy $x_i = 0$. For $P_{t,v,k,\lambda}$, we have that $\{e_j\}_{j \neq i} \cup \{\mathbf{0}\}$ satisfy these properties. If $\lambda < \binom{v-t}{k-t} - 1$, then $\{\mathbf{1} \setminus \{e_i, e_j\}\}_{j \neq i} \cup \{\mathbf{1} \setminus e_i\} \subseteq C_{t,v,k,\lambda}$. The result follows from the affine independence of these points. □

In the remaining of this section, we concentrate in designs with $\lambda = 1$, since in this case we can apply the results from Section 4. The following proposition gives a characterization of clique inequalities for the packing design polytope.

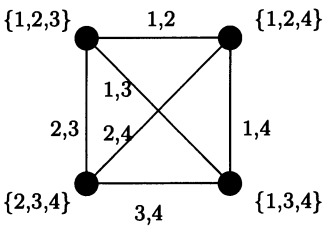
Proposition 4 *Let C be a set of k-subsets of V. Then C corresponds to a clique inequality of $P_{t,v,k,1}$ if and only if C satisfies the following properties:*

i. $|K_1 \cap K_2| \geq t$ *for any* $K_1, K_2 \in C$, *and*

ii. if there exists a k-subset K' of V such that $|K \cap K'| \geq t$ for all $K \in C$, then $K' \in C$.

Proof By the definition of $W^v_{t,k}$, property i is equivalent to C inducing a complete subgraph of $W^v_{t,k}$, and property ii is equivalent to this complete subgraph being maximal. □

Example: We give an example of a clique inequality for $P_{2,v,3,1}$, $v \geq 4$.



The set $C = \{\{1,2,3\},\{1,2,4\},\{1,3,4\},\{2,3,4\}\}$ corresponds to a clique of the intersection graph of $W^v_{2,3}$, since there is no triple intersecting all the sets in C in at least 2 points. Therefore, the inequality

$$x_{\{1,2,3\}} + x_{\{1,2,4\}} + x_{\{1,3,4\}} + x_{\{2,3,4\}} \leq 1$$

is a facet inducing inequality for $P_{2,v,3,1}$, $v \geq 4$.

The following proposition gives a sufficient condition for the inequalities in $W^v_{t,k}\ x \leq \mathbf{1}$ to be indispensable in the description of the packing design polytope.

Proposition 5 *If $v \geq (t+1)(k-t+1)$ then each one of the inequalities in $W^v_{t,k}\ x \leq \mathbf{1}$ is a facet inducing inequality for $P_{t,v,k,1}$.*

D_1	123	145	167	246	257	347	356
D_2	123	146	157	247	256	345	367
D_3	123	147	156	245	267	346	357
D_4	124	135	167	237	256	346	457
D_5	124	136	157	235	267	347	456
D_6	124	137	156	236	257	345	467
D_7	125	134	167	236	247	357	456
D_8	125	136	147	237	246	345	567
D_9	125	137	146	234	267	356	457
D_{10}	126	134	157	237	245	356	467
D_{11}	126	135	147	234	257	367	456
D_{12}	126	137	145	235	247	346	567
D_{13}	127	134	156	235	246	367	457
D_{14}	127	135	146	236	245	347	567
D_{15}	127	136	145	234	256	357	467

D_{16}	123	145	167	247	256	346	357
D_{17}	123	146	157	245	267	347	356
D_{18}	123	147	156	246	257	345	367
D_{19}	125	134	167	237	246	356	457
D_{20}	126	134	157	235	247	367	456
D_{21}	127	134	156	236	245	357	467
D_{22}	124	135	167	236	257	347	456
D_{23}	126	135	147	237	245	346	567
D_{24}	127	135	146	234	256	367	457
D_{25}	124	136	157	237	256	345	467
D_{26}	125	136	147	234	267	357	456
D_{27}	127	136	145	235	246	347	567
D_{28}	124	137	156	235	267	346	457
D_{29}	125	137	146	236	247	345	567
D_{30}	126	137	145	234	257	356	467

Table 1 The 30 distinct 2-$(7,3,1)$ designs.

Proof Each inequality of $W^v_{t,k}\ x \leq \mathbf{1}$ is of the form $\sum_{K \supseteq T} x_K \leq 1$, where T is a t-subset and each K is a k-subset of a v-set V. It is trivial that $C_T = \{K : K \supseteq T\}$ satisfies property i in Proposition 4. By the Erdös-Ko-Rado theorem (see [26]), since $v \geq (t+1)(k-t+1)$, we have that the size of C_T is maximum among the sets satisfying property i. Therefore, set C_T also satisfies property ii in Proposition 4. □

The polytope of the Fano planes

Now we study one particular t-design polytope: $T_{2,7,3,1}$. We provide some information about this polytope to serve as an example.

The vertices of this polytope are the incidence vectors of the $2-(7,3,1)$ designs. These designs are symmetric, since the number of blocks is equal to the number of points. They are also known to correspond to projective planes of order 2. Table 1 shows these designs.

One can classify each pair of such designs in one of 3 types: the pairs that intersect in 3 blocks, the pairs that intersect in 1 block, and the pairs that intersect in no block. Figure 2 represents these intersections. The edges in G2 connect designs that intersect in 3 blocks, the edges in G1 connect designs that intersect in 1 block, the edges in G3 (the ones that are missing between the top and bottom vertices) connect designs that intersect in no block.

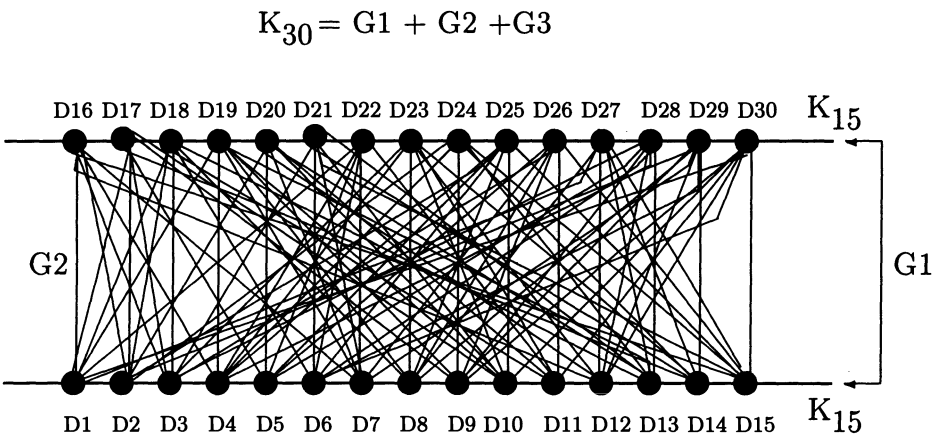


Figure 2 The three types of intersection among $2-(7,3,1)$ designs

It turns out that every pair of designs are linked by an edge, i.e., a face of dimension 1, in the polytope $T_{2,7,3,1}$. This can be verified, by examining every pair of distinct designs (D_i, D_j). If $\mathcal{B}_i, \mathcal{B}_j$ are their respective set of blocks, we observe that $(\mathcal{B}_i \setminus \mathcal{B}_j) \cup (\mathcal{B}_j \setminus \mathcal{B}_i)$ induces a connected subgraph of $W^7_{2,3}$, and by theorem 6.2 in [6], there is an edge between D_i and D_j in $P_{2,7,3,1}$. Since $T_{2,7,3,1}$ is a face of $P_{2,7,3,1}$, there is an edge connecting D_i and D_j in $T_{2,7,3,1}$.

Other properties of this polytope were determined using the PORTA software by Thomas Christof and Andreas Loebel. Given the 30 vertices of $T_{2,7,3,1}$ the software computed its equality and inequality representation. The information obtained is summarized in the following:

- $T_{2,7,3,1} \in \mathbb{R}^{35}$ has dimension 14 (its dimension achieves the upper bound of Theorem 5.1).
- Its equality system is given only by $W^7_{2,3}x = \mathbf{1}$.
- Its inequality system contains 155 facet inducing inequalities of two types. Thirty five of these inequalities are equivalent to $x_i \geq 0$, for $i = 1, \ldots, 35$. Each of these inequalities passes through exactly 24 vertices . Each of the others inequalities passes through exactly 14 vertices.
- There are 84 inequalities passing through each vertex.

Acknowledgements

The author would like to thank Rudi Mathon for suggesting the polyhedral approach to design problems, and for his valuable comments on this paper.

REFERENCES

[1] D. APPLEGATE, R. BIXBY, V. CHVÁTAL AND W. COOK, Finding cuts in the TSP, *preliminary report*, August 1994.

[2] A. BACHEM AND M. GRÖTSCHEL, New aspects of polyhedral theory, in *Modern Applied Mathematics, Optimization and Operations Research*, North-Holland Publishing Company, 1982, 51–106.

[3] E. BALAS AND S.M. NG, On the set covering polytope: I. all the facets with coefficients in $\{0, 1, 2\}$, *Mathematical Programming* **43**(1989), 57–69.

[4] E. BALAS AND M.W. PADBERG, Set partitioning: a survey, *SIAM Review* **18**(1976), 710–760.

[5] V. CHVÁTAL, Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete Mathematics* **4**(1973), 305–337.

[6] V. CHVÁTAL, On certain polytopes associated with graphs, *Journal of Combinatorial Theory(B)***18**(1975), 138–154.

[7] M. CONFORTI, D.G. CORNEIL AND A.R. MAHJOUB, K_i-covers I: complexity and polytopes, *Discrete Mathematics* **58**(1986), 121–142.

[8] G. CORNUÉJOLS AND A. SASSANO, On the 0,1 facets of the set covering polytope, *Mathematical Programming* **43**(1989), 45–55.

[9] D.R. FULKERSON, Blocking and anti-blocking pairs of polyhedra, *Mathematical programming* **1**(1971), 168–194.

[10] C.D. GODSIL, Linear algebra and designs, *preprint*, 1994.

[11] E.S. KRAMER, D.W. LEAVITT AND S.S. MAGLIVERAS, Construction procedures for t-designs and the existence of new simple 6-designs, *Annals of Discrete Mathematics* **26**(1985), 247–274.

[12] E.S. KRAMER AND D.M. MESNER, t-designs on hypergraphs, *Discrete Mathematics* **15**(1976), 263–296.

[13] D.L. KREHER AND S.P. RADZISZOWSKI, Finding simple t-designs by using basis reduction, *Congressus Numerantium* **55**(1986), 235–244.

[14] J.K. LENSTRA AND A.H.G. RINNOOY KAN, Complexity of packing, covering and partitioning, *in Packing and Covering in Combinatorics*, A. Schrijver, ed., Mathematical Centre Trats 106, 1979, 275–291.

[15] W.H. MILLS AND R.C. MULLIN, Coverings and Packings, in *Contemporary Design Theory: a collection of surveys*, J.H. Dinitz and D.R. Stinson, eds., Wiley, 1992, 371–399.

[16] G.L. NEMHAUSER AND L.E. TROTTER,JR., Properties of vertex packing and independence system polyhedra, *Mathematical Programming* **6**(1974), 48–61.

[17] G.L. NEMHAUSER AND L.A. WOLSEY, *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988, 83–113.

[18] M.W. PADBERG, On the facial structure of set packing polyhedra, *Mathematical Programming* **5**(1973), 199–215.

[19] M.W. PADBERG AND G. RINALDI, A branch-and-cut algorithm for the solution of large-scale traveling salesman problems, *SIAM Review* **33**(1991), 60–100.

[20] W.R. PULLEYBLANK, Polyhedral Combinatorics, in *Handbooks in OR & MS*, vol.1, N.L. Nemhauser et al., eds., North-Holland, 1989, 371–446.

[21] A. SASSANO, On the facial structure of the set covering polytope, *Mathematical Programming* **44**(1989). 181–202.

[22] B. SCHMALZ, The t-designs with prescribed automorphism group, new simple 6-designs, *Journal of Combinatorial Designs* **1**(1993), 125–170.

[23] A. SCHRIJVER, On cutting planes, in *Combinatorics 79 Part II, Annals of Discrete Mathematics* **9**(1980), 291–296.

[24] L.E. TROTTER,JR., A class of facet producing graphs for vertex packing polyhedra, *Discrete Mathematics* **12**(1975), 373–388.

[25] R.M. WILSON, On the theory of t-designs, in *Enumeration and Designs*, D.M. Jackson and S.A. Vanstone, eds., Academic Press, Toronto, 1984, 19–49.

[26] R.M. WILSON, The exact bound on the Erdös-Ko-Rado theorem, *Combinatorica* **4**(1984), 247–257.

[27] E. ZEHENDNER, Methoden der Polyedertheorie zur Herleitung von oberen Schranken für die Mächtigkeit von Blockcodes, PhD thesis, Universität Augsburg, 1986.

10

ANOTHER LOOK AT LARGE SETS OF STEINER TRIPLE SYSTEMS

Martin J. Sharry and Anne Penfold Street

Centre for Combinatorics,
Department of Mathematics,
The University of Queensland,
Queensland 4072 Australia

ABSTRACT

If $v \equiv 1$ or 3 mod 6 and $v > 7$, then there exists a large set of Steiner triple systems of order v, $LS(STS(v))$.

This result was largely known by 1984, though the six cases which were unsettled then have since been solved, and the proof simplified. The proof is by construction, using a fairly small number of initial large sets. Each construction is explained, often with the aid of tables of blocks. In the interests of brevity, proofs and complete examples are not given, though the ingredients needed for the construction of the examples are. The recursive constructions are of three types:

(i) extension of a large set of $STS(v)$ to a large set of $STS(3v)$ by using idempotent commutative quasigroups;

(ii) extension of a large set of $STS(v)$ to a large set of $STS(2v+1)$ by using good one–factorizations;

(iii) special constructions to fill in the remaining cases.

1 INTRODUCTION

Let X be a set of v elements, that is, a *v–set.* Let $\binom{X}{k}$ denote the collection of all the $\binom{v}{k}$ k–subsets of X. Various versions of the following question have been studied for some time: How may the family $\binom{X}{k}$ be partitioned in some 'nice' way ?

W. D. Wallis (ed.), Computational and Constructive Design Theory, 255–335.
© 1996 *Kluwer Academic Publishers.*

The earliest case considered was that where $k = 2$. It is most easily described using graph theory. A *graph* $G = (V, E)$ consists of a finite non–empty set, V, of *vertices*, and a finite set, E, of *edges*, where each edge is associated with two vertices called its *endpoints*. It is convenient to name an edge by its endpoints, say $\{v_1, v_2\}$, usually abbreviated to v_1v_2. The *complete graph*, K_v, is a graph with v vertices and all $\binom{v}{2}$ possible edges, so that each vertex in the graph is joined by an edge to every other vertex. If $k = 2$, then $\binom{X}{k}$ may be regarded as the collection of edges of K_v.

A set of edges of K_v such that every vertex is incident with precisely one edge in the set is called a *one–factor*, and a collection of one–factors which partition the set of edges of the graph is called a *one–factorization*. A survey of one–factorizations has been given by Mendelsohn and Rosa [15]. If v is odd, then K_v clearly has no one-factorization but for v even, one obvious way to restate our question is to ask whether K_v has a one–factorization.

In fact if $v = 2n$, then a one–factorization of K_v always exists. Each one–factor consists of n edges, and a one–factorization consists of $2n-1$ edge–disjoint one–factors. The easiest construction of a one–factorization of K_{2n} is also one of the oldest: let $V = \mathbb{Z}_{2n-1} \cup \{\infty\}$, where $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$, let $F_0 = \{\{j, -j\} : j \in \mathbb{Z}_n \setminus \{0\}\} \cup \{\{0, \infty\}\}$, and finally, for $i = 0, 1, \ldots, 2n-2$, let $F_i = F_0 + i$, where $\infty + i = \infty$ and addition is otherwise taken modulo $2n-1$. These one–factors $F_0, F_1, \ldots, F_{2n-2}$ form a one–factorization of K_{2n} generated from a starter set by addition modulo $2n-1$, with one fixed point. For example, if $n = 4$, then $V = \mathbb{Z}_7 \cup \{\infty\}$, and we obtain the following one–factorization of K_8: $F_0 = \{16, 25, 34, 0\infty\}$; $F_1 = \{20, 36, 45, 1\infty\}$; $F_2 = \{31, 40, 56, 2\infty\}$; $F_3 = \{42, 51, 60, 3\infty\}$; $F_4 = \{53, 62, 01, 4\infty\}$; $F_5 = \{64, 03, 12, 5\infty\}$; $F_6 = \{05, 14, 23, 6\infty\}$.

We have now seen that for $v = 2n$ we can partition the set of all the copies of K_2 (edges) covered by K_v into a family of sets of copies of K_2 such that the copies of K_2 in each set cover all of the vertices of K_v once each, and no two copies of K_2 belonging to the same set contain a common point (K_1).

Suppose that we consider instead the copies of K_3 (triples) covered by K_v. An analogue of a one–factorization is a partition of the set of all triples chosen from the set X into a family of sets of triples such that the triples in each set cover all of the vertices of K_v, r times each for some fixed r, and all the edges of K_v once each, and no two triples belonging to the same set contain a common edge (K_2). Each set of triples, the analogue of a one–factor, is called a *Steiner triple system* on v points, often denoted by $STS(v)$. The family of sets, containing

all the triples and hence the analogue of a one–factorization, is called a *large set* of Steiner triple systems on v points, often denoted by $LS\,(STS\,(v))$.

More formally, let X be a finite set of elements, called *points*, and let $\mathcal{T} = \{\,T_i : i \in I\,\}$ be a family of b triples T_i of X, called *blocks*. The pair $(X, \mathcal{T})$ is a Steiner triple system provided that every element occurs in r of the blocks, for some constant r, and every pair of elements occurs in precisely one of the blocks. The *order* of a Steiner triple system $(X, \mathcal{B})$ is $|\,X\,|$ (the cardinality of X), denoted by v. From this definition, a counting argument shows that in any $STS\,(v)$, $3b = v\,r$, $v - 1 = 2r$, and hence that a Steiner triple system of order v can only exist for $v \equiv 1$ or 3 mod 6. Kirkman [7] showed that this necessary condition is also sufficient. For example, let $X_1 = \{\,1, 2, 3, 4, 5, 6, 7\,\}$ and

$$\mathcal{T}_1 = \{\,123\,,\ 145\,,\ 167\,,\ 246\,,\ 257\,,\ 347\,,\ 356\,\}\,.$$

Then $(X_1, \mathcal{T}_1)$ is a Steiner triple system of order 7, or $STS\,(7)$.

Two Steiner triple systems, $(X, \mathcal{T})$ and $(X', \mathcal{T}')$, are *disjoint* if they have no blocks in common. For instance, if $X_2 = X_1$ and if

$$\mathcal{T}_2 = \{\,124\,,\ 136\,,\ 157\,,\ 237\,,\ 256\,,\ 345\,,\ 467\,\}\,,$$

then the two Steiner triple systems $(X_1, \mathcal{T}_1)$ and $(X_2, \mathcal{T}_2)$ are disjoint.

In a Steiner triple system, $(X, \mathcal{T})$, a *parallel class* of blocks is a subfamily $\mathcal{P} \subseteq \mathcal{T}$ of disjoint blocks, the union of which equals X. A *resolvable* Steiner triple system, $STS\,(v)$, is one in which the set of blocks can be partitioned into $(v - 1)/2$ parallel classes. A resolvable $STS\,(v)$ exists for all $v \equiv 3$ mod 6; see Ray–Chaudhuri and Wilson [16]. For example, if $X_3 = \{\,1, 2, 3, 4, 5, 6, 7, 8, 9\,\}$ and

$$\mathcal{T}_3 = \{\,123\,,\ 147\,,\ 159\,,\ 168\,,\ 249\,,\ 258\,,\ 267\,,\ 348\,,\ 357\,,\ 369\,,\ 456\,,\ 789\,\}\,,$$

then $(X_3, \mathcal{T}_3)$ is a resolvable $STS\,(9)$, with four parallel classes made up of the following sets of blocks:

$$\{\,123\,,\ 456\,,\ 789\,\}\ ;\quad \{\,147\,,\ 258\,,\ 369\,\}\ ;$$
$$\{\,159\,,\ 267\,,\ 348\,\}\ ;\quad \{\,168\,,\ 249\,,\ 357\,\}\,.$$

These are conveniently regarded as the rows, columns, forward diagonals and backward diagonals of the following 3×3 array.

$$\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}$$

Let X be a v–set. A *large set of Steiner triple systems*, $LS\,(STS\,(v))$, is a set of mutually disjoint $STS\,(v)$ which partition $\binom{X}{3}$, just as the set of mutually disjoint one–factors of a one–factorization of K_{2n}, $OF(K_{2n})$, partition $\binom{X}{2}$.

Theorem. *There exists an* LS(2–$(v, 3, 1)$) *for* $v \equiv 1$ *or* 3 mod 6, $v > 7$.

Proof. This result is due to Rosa [17], Teirlinck [21] and [22] and Lu [8], [9], [10], [11], [12], [13] and [14]. The details follow. □

2 DIRECT CONSTRUCTIONS FROM QUASIGROUPS

Bose [1] gave a direct construction of a $STS\,(6n + 3)$ from a Latin square (or quasigroup) of order $2n+1$. This idea was adapted by Teirlinck [21] to construct a $LS\,(STS\,(3v))$ from a $LS\,(STS\,(v))$. We outline these constructions here.

A *Latin square of order* n is an $n \times n$ array with entries chosen from a set of n elements $\{v_1, v_2, \ldots, v_n\}$ such that each row and each column contains each element of this set precisely once; we choose the underlying set to be either $\{1, 2, \ldots, n\}$ or $\{0, 1, \ldots, n-1\}$. Some small examples are shown in Tables 1 and 2.

Let $Q = \{x_1, x_2, \ldots, x_v\}$ be a finite set of v elements on which a binary operation $\circ$ is defined. The pair $(Q, \circ)$ is a *quasigroup of order* v if, for any two elements, $x_i, x_j \in Q$, the two equations $x_i \circ y_1 = x_j$ and $y_2 \circ x_i = x_j$ have exactly one solution each in Q. The Cayley table of a quasigroup $(Q, \circ)$ is a Latin square, and any Latin square, equipped with a headline and a sideline, is the Cayley table of a quasigroup.

A quasigroup $(Q, \circ)$ is *commutative* if it satisfies the additional condition $x_i \circ x_j = x_j \circ x_i$ for any $x_i, x_j \in Q$. Its Cayley table is a symmetric Latin square provided that its elements appear in the same order in the headline and the sideline. Similarly $(Q, \circ)$ is *idempotent* if it satisfies the additional condition $x_i \circ x_i = x_i$ for any $x_i \in Q$. Its Cayley table is an idempotent Latin square and, if its elements appear in the same order in the headline and sideline, then they appear again in that order, down its main diagonal. Table 1 shows examples of commutative and idempotent quasigroups of order 6.

$\circ$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

$\circ$	1	2	3	4	5	6
1	1	6	2	5	3	4
2	4	2	6	3	1	5
3	2	5	3	6	4	1
4	5	3	1	4	6	2
5	6	1	4	2	5	3
6	3	4	5	1	2	6

Table 1 The Cayley tables of commutative and idempotent quasigroups of order 6 respectively

In Bose's construction, we let $(Q, \circ)$ be an idempotent commutative quasigroup of order $2n+1$, where $Q = \{1, 2, \ldots, 2n+1\}$. We let $S = Q \times \{1, 2, 3\}$ and denote any element (x, e) of S by x_e. We define a collection $\mathcal{T}$ of triples of S as follows :

(i) for every $x \in Q$, $\{x_1, x_2, x_3\} \in \mathcal{T}$;

(ii) for every $x, y \in Q$, $x \neq y$, $z = x \circ y$,

$$\{x_1, y_1, z_2\},\ \{x_2, y_2, z_3\},\ \{x_3, y_3, z_1\} \in \mathcal{T}\,.$$

Then $(S, \mathcal{T})$ is a $STS\,(6n+3)$. For instance, a $STS\,(15)$ can be constructed from the idempotent commutative quasigroup of order 5 given in Table 2. Its blocks are listed in Table 3.

$\circ$	1	2	3	4	5
1	1	3	5	2	4
2	3	2	4	5	1
3	5	4	3	1	2
4	2	5	1	4	3
5	4	1	2	3	5

Table 2 An idempotent commutative quasigroup of order 5

Teirlinck's construction [21] for a $LS\,(STS\,(3v))$ from a $LS\,(STS\,(v))$ is based on Bose's construction for individual Steiner triple systems. The large sets of

Type (i) triples :	$1_1\,1_2\,1_3$	$2_1\,2_2\,2_3$	$3_1\,3_2\,3_3$	$4_1\,4_2\,4_3$	$5_1\,5_2\,5_3$
Type (ii) triples :	$1_1\,2_1\,3_2$	$1_1\,3_1\,5_2$	$1_1\,4_1\,2_2$	$1_1\,5_1\,4_2$	$2_1\,3_1\,4_2$
	$2_1\,4_1\,5_2$	$2_1\,5_1\,1_2$	$3_1\,4_1\,1_2$	$3_1\,5_1\,2_2$	$4_1\,5_1\,3_2$
	$1_2\,2_2\,3_3$	$1_2\,3_2\,5_3$	$1_2\,4_2\,2_3$	$1_2\,5_2\,4_3$	$2_2\,3_2\,4_3$
	$2_2\,4_2\,5_3$	$2_2\,5_2\,1_3$	$3_2\,4_2\,1_3$	$3_2\,5_2\,2_3$	$4_2\,5_2\,3_3$
	$1_3\,2_3\,3_1$	$1_3\,3_3\,5_1$	$1_3\,4_3\,2_1$	$1_3\,5_3\,4_1$	$2_3\,3_3\,4_1$
	$2_3\,4_3\,5_1$	$2_3\,5_3\,1_1$	$3_3\,4_3\,1_1$	$3_3\,5_3\,2_1$	$4_3\,5_3\,3_1$

Table 3 A Steiner triple system of order 15

Steiner triple systems are obtained by 'twisting' the blocks of the Steiner triple system obtained from that construction.

Let $Q = \{1, 2, \ldots, v\}$ and $S = Q \times \{1, 2, 3\}$. Denote any element (x, e) of S by x_e. Further, let $Q_e = Q \times \{e\}$ for $e \in \{1, 2, 3\}$. Let

$$\{(Q_e, \mathcal{T}_k) : k = 1, 2, \ldots, v-2\}$$

be a $LS(STS(v))$ based on the set Q_e for $e \in \{1, 2, 3\}$. Let α be a permutation of the elements of Q consisting of a single cycle of length v. Define the following collections of Steiner triple systems on S.

(i) For $j = 1, 2, \ldots, v$, define a collection $\mathcal{C}_j$ of triples of S as follows:

(a) for every $x \in Q$, $\{x_1, \alpha^j(x_2), \alpha^{2j}(x_3)\} \in \mathcal{C}_j$;

(b) for every $x, y \in Q$, $x \neq y$, $z = x \circ y$,

$$\{x_1, y_1, \alpha^j(z_2)\},\ \{x_2, y_2, \alpha^j(z_3)\},\ \text{and}\ \{x_3, y_3, \alpha^{-2j}(z_1)\} \in \mathcal{C}_j.$$

Then $(S, \mathcal{C}_j)$ is a $STS(3v)$ for $j = 1, 2, \ldots, v$, and these $STS(3v)$ are pairwise disjoint.

(ii) For $j = 1, 2, \ldots, v$, define a collection $\mathcal{D}_j$ of triples of S as follows:

(a) for every $x \in Q$, $\{x_1, \alpha^j(x_2), \alpha^{2j+1}(x_3)\} \in \mathcal{D}_j$;

(b) for every $x, y \in Q$, $x \neq y$, $z = x \circ y$,

$$\{x_1, y_1, \alpha^{2j+1}(z_3)\},\ \{x_2, y_2, \alpha^{-j}(z_1)\},\ \text{and}\ \{x_3, y_3, \alpha^{-j-1}(z_2)\} \in \mathcal{D}_j.$$

Then $(S, \mathcal{D}_j)$ is a $STS(3v)$ for $j = 1, 2, \ldots, v$, and these $STS(3v)$ are pairwise disjoint.

(iii) For $k = 1, 2, \ldots, v-2$ define a collection $\mathcal{E}_k$ of triples of S as follows:

(a) include the triples of $(Q_e, \mathcal{T}_k)$ for $e \in \{1, 2, 3\}$;

(b) for $j = 1, 2, \ldots, v$, and for every $x \in Q$,

$$\{ x_1 , \alpha^{j-1}(x_2) , \alpha^{2j+k-1}(x_3) \} \in \mathcal{E}_j .$$

Then $(S, \mathcal{E}_k)$ is a $STS(3v)$ for $k = 1, 2, \ldots, v-2$, and these $STS(3v)$ are pairwise disjoint.

Then a $LS(STS(3v))$ based on S can be constructed by taking the following collections of disjoint $STS(3v)$ based on S: $(S, \mathcal{C}_j)$ and $(S, \mathcal{D}_j)$ for $j = 1, 2, \ldots, v$; and $(S, \mathcal{E}_k)$ for $k = 1, 2, \ldots, v-2$.

Counting arguments which show that each triple of elements of S occurs precisely once in this set of Steiner triple systems follow from those in Doyen [4].

3 RECURSIVE CONSTRUCTIONS

Let $(S, \mathcal{T})$ be a Steiner triple system of order v. Let X be a set of $v+1$ points which is disjoint from S and let K_X denote the complete graph K_{v+1} with vertices labelled by the elements of X. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_v\}$ be any one–factorization of the complete graph K_X. Let α be a one–to–one mapping from $\{1, 2, \ldots, v\}$ onto S. Define a collection $\mathcal{T}'$ of triples of $S \cup X$ as follows :

(i) the triples of $\mathcal{T}$ belong to $\mathcal{T}'$; and

(ii) for every $i \in \{1, 2, \ldots, v\}$, $\{x, y, a\} \in \mathcal{T}'$ if and only if $\{x, y\} \in F_i$ and $\alpha(i) = a$.

Then $(S \cup X, \mathcal{T}')$ is a $STS(2v+1)$. For example, a $STS(15)$ can be constructed from a $STS(7)$, $(S, \mathcal{T})$, and a one–factorization of K_X, where $|X| = 8$. Let $S = \{1, 2, 3, 4, 5, 6, 7\}$, $X = \{8, 9, A, B, C, D, E, F\}$ and

$$\mathcal{T} = \{124, 235, 346, 457, 561, 672, 713\}.$$

Then $(S, \mathcal{T})$ is a Steiner triple system of order 7. A one–factorization of K_X is given in Table 4. Finally, let α be the identity mapping on S. The blocks of the $STS(15)$ obtained from this construction are listed in Table 5.

89	8A	8B	8C	8D	8E	8F
AB	9B	9A	9D	9C	9F	9E
CD	CE	CF	AE	AF	AC	AD
EF	DF	DE	BF	BE	BD	BC
F_1	F_2	F_3	F_4	F_5	F_6	F_7

Table 4 A one–factorization of K_X where $X = \{8, 9, A, B, C, D, E, F\}$

Type (i) triples :	124	235	346	457	561	672	713
Type (ii) triples :	891	8A2	8B3	8C4	8D5	8E6	8F7
	AB1	9B2	9A3	9D4	9C5	9F6	9E7
	CD1	CE2	CF3	AE4	AF5	AC6	AD7
	EF1	DF2	DE3	BF4	BE5	BD6	BC7

Table 5 A Steiner triple system of order 15

Using a one–factorization with special properties, Rosa [17] adapted this technique to construct a $LS\,(STS\,(2v+1))$ from a $LS\,(STS\,(v))$.

The graph $G = (V, E)$ is called *bipartite* if V can be partitioned into two subsets, V_1 and V_2, such that each edge in E is of the form $\{v_1, v_2\}$, where $v_1 \in V_1$ and $v_2 \in V_2$. A *k–good set of one–factors of K_{2n}* is a set of k edge–disjoint one–factors whose union is a bipartite graph.

Any two edge–disjoint one–factors of K_{2n} form a 2–good set of one–factors. A *good one–factorization of K_{2n}* is a one–factorization of K_{2n} which contains a 3–good set of one–factors, and a good one–factorization of K_{2n} exists for all $n \geq 4$ [17]. Since the proof deals separately with the cases n odd and n even, we give two examples: seven one–factors of K_8 whose union forms a good one–factorization of K_8 are listed in Table 6, and nine one–factors of K_{10} whose union forms a good one–factorization of K_{10} are listed in Table 7. In each case, $\{F_1, F_2, F_3\}$ form a 3–good set.

F_1	F_2	F_3	F_4	F_5	F_6	F_7
12	14	16	13	17	18	15
34	23	27	24	28	25	26
56	58	38	57	35	36	37
78	67	45	68	46	47	48

Table 6 A good one–factorization of K_8 with a 3–good set F_1, F_2, F_3

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
17	18	19	1A	14	16	13	15	12
28	29	2A	26	23	25	27	24	35
39	3A	36	37	5A	34	45	38	49
4A	46	47	48	69	7A	68	6A	67
56	57	58	59	78	89	9A	79	8A

Table 7 A good one–factorization of K_{10} with a 3–good set F_1, F_2, F_3

Let $N = \{1, 2, \ldots, 2n\}$, $n \geq 4$, and let $\mathcal{F} = \{F_1, F_2, \ldots, F_{2n-1}\}$ be a good one–factorization of K_N. Let $\{F_1, F_2, F_3\}$ be a 3–good set of one–factors in $\mathcal{F}$. We construct the *Latin square associated with the good one–factorization* $\mathcal{F}$.

Denote by $F_{i,j}$ the union of one–factors F_i and F_j. Then $F_{i,j}$ is a *two–factor*, that is, a subgraph of K_N with two edges incident with each vertex, and consists of edge–disjoint cycles of even length. Assign an orientation to each cycle of $F_{2k,2k+1}$, $k = 1, 2, \ldots, n-1$. The notation $(r, s) \in \vec{F}_{i,j}$ means that the directed edge (r, s) belongs to the directed two–factor $\vec{F}_{i,j}$. Finally, assign an orientation to each cycle of $\vec{F}_{1,2}$ and $\vec{F}_{1,3}$ so that for $\{r, s\} \in F_2$, $(r, s) \in \vec{F}_{1,2}$ if and only if $(s, r) \in \vec{F}_{2,3}$, and for $\{r, s\} \in F_3$, $(r, s) \in \vec{F}_{1,3}$ if and only if $(s, r) \in \vec{F}_{2,3}$. Such an assignment is always possible since $\{F_1, F_2, F_3\}$ is a 3–good set and hence, for $\{r, s\} \in F_3$, $(r, s) \in \vec{F}_{1,2}$ if and only if $(s, r) \in \vec{F}_{1,3}$.

Sets of directed two–factors obtained from the good one–factorizations of Table 6 and Table 7 are shown in Table 8 and Table 9 respectively.

$$\vec{F}_{1,2} = \{ (4,1), (1,2), (2,3), (3,4) \} \cup \{ (8,5), (5,6), (6,7), (7,8) \}$$

$$\vec{F}_{1,3} = \{ (5,4), (4,3), (3,8), (8,7), (7,2), (2,1), (1,6), (6,5) \}$$

$$\vec{F}_{2,3} = \{ (1,4), (4,5), (5,8), (8,3), (3,2), (2,7), (7,6), (6,1) \}$$

$$\vec{F}_{4,5} = \{ (1,3), (3,5), (5,7), (7,1) \} \cup \{ (2,4), (4,6), (6,8), (8,2) \}$$

$$\vec{F}_{6,7} = \{ (1,8), (8,4), (4,7), (7,3), (3,6), (6,2), (2,5), (5,1) \}$$

Table 8 Directed two–factors associated with the good one–factorization of K_8 in Table 6

$$\vec{F}_{1,2} = \{ (8,1), (1,7), (7,5), (5,6), (6,4), (4,\mathtt{A}), (\mathtt{A},3), (3,9), (9,2), (2,8) \}$$

$$\vec{F}_{1,3} = \{ (5,8), (8,2), (2,\mathtt{A}), (\mathtt{A},4), (4,7), (7,1), (1,9), (9,3), (3,6), (6,5) \}$$

$$\vec{F}_{2,3} = \{ (1,8), (8,5), (5,7), (7,4), (4,6), (6,3), (3,\mathtt{A}), (\mathtt{A},2), (2,9), (9,1) \}$$

$$\vec{F}_{4,5} = \{ (1,\mathtt{A}), (\mathtt{A},5), (5,9), (9,6), (6,2), (2,3), (3,7), (7,8), (8,4), (4,1) \}$$

$$\vec{F}_{6,7} = \{ (1,6), (6,8), (8,9), (9,\mathtt{A}), (\mathtt{A},7), (7,2), (2,5), (5,4), (4,3), (3,1) \}$$

$$\vec{F}_{8,9} = \{ (1,5), (5,3), (3,8), (8,\mathtt{A}), (\mathtt{A},6), (6,7), (7,9), (9,4), (4,2), (2,1) \}$$

Table 9 Directed two–factors associated with the good one–factorization of K_{10} in Table 7

We now define a quasigroup $(N, \circ)$ as follows :

$$
\begin{array}{lll}
(i) & r \circ 1 = s & \text{whenever } (r,s) \in \vec{F}_{2,3}\,, \\
(ii) & r \circ 2 = s & \text{whenever } (r,s) \in \vec{F}_{1,2}\,, \\
(iii) & r \circ 3 = s & \text{whenever } (r,s) \in \vec{F}_{1,3}\,, \\
(iv) & \left.\begin{array}{c} r \circ (2k) = s \\ s \circ (2k+1) = r \end{array}\right\} & \text{whenever } (r,s) \in \vec{F}_{2k,2k+1},\ k = 2,3,\ldots,n-1\,, \\
(v) & r \circ (2n) = r, & r = 1,2,\ldots,2n\,.
\end{array}
$$

The Latin square of order $2n$ associated with this quasigroup will be denoted by N^*; it is the one we need. In particular it has the property that no element of its last column is contained in any Latin subsquare of order 2.

The sets of directed two–factors given above are used in this construction to obtain the Latin squares shown in Table 10.

4	2	6	3	7	8	5	1
7	3	1	4	8	5	6	2
2	4	8	5	1	6	7	3
5	1	3	6	2	7	8	4
8	6	4	7	3	1	2	5
1	7	5	8	4	2	3	6
6	8	2	1	5	3	4	7
3	5	7	2	6	4	1	8

8	7	9	A	4	6	3	5	2	1
9	8	A	3	6	5	7	1	4	2
A	9	6	7	2	1	4	8	5	3
6	A	7	1	8	3	5	2	9	4
7	6	8	9	A	4	2	3	1	5
3	4	5	2	9	8	1	7	A	6
4	5	1	8	3	2	A	9	6	7
5	1	2	4	7	9	6	A	3	8
1	2	3	6	5	A	8	4	7	9
2	3	4	5	1	7	9	6	8	A

Table 10 The Latin squares associated with the good one–factorizations of K_8 and K_{10} in Tables 6 and 7 respectively

As well as a good one–factorization, we need the idea of a *Steiner quadruple system* of order $v+1$, usually denoted by $SQS\,(v+1)$. This is a design $(Y, \mathcal{B})$ of blocks chosen from $\binom{Y}{4}$ in such a way that every triple in $\binom{Y}{3}$ is covered by precisely one block of $\mathcal{B}$. For example, suppose we take the $STS\,(7)$ $(X_1, \mathcal{T}_1)$ given in the introduction and adjoin the extra element 8 to the set X_1 giving Y_1. Next we adjoin 8 to each triple of $\mathcal{T}_1$ giving seven quadruples, and adjoin to that set the complement in X of each quadruple. This gives a set of 14 quadruples $\mathcal{Q}_1$ such that $(Y_1, \mathcal{Q}_1)$ is a $SQS\,(8)$.

Given a $SQS\,(v+1)$ based on the set Y, suppose we *contract* it on each point in turn. That is, for a given point x, we choose each quadruple to which x belongs, and delete x to give a triple. This collection of triples is a $STS\,(v)$ based on

the set $Y \setminus x$ and the collection of $v+1$ Steiner triple systems constructed in this way is a partition of the set of triples $\binom{Y}{3}$ into mutually disjoint $STS(v)$ based on the $v+1$ distinct v–subsets of Y. This is called an *overlarge set* of $STS(v)$.

We are now ready to describe Rosa's construction. Let $N = \{1, 2, \ldots, v+1\}$ and $X = \{x_1, x_2, \ldots, x_v\}$, with X and N disjoint. Let $\mathcal{F} = \{F_1, F_2, \ldots, F_v\}$ be a good one–factorization of K_N. Let $\mathcal{H} = \{H_1, H_2, \ldots, H_v\}$ be any one–factorization of $K_{X \cup \{\infty\}}$, such that the edge $[x_j, \infty]$ belongs to H_j. Let $N^* = \{a_{ij}\}$ be any Latin square of order $v+1$ associated with the good one–factorization $\mathcal{F}$. Let $L = \{b_{ij}\}$ be any Latin square of order v, based on the set X, having the last two rows as follows.

$$\begin{array}{cccccccccc} x_3 & x_1 & x_2 & x_4 & x_5 & x_6 & x_7 & \cdots & x_{v-1} & x_v \\ x_2 & x_3 & x_1 & x_5 & x_4 & x_7 & x_6 & \cdots & x_v & x_{v-1} \end{array}$$

Let $(N, \mathcal{B})$ be any Steiner quadruple system of order $v+1$ and let

$$OS = \{(N \setminus \{y\}, \mathcal{B}_y) : y \in N\}$$

be the overlarge set of Steiner triple systems of order v which is derived from the Steiner quadruple system $(N, \mathcal{B})$ of order $v+1$. For each $i \in N$, let $\mathcal{H}^i = \{H_1^i, H_2^i, \ldots, H_v^i\}$ be the one–factorization of $K_{X \cup \{i\}}$ obtained from $\mathcal{H}$ by replacing the element ∞ with the element i in every one–factor, H_j, for $j = 1, 2, \ldots, v$. Finally, let $S = X \cup N$, so that $|S| = 2v+1$. Define the following collections of triples on S:

(i) For $i \in N$ and $j = 1, 2, \ldots, v$, let

$$C_{ij} = \{\{u, w, a_{ij}\} : [u, w] \in H_j^i\} \text{ and } \mathcal{C}_i = \bigcup_{j=1}^{v} C_{ij}.$$

Then $(S, (\mathcal{B}_i \cup \mathcal{C}_i))$ is a $STS(2v+1)$ for $i \in N$ and these $v+1$ $STS(2v+1)$ are pairwise disjoint.

(ii) For $i = 1, 2, \ldots, v-2$ and $j = 1, 2, \ldots, v+1$, let

$$E_{ij} = \{\{u, w, b_{ij}\} : [u, w] \in F_j\} \text{ and } \mathcal{E}_i = \bigcup_{j=1}^{v} E_{ij}.$$

Then $(S, (\mathcal{D}_i \cup \mathcal{E}_i))$ is a $STS(2v+1)$ for $i = 1, 2, \ldots, v-2$ and these $v-2$ $STS(2v+1)$ are pairwise disjoint.

Then a $LS\,(STS\,(2v+1))$ based on S can be constructed by taking the following collections of $2v-1$ disjoint $STS\,(2v+1)$ based on S:

$$\begin{array}{ll} (S\,,\,(\mathcal{B}_i \cup \mathcal{C}_i)) & \text{for } i \in N\,; \\ (S\,,\,(\mathcal{D}_i \cup \mathcal{E}_i)) & \text{for } i = 1\,,\,2\,,\,\ldots\,,\,v-2\,. \end{array}$$

Counting arguments which show that each triple of elements of S occurs precisely once in this large set of Steiner triple systems can be found in [17].

4 WHAT ELSE DO WE NEED?

We work by recursion on v, modulo 72.

Teirlinck's construction deals with the cases $v \equiv 3, 9, 21, 27, 39, 45, 57, 63$. Rosa's construction deals with the cases $v \equiv 7, 19, 31, 43, 55, 67$ by one argument, and with the cases $v \equiv 15, 51$ by another. The remaining cases must be dealt with by several constructions, due to Lu and Teirlinck. The cases are:

(i) $v \equiv 33, 69$;

(ii) $v \equiv 1, 25, 37, 61$;

(iii) $v \equiv 49$; and

(iv) $v \equiv 13$. This case subdivides, depending on which (if any) of the primes 5, 7 and 11 divides v.

To state the lemmas needed for this proof, we must first define orthogonality of Latin squares. Let L_1 and L_2 be two Latin squares of order n. Construct a new $n \times n$ array, M, of ordered pairs, where the ordered pair (u, v) occurs in the (i, j) position if and only if u occurs in the (i, j) position of L_1 and v occurs in the (i, j) position of L_2. If this new array contains all possible ordered pairs of elements from the underlying set, then L_1 and L_2 are *orthogonal*. A Latin square is *self–orthogonal* if it is orthogonal to its transpose. For example, Table 11 shows a pair of orthogonal Latin squares of order 3, and a self–orthogonal Latin square, L, of order 4, together with its transpose L^T. Further, self–orthogonal Latin squares of orders 10 and 14 respectively are shown in Table 12.

A set of *mutually orthogonal Latin squares of order n*, denoted by $MOLS\,(n)$, is a set of Latin squares of order n such that each pair is orthogonal. Let $N\,(n)$

1	2	3	1	3	2	1	3	4	2	1	4	2	3
3	1	2	3	2	1	4	2	1	3	3	2	4	1
2	3	1	2	1	3	2	4	3	1	4	1	3	2
						3	1	2	4	2	3	1	4
L_1			L_2			L				L^T			

Table 11 A pair of orthogonal Latin squares of order 3 and a self–orthogonal Latin square of order 4 together with its transpose.

0	3	9	7	8	1	2	5	6	4
1	6	0	9	2	3	5	4	7	8
2	5	1	6	9	4	0	7	8	3
3	4	7	5	1	9	8	6	2	0
4	0	8	2	7	5	9	3	1	6
5	8	6	3	4	2	7	9	0	1
6	7	3	1	0	8	4	2	9	5
9	1	2	0	5	6	3	8	4	7
8	9	5	4	6	7	1	0	3	2
7	2	4	8	3	0	6	1	5	9

1	9	4	C	0	3	6	A	7	B	2	5	D	8
D	2	0	5	1	A	4	7	B	8	C	3	6	9
7	D	3	A	6	2	B	5	8	C	9	1	4	0
5	8	D	4	B	7	3	C	6	9	1	0	2	A
3	6	9	D	5	C	8	4	1	7	0	2	A	B
B	4	7	0	D	6	1	9	5	2	8	A	3	C
4	C	5	8	A	D	7	2	0	6	3	9	B	1
C	5	1	6	9	B	D	8	3	A	7	4	0	2
A	1	6	2	7	0	C	D	9	4	B	8	5	3
6	B	2	7	3	8	A	1	D	0	5	C	9	4
0	7	C	3	8	4	9	B	2	D	A	6	1	5
2	A	8	1	4	9	5	0	C	3	D	B	7	6
8	3	B	9	2	5	0	6	A	1	4	D	C	7
9	0	A	B	C	1	2	3	4	5	6	7	8	D

Table 12 Self–orthogonal Latin squares of orders 10 and 14 respectively

denote the largest possible number of $MOLS(n)$. For convenience, we write $N(1) = \infty$. For $n > 1$, $N(n) \leq n - 1$, and a set of $n - 1$ $MOLS(n)$ is said to be a *complete set* of $MOLS(n)$. Note that $N(2) = N(6) = 1$; see for instance Stinson [19].

If $n = p^k$ for some prime p then a standard construction gives a complete set of $MOLS(n)$; see for instance [20]. For example, a complete set of $MOLS(5)$ constructed in this way from the field of order 5, $GF[5]$, is shown in Table 13.

0 1 2 3 4	0 1 2 3 4	0 1 2 3 4	0 1 2 3 4
1 2 3 4 0	2 3 4 0 1	3 4 0 1 2	4 0 1 2 3
2 3 4 0 1	4 0 1 2 3	1 2 3 4 0	3 4 0 1 2
3 4 0 1 2	1 2 3 4 0	4 0 1 2 3	2 3 4 0 1
4 0 1 2 3	3 4 0 1 2	2 3 4 0 1	1 2 3 4 0

Table 13 A complete set of $MOLS(5)$

It is easily shown that $N(mn) \geq \min\{N(m), N(n)\}$, for $m, n > 1$. We apply to this the construction based on $GF[p^k]$. Let $n = p_1^{e_1} p_2^{e_2} \ldots p_h^{e_h}$ be the factorization of n into prime powers where $p_1, p_2, \ldots, p_h$ are distinct primes and $e_1, e_2, \ldots, e_h \geq 1$. Then $N(n) \geq \min\{p_1^{e_1} - 1, p_2^{e_2} - 1, \ldots, p_h^{e_h} - 1\}$.

We now state the theorem, and the lemmas which finish the proof.

Theorem 4.1 (Lu [13]) *If $v \equiv 1$ or $3 \bmod 6$ and $v > 7$, then there exists a large set of Steiner triple systems of order v, $LS(STS(v))$.* □

Lemma 4.1 (Teirlinck [21]) *If there exists a $LS(STS(v))$ with $v > 2$, then there exists a $LS(STS(3v))$.* □

Lemma 4.2 (Rosa [17]) *If there exists a $LS(STS(v))$ with $v > 7$, then there exists a $LS(STS(2v + 1))$.* □

Lemma 4.3 (Lu [10], [12]) *If there exists a $LS(STS(n + 2))$, with $n \equiv 11$ mod12, then there exists a $LS(STS(3n))$.* □

Lemma 4.4 (Lu [9]) *Let n be an odd integer. If there exists 12 $MOLS(n)$ and a $LS(STS(2n + 1))$, then there exists a $LS(STS(12n + 1))$.* □

Lemma 4.5 (Lu [11]) *Let n be an integer. If there exists a $LS(STS(4n+1))$, then each of the following exists:*

(i) *a $LS(STS(12n+1))$;*

(ii) *a $LS(STS(24n+1))$; and*

(iii) *a $LS(STS(60n+1))$.* □

Lemma 4.6 (Lu [11]) *If there exists a $LS(STS(12n+1))$, where n is an odd integer, then each of the following exists:*

(i) *a $LS(STS(84n+1))$; and*

(ii) *a $LS(STS(132n+1))$.* □

5 PROVING THE THEOREM

Once the lemmata of Section 4 are proved, then the proof is by induction on the order of the Steiner triple system working modulo 72. There are seven cases to be considered.

(i) $v \equiv v' \bmod 72$ where $v' \in \{3, 9, 21, 27, 39, 45, 57, 63\}$. Let $v = 18t+3$ or $v = 18t+9$. By Lemma 4.1, there exists a $LS(STS(v))$ if there exists a $LS(STS(v/3))$, that is, a $LS(STS(6t+1))$ or $LS(STS(6t+3))$. By the induction hypothesis this holds for all $v = 18t+3$, and for all $v = 18t+9$ except for $v = 21$, since Cayley [2] showed that there does not exist a $LS(STS(7))$. A construction for a $LS(STS(21))$ was found by Denniston [3], using a starter $STS(21)$, $\mathcal{D}_0$, based on the set $\{0, 1, \ldots, 20\}$, $\mathcal{D}_i = \pi^i \mathcal{D}_0$, for $i = 0, 1, \ldots, 18$, where $\pi = (0, 1, ..., 18)$. $\mathcal{D}_0$ is listed in [18] Table A.2.

(ii) $v \equiv v' \bmod 72$ where $v' \in \{15, 51\}$. Let $v = 36t+15$. By Lemma 4.2, there exists a $LS(STS(v))$ if there exists a $LS(STS((v-1)/2))$, that is, a $LS(STS(18t+7))$. By the induction hypothesis this holds for all $v = 36t+15$ except for $v = 15$, since Cayley [2] showed that there does not exist a $LS(STS(7))$. A construction for a $LS(STS(15))$ was found by Denniston [3], using a starter $STS(15)$, $\mathcal{D}_0$, based on the set $\{0, 1, \ldots, 14\}$, $\mathcal{D}_i = \pi^i \mathcal{D}_0$, for $i = 0, 1, \ldots, 12$, where $\pi = (0, 1, ..., 12)$. $\mathcal{D}_0$ is listed in [18], Table A.1.

(iii) $v \equiv v' \bmod 72$ where $v' \in \{33, 69\}$. Let $v = 36t + 33$. By Lemma 4.3, if $(v/3) \equiv 11 \bmod 12$ and if there exists a $LS\,(STS\,((v/3) + 2))$, that is, a $LS\,(STS\,(12t + 13))$, then there exists a $LS\,(STS\,(v))$. By the induction hypothesis this holds for all $v = 36t + 33$.

(iv) $v \equiv v' \bmod 72$ where $v' \in \{7, 19, 31, 43, 55, 67\}$. Let $v = 12t + 7$. If there exists a $LS\,(STS\,((v-1)/2))$, that is, a $LS\,(STS\,(6t + 3))$, then by Lemma 4.2, there exists a $LS\,(STS\,(v))$. By the induction hypothesis this holds for all $v = 12t + 7$.

(v) $v \equiv v' \bmod 72$ where $v' \in \{1, 25, 37, 61\}$. Let $v = 36t+1$ or $v = 36t+25$. By case (i) of Lemma 4.5, with $n = 3t$ or $n = 3t+2$, there exists a $LS\,(STS\,(v))$ if there exists a $LS\,(STS\,((v+2)/3))$, that is, a $LS\,(STS\,(4n + 1))$. By the induction hypothesis this holds for all $v = 36t + 1$ or $v = 36t + 25$.

(vi) $v \equiv 49 \bmod 72$. Let $v = 72t+49$. By case (ii) of Lemma 4.5, with $n = 3t+2$, there exists a $LS\,(STS\,(v))$ if there exists a $LS\,(STS\,((v-1)/6 + 1))$, that is, a $LS\,(STS\,(4n + 1))$. By the induction hypothesis this holds for all $v = 72t + 49$.

(vii) $v \equiv 13 \bmod 72$. Let $v = 72t + 13$. There are four cases to be considered.

(a) $6t + 1$ is not divisible by 5, 7 or 11. The smallest prime dividing $6t + 1$ is 13 so there exist (at least) 12 $MOLS\,(6t + 1)$. By Lemma 4.4, with $n = 6t + 1$, there exists a $LS\,(STS\,(v))$ if there exists a $LS\,(STS\,((v-1)/6 + 1))$, that is, a $LS\,(STS\,(2n + 1))$. By the induction hypothesis this holds for all $v = 72t + 13$ where none of $5, 7, 11$ divides $6t + 1$.

(b) $6t+1$ is divisible by 5. By case (iii) of Lemma 4.5, with $n = (6t+1)/5$, there exists a $LS\,(STS\,(v))$ if there exists a $LS\,(STS\,((v-1)/15 + 1))$, that is, a $LS\,(STS\,(4n + 1))$. By the induction hypothesis this holds for all $v = 72t + 13$.

(c) $6t+1$ is divisible by 7. By case (i) of Lemma 4.6, with $n = (6t+1)/7$, there exists a $LS\,(STS\,(v))$ if there exists a $LS\,(STS\,((v-1)/7 + 1))$, that is, a $LS\,(STS\,(12n + 1))$. By the induction hypothesis this holds for all $v = 72t + 13$.

(d) $6t + 1$ is divisible by 11. By case (ii) of Lemma 4.6, with $n = (6t + 1)/11$, if there exists a $LS\,(STS\,((v-1)/11 + 1))$, that is, a $LS\,(STS\,(12n + 1))$, then there exists a $LS\,(STS\,(v))$. By the induction hypothesis this holds for all $v = 72t + 13$. □

6 AUXILIARY DESIGNS AND LARGE SETS OF STEINER TRIPLE SYSTEMS

This section is an account of the special constructions which are needed to fill in the missing cases for the results presented in Section 4. All of the constructions are due to Lu [9], [10] and [11], except for those of Teirlinck [22] in Lemma 6.2 and Lemma 6.3.

6.1 Auxiliary Designs of Type 1

To clarify our first definition, we need the idea of a pairwise balanced design. Let v and λ be positive integers and K be a set of positive integers. A design $(X, \mathcal{B})$ is a *pairwise balanced design*, $B[K, \lambda; v]$, if: $|X| = v$ (the design is of order v); $\{|B_i| : B_i \in \mathcal{B}\} \subseteq K$ (the block sizes are from K); every 2–set $\{x, y\} \subseteq X$ is contained in exactly λ blocks of $\mathcal{B}$. For instance, if $X_6 = \{0, 1, 2, 3, 4, 5\}$ and $\mathcal{B}_6 = \{01, 02, 03, 04, 05, 12345\}$, then $(X_6, \mathcal{B}_6)$ is a $B[\{2, 5\}, 1; 6]$.

Note that a pairwise balanced design $B[K, \lambda; v]$, where $K = \{3\}$ and $\lambda = 1$, is a $STS(v)$.

Definition 6.1 *Let* $I_{12} = \{0, 1, \cdots, 9, A, B\}$. *Let* $\widetilde{\mathcal{B}}$ *be a set of ordered pairs and ordered triples of elements of* I_{12} *and let* ρ^1 *and* ρ^2 *be sets of ordered pairs of elements of* I_{12} *which satisfy the following conditions:*

(i) *Every unordered pair of elements of* I_{12} *occurs in precisely one block of* $\widetilde{\mathcal{B}}$;

(ii) ρ^1 *and* ρ^2 *are disjoint;*

(iii) *If* $\rho = \rho^1 \cup \rho^2$, *then*

  (a) $(i_1, i_2) \in \rho$ *implies* $(i_2, i_1) \notin \rho$ *and* $\{i_1, i_2\} \in \widetilde{\mathcal{B}}$ *; and*

  (b) $\{i_1, i_2\} \in \widetilde{\mathcal{B}}$ *implies either* $(i_1, i_2) \in \rho$ *or* $(i_2, i_1) \in \rho$ *; and*

(iv) *For any* $i_1 \in I_{12}$, *there exists a unique* $i_2 \in I_{12}$, *such that either* $(i_1, i_2) \in \rho$ *or* $(i_2, i_1) \in \rho^1$.

Then the triple $\left(\widetilde{\mathcal{B}}, \rho^1, \rho^2\right)$ *forms an* auxiliary design of type 1, *and is denoted by* LJX_1. *(Note that when the order of elements within the blocks of* $\widetilde{\mathcal{B}}$ *is ignored, then* $\left(I_{12}, \widetilde{\mathcal{B}}\right)$ *is a pairwise balanced design* $B[\{2,3\},1;12]$.*)*

In the next constructions, we use a structure closely related to orthogonal Latin squares. Let k, n and λ be positive integers. A design $(X, \mathcal{G}, \mathcal{A})$, where X is a finite set of points, $\mathcal{G}$ is a parallel class of subsets of X called *groups* and $\mathcal{A}$ is a family of subsets of X called *blocks*, is a *transversal design* $T[k, \lambda; n]$ if : $|X| = kn$; $|G_i| = n$ for every $G_i \in \mathcal{G}$ and hence $|\mathcal{G}| = k$; $|B_j| = k$ for every $B_j \in \mathcal{A}$; $|G_i \cap B_j| = 1$ for every $G_i \in \mathcal{G}$ and every $B_j \in \mathcal{A}$; every 2–set $\{x, y\} \subseteq X$, such that x and y belong to distinct groups, is contained in exactly λ blocks of $\mathcal{A}$.

For example, if $X = \{0,1,2,3,4,5,6,7,8,9,\mathtt{A},\mathtt{B}\}$, $\mathcal{G} = \{0123, 4567, 89\mathtt{AB}\}$ and

$$\mathcal{A} = \left\{ \begin{array}{cccc} 048 & 059 & 06\mathtt{A} & 07\mathtt{B} \\ 149 & 15\mathtt{A} & 16\mathtt{B} & 178 \\ 24\mathtt{A} & 25\mathtt{B} & 268 & 279 \\ 34\mathtt{B} & 358 & 369 & 37\mathtt{A} \end{array} \right\},$$

then $(X, \mathcal{G}, \mathcal{A})$ is a transversal design $T[3, 1; 4]$.

A transversal design $T[k, \lambda; n]$ has λn^2 blocks, and if $k \leq n$ and $2 \leq n$, then any of its blocks is disjoint from at least one other block.

Let $T(k, \lambda)$ be the set of values of n for which a transversal design $T[k, \lambda; n]$ exists. If s, m and t are positive integers such that $0 \leq s \leq t$, $1 \leq m$, $t \in T(k+1, 1)$ and s, m, $m+1 \in T(k, 1)$, then $mt + s \in T(k, 1)$.

The existence of a set of $k-2$ $MOLS(n)$ is equivalent to the existence of a transversal design $T[k, 1; n]$. For example, let

$$X = \{x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}, x_{41}, x_{42}, x_{43}\},$$

$$\mathcal{G} = \{G_1, G_2, G_3, G_4\},$$

where

$$G_1 = \{x_{11}, x_{12}, x_{13}\}, \qquad G_2 = \{x_{21}, x_{22}, x_{23}\},$$
$$G_3 = \{x_{31}, x_{32}, x_{33}\}, \qquad G_4 = \{x_{41}, x_{42}, x_{43}\}$$

and

$$\mathcal{A} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9\},$$

where

$$A_1 = \{x_{11}, x_{21}, x_{31}, x_{41}\}, \quad A_2 = \{x_{11}, x_{22}, x_{32}, x_{42}\},$$
$$A_3 = \{x_{11}, x_{23}, x_{33}, x_{43}\}, \quad A_4 = \{x_{12}, x_{21}, x_{32}, x_{43}\},$$
$$A_5 = \{x_{12}, x_{22}, x_{33}, x_{41}\}, \quad A_6 = \{x_{12}, x_{23}, x_{31}, x_{42}\},$$
$$A_7 = \{x_{13}, x_{31}, x_{33}, x_{42}\}, \quad A_8 = \{x_{13}, x_{32}, x_{31}, x_{43}\},$$
$$A_9 = \{x_{13}, x_{33}, x_{32}, x_{41}\}.$$

Then $(X, \mathcal{G}, \mathcal{A})$ is a transversal design $T[4, 1; 3]$. Two orthogonal Latin squares, L_1 and L_2, of order 3 can be constructed from this transversal design. Let G_3 and G_4 give the two coordinates and let G_1 and G_2 be associated with the latin squares L_1 and L_2, respectively. To find the elements in position $(1, 2)$ of each square consider the elements x_{31} and x_{42} of the design. These occur together in block A_6 with x_{12} and x_{23}. Hence the $(1, 2)$ position of L_1 is 2 and of L_2 is 3. The complete construction is shown in Table 14. The pair of orthogonal Latin squares, L_1 and L_2, of order 3 which result are those of Table 11.

Position	Coordinate elements	Block containing coordinate elements	L_1	L_2
$(1,1)$	x_{31} , x_{41}	$A_1 = \{x_{11}, x_{21}, x_{31}, x_{41}\}$	1	1
$(1,2)$	x_{31} , x_{42}	$A_6 = \{x_{12}, x_{23}, x_{31}, x_{42}\}$	2	3
$(1,3)$	x_{31} , x_{43}	$A_8 = \{x_{13}, x_{32}, x_{31}, x_{43}\}$	3	2
$(2,1)$	x_{32} , x_{41}	$A_9 = \{x_{13}, x_{33}, x_{32}, x_{41}\}$	3	3
$(2,2)$	x_{32} , x_{42}	$A_2 = \{x_{11}, x_{22}, x_{32}, x_{42}\}$	1	2
$(2,3)$	x_{32} , x_{43}	$A_4 = \{x_{12}, x_{21}, x_{32}, x_{43}\}$	2	1
$(3,1)$	x_{33} , x_{41}	$A_5 = \{x_{12}, x_{22}, x_{33}, x_{41}\}$	2	2
$(3,2)$	x_{33} , x_{42}	$A_7 = \{x_{13}, x_{31}, x_{33}, x_{42}\}$	3	1
$(3,3)$	x_{33} , x_{43}	$A_3 = \{x_{11}, x_{23}, x_{33}, x_{43}\}$	1	3

Table 14 Constructing two $MOLS(3)$ from the $T[4, 1; 3]$ given above

If $0 \leq s \leq t$ and $1 \leq m$ then

$$N(mt + s) \geq \min\{N(m), N(m+1), N(t) - 1, N(s)\}.$$

Although $N(6) = 1$, this recursive construction can be used to show that for $n > 6$, $n \equiv 2 \bmod 4$, then $N(n) \geq 2$.

Lemma 6.1 *Let LJX_1 be an auxiliary design of type* 1. *Let $|\rho^1| = m_1$ and $|\rho^2| = m_2$. Then $2m_1 + m_2 = 12$.*

Proof. This follows from property (iv) of an auxiliary design of type 1. □

Construction 6.2 (Lu [10, Lemma 1]) Let $I_{12} = \{0, 1, \cdots, 9, A, B\}$. Let LJX_1 be an auxiliary design of type 1 with $\rho^2 \neq \emptyset$. Let n be a positive integer such that a transversal design $T[14, 1; n]$ exists. Let $I_n = \{0, 1, \cdots, n-1\}$ and $I_{14} = \{0, 1, \cdots, 9, A, B, C, D\}$. Let $(I_{14} \times I_n, \mathcal{G}, \mathcal{A})$ be a transversal design $T[14, 1; n]$, where $\mathcal{G} = \{\{i\} \times I_n : i \in I_{14}\}$. Let $S' = \{\infty\} \cup (I_2 \times I_n)$ and let $(S', \mathcal{B}')$ be a $STS(2n+1)$. Let $S'' = \{\infty\} \cup I_{12}$ and let $(S'', \mathcal{B})$ be a $STS(13)$ with the property that $(i_1, i_2) \in \rho^1$ $\{\infty, i_1, i_2\} \in \mathcal{B}$. Finally, let $S = \{\infty\} \cup (I_{12} \times I_n)$. Then, for any $s \in I_n$, the system of triples given in Table 29 at the end of this chapter form a $STS(12n+1)$ based on the set S. The Steiner triple system defined above will be denoted by $(S, \langle LJX_1, \mathcal{B}, \mathcal{B}' \rangle_s)$. Counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [10], Lemma 1.

Construction 6.3 (Lu [10, Lemma 2]) Let $I_{12} = \{0, 1, \cdots, 9, A, B\}$. Let LJX_1 be an auxiliary design of type 1 with $\rho^2 = \emptyset$. Let n be a positive integer such that a transversal design $T[14, 1; n]$ exists. Let $I_n = \{0, 1, \cdots, n-1\}$, and $I_{14} = \{0, 1, \cdots, 9, A, B, C, D\}$. Let $(I_{14} \times I_n, \mathcal{G}, \mathcal{A})$ be a transversal design $T[14, 1; n]$, where $\mathcal{G} = \{\{i\} \times I_n : i \in I_{14}\}$. Let $S' = \{\infty\} \cup (I_2 \times I_n)$ and let $(S', \mathcal{B}')$ be a $STS(2n+1)$. Finally, let $S = \{\infty\} \cup (I_{12} \times I_n)$. Then, for any $s \in I_n$, the system of triples given in Table 30 at the end of this chapter form a $STS(12n+1)$ based on the set S. The Steiner triple system defined above will be denoted by $(S, \langle LJX_1, \mathcal{B}' \rangle_s)$. Counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [10], Lemma 2.

6.2 The Proof of Lemma 4.4

The proof is by construction and is based on auxiliary designs of type 1. This construction uses specific examples given in Lu [9].

Example 6.4 A collection of auxiliary designs of type 1, $\{LJX_1^i : i \in I_{11}\}$ is shown in Table 15. This collection of auxiliary designs of type 1 has the following properties:

(i) Any unordered triple of I_{12} belongs to a unique $\widetilde{\mathcal{B}}_i$;

(ii) $\rho_i^1 \neq \emptyset$ if and only if $i \in \{0, 1, 11\}$;

(iii) $\rho_0^1 \cap \rho_1^1 = \emptyset$;

(iv) $\rho_0^1 \cup \rho_1^1 = \rho_{11}^1$;

(v) $\rho_i^2 \neq \emptyset$ for $i \in I_{11}$; and

(vi) For any ordered pair (i_1, i_2) of elements of I_{12}, with $i_1 \neq i_2$, if $(i_1, i_2) \notin \rho_{11}^1$ and $(i_2, i_1) \notin \rho_{11}^1$, then (i_1, i_2) belongs to a unique ρ_i^2 for $i \in I_{11}$.

To use Construction 6.2 with $\{ LJX_1^i : i \in I_{11} \}$, a collection of eleven $STS(13)$ are needed. Let $S'' = \{\infty\} \cup I_{12}$. A $LS(STS(13))$, LS, based on the set S'' is shown in Table 16. Note that, for each $i \in I_{11}$, $(i_1, i_2) \in \rho_i^1$ implies $\{\infty, i_1, i_2\} \in \mathcal{B}_i$.

Construction 6.5 (Lu [10, Theorem]) Let $I_{12} = \{0, 1, \cdots, 9, A, B\}$. Let $\{ LJX_1^i : i \in I_{11} \}$ be the collection of auxiliary designs of type 1 given in Example 6.4, where $LJX_1^i = \left(\widetilde{\mathcal{B}}_i, \rho_i^1, \rho_i^2\right)$. Let n be a positive integer such that a transversal design $T[14, 1; n]$ exists. Let $I_n = \{0, 1, \cdots, n-1\}$ and $I_{14} = \{0, 1, \cdots, 9, A, B, C, D\}$. Let $(I_{14} \times I_n, \mathcal{G}, \mathcal{A})$ be a transversal design $T[14, 1; n]$, where $\mathcal{G} = \{\{i\} \times I_n : i \in I_{14}\}$. Let $S' = \{\infty\} \cup (I_2 \times I_n)$ and let LS be a $LS(STS(2n+1))$, based on S', where

$$LS = \left\{ \left(S', \mathcal{B}_{is}{}'\right) : (i, s) \in (I_2 \times I_n) \setminus (0, 0) \right\}.$$

Let $S'' = \{\infty\} \cup I_{12}$ and let LS be the $LS(STS(13))$ given in Example 6.4. Finally, let $S = \{\infty\} \cup (I_{12} \times I_n)$. Then a $LS(STS(12n+1))$ based on S can be constructed by taking the following collections of disjoint $STS(12n+1)$ based on S: $\left(S, \left\langle LJX_1^i, \mathcal{B}_i, \mathcal{B}_{0s}{}' \right\rangle_s\right)$ for $(j, y) \in (I_{11} \times I_n)$ and $\left(S, \left\langle LJX_1^{11}, \mathcal{B}_{1s}{}' \right\rangle_s\right)$ for $s \in I_n \setminus \{0\}$. Counting arguments which show that each triple of elements of S occurs precisely once in this large set of Steiner triple systems can be found in [9].

6.3 Auxiliary Designs of Type 2

Let X be a finite set of n elements. An *orthogonal array*, $OA(t, k, n)$, is a subset $\mathcal{L}$ of X^k, such that for any t–element subset $\{i_1, i_2, \ldots, i_t\}$ of $\{1, 2, \ldots, k\}$

$\widetilde{\mathcal{B}_0}$	12 38 B0 58 A8 40 70 62 92 280 23B 18B 130 37A 369 345 B79 BA5 B64 1A9 147 165 867 849 059 06A 2A4 257
$\widetilde{\mathcal{B}_1}$	57 6A 49 BA 2A 39 09 17 87 7A9 764 5A4 569 62B 610 638 48B 430 412 581 532 5B0 A80 A13 928 9B1 720 73B
$\widetilde{\mathcal{B}_2}$	13 23 3B 8B B1 01 50 A0 42 72 68 98 128 380 B02 B7A B69 B45 179 1A5 164 3A9 347 365 067 049 259 26A 8A4 857
$\widetilde{\mathcal{B}_3}$	28 18 80 30 02 B2 51 A1 43 73 6B 9B 213 83B 0B1 27A 269 254 879 8A5 864 0A9 047 056 167 149 359 36A BA4 B57
$\widetilde{\mathcal{B}_4}$	1B 2B 31 81 B3 03 53 A3 4B 7B 61 91 120 382 B08 87A 869 845 079 0A5 064 2A9 247 265 367 349 B59 B6A 1A4 157
$\widetilde{\mathcal{B}_5}$	20 10 82 32 08 B8 5B AB 41 71 63 93 12B 381 B03 07A 069 045 279 2A5 264 8A9 847 865 B67 B49 159 16A 3A4 357
$\widetilde{\mathcal{B}_6}$	56 76 64 45 95 B9 29 37 07 A4 1A 8A 57A 6A9 497 42B 410 438 58B 530 512 681 632 6B0 980 913 728 7B1 A20 A3B
$\widetilde{\mathcal{B}_7}$	7A 5A A9 69 97 47 B5 25 36 06 14 84 756 A64 945 72B 710 738 A8B A30 A12 981 932 9B0 580 513 628 6B1 420 43B
$\widetilde{\mathcal{B}_8}$	54 74 65 A5 46 96 B6 26 34 04 15 85 579 6A7 49A A2B A10 A38 98B 930 912 781 732 7B0 680 613 428 4B1 520 53B
$\widetilde{\mathcal{B}_9}$	79 59 A7 67 9A 4A B4 24 35 05 16 86 754 A65 946 92B 910 938 78B 730 712 A81 A32 AB0 480 413 528 5B1 620 63B
$\widetilde{\mathcal{B}_{10}}$	B7 27 3A 0A 19 89 52 A2 48 78 60 90 180 230 28B 5A9 769 7A4 385 126 B04 137 B1A 3B9 541 643 65B 492 6A8 570
$\widetilde{\mathcal{B}_{11}}$	38 B0 12 6A 49 57 BA9 379 0A4 859 17A 267 3A5 B47 169 418 630 52B 423 920 510 A28 780 68B 13B 564

Table 15 A collection of 12 LJX_1

ρ_0^1	12 38 B0
ρ_0^2	58 A8 40 70 62 92
ρ_1^1	57 6A 49
ρ_1^2	BA 2A 39 09 17 87
ρ_{11}^1	38 B0 12 6A 49 57

ρ_2^2	13 23 3B 8B B1 01 50 A0 42 72 68 98
ρ_3^2	28 18 80 30 02 B2 51 A1 43 73 6B 9B
ρ_4^2	1B 2B 31 81 B3 03 53 A3 4B 7B 61 91
ρ_5^2	20 10 82 32 08 B8 5B AB 41 71 63 93
ρ_6^2	56 76 64 45 95 B9 29 37 07 A4 1A 8A
ρ_7^2	7A 5A A9 69 97 47 B5 25 36 06 14 84
ρ_8^2	54 74 65 A5 46 96 B6 26 34 04 15 85
ρ_9^2	79 59 A7 67 9A 4A B4 24 35 05 16 86
ρ_{10}^2	B7 27 3A 0A 19 89 52 A2 48 78 60 90

Table 15 (cont'd) A collection of 12 LJX_1

and for any t (not necessarily distinct) elements $x_1, x_2, \ldots, x_t$ of X, there is exactly one row of $\mathcal{L}$ containing $(y_1, y_2, \ldots, y_t)$ with $y_{i_j} = x_j$ for $j = 1, 2, \ldots, t$. A Latin square of order n is equivalent to an orthogonal array $OA(2, 3, n)$.

For example, an $OA(2, 4, 3)$ is shown in Table 17 and a Latin square of order 4 together with its associated orthogonal array is shown in Table 18.

Definition 6.6 *Let X be a set of n elements. Let $\mathcal{L}^1$ and $\mathcal{L}^2$ be orthogonal arrays $OA(2, 4, n)$ based on the set X and, for each $x \in X$, let $\mathcal{L}_x$ be an orthogonal array $OA(2, 3, n-1)$ based on the set $X \setminus \{x\}$ which satisfy the following conditions:*

(i) There exists a $c_0 \in X$, such that $(x, x, x, c_0) \in \mathcal{L}^1 \cap \mathcal{L}^2$ for all $x \in X$; and

(ii) Any $(x_1, x_2, x_3) \in X^3$ is contained in one of the following:

(a) $\mathcal{L}_x$, for some $x \in X$;

(b) $\overline{\mathcal{L}^1} = \{(x_1, x_2, x_3) : (x_1, x_2, x_3, x) \in \mathcal{L}^1$ for some $x \in X\}$; or

(c) $\overline{\mathcal{L}^2} = \{(x_1, x_2, x_3) : (x_1, x_2, x_3, x) \in \mathcal{L}^2$ for some $x \in X\}$.

$\mathcal{B}_0$	016 023 048 057 09A 13B 145 179 18A 24A 25B 267 289 347 35A 369 49B 568 6AB 78B ∞0B ∞12 ∞38 ∞46 ∞59 ∞7A
$\mathcal{B}_1$	01A 029 035 046 07B 127 134 159 168 24B 256 28A 36B 378 39A 458 47A 5AB 679 89B ∞08 ∞1B ∞23 ∞49 ∞57 ∞6A
$\mathcal{B}_2$	012 039 04A 058 06B 13A 146 157 18B 238 245 26A 279 35B 367 47B 489 569 78A 9AB ∞07 ∞19 ∞2B ∞34 ∞5A ∞68
$\mathcal{B}_3$	015 024 037 089 0AB 123 14A 169 17B 257 268 29B 349 356 38A 46B 478 58B 59A 67A ∞06 ∞18 ∞2A ∞3B ∞45 ∞79
$\mathcal{B}_4$	01B 025 049 06A 078 126 135 148 19A 234 27A 28B 368 379 3AB 45A 467 57B 589 69B ∞03 ∞17 ∞29 ∞4B ∞56 ∞8A
$\mathcal{B}_5$	017 02A 038 04B 056 12B 136 15A 189 237 246 259 345 39B 479 48A 578 68B 69A 7AB ∞09 ∞14 ∞28 ∞3A ∞5B ∞67
$\mathcal{B}_6$	013 026 059 07A 08B 128 149 15B 167 23B 247 29A 348 357 36A 456 4AB 58A 689 79B ∞04 ∞1A ∞25 ∞39 ∞6B ∞78
$\mathcal{B}_7$	018 03A 047 05B 069 124 137 16A 19B 239 25A 26B 278 34B 358 459 468 567 79A 8AB ∞02 ∞15 ∞36 ∞4A ∞7B ∞89
$\mathcal{B}_8$	014 027 036 08A 09B 129 158 16B 17A 235 248 2AB 34A 37B 389 45B 469 56A 579 678 ∞05 ∞13 ∞26 ∞47 ∞8B ∞9A
$\mathcal{B}_9$	019 028 03B 045 067 125 138 147 1AB 23A 269 27B 346 359 48B 49A 56B 57A 68A 789 ∞0A ∞16 ∞24 ∞37 ∞58 ∞9B
$\mathcal{B}_{10}$	02B 034 05A 068 079 12A 139 14B 156 178 236 249 258 37A 38B 457 46A 59B 67B 89A ∞01 ∞27 ∞35 ∞48 ∞69 ∞AB

Table 16 A $LS(STS(13))$ based on the set $\{\infty\} \cup I_{12}$

1 1 1 1	2 1 2 3	3 1 3 2
1 2 2 2	2 2 3 1	3 2 1 3
1 3 3 3	2 3 1 2	3 3 2 1

Table 17 An $OA(2,4,3)$

1 2 3 4 ‖ 1 1 1	2 1 2	3 1 3	4 1 4
2 3 4 1 ‖ 1 2 2	2 2 3	3 2 4	4 2 1
3 4 1 2 ‖ 1 3 3	2 3 4	3 3 1	4 3 2
4 1 2 3 ‖ 1 4 4	2 4 1	3 4 2	4 4 3

Table 18 A Latin square of order 4 and its associated $OA(2,3,4)$

Then the quadruple $\left(X, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x\in X}\right)$ *forms an* auxiliary design of type 2, *based on the set* X, *and is denoted by* $LJX_2(n)$. *An auxiliary design of type* 2 *which satisfies the additional condition that:*

(iii) The set $\mathcal{L}^0 = \left\{(x_0,x_1,x_2,x_3) : (x_0,x_2,x_3) \in \overline{\mathcal{L}^1} \text{ and } (x_1,x_2,x_3) \in \overline{\mathcal{L}^2}\right\}$ *is an orthogonal array* $OA(2,4,n)$

is denoted by $LJX_2^*(n)$. *An auxiliary design of type* 2 *with condition (i) replaced by:*

(i') $(x,x,x,x) \in \mathcal{L}^1 \cap \mathcal{L}^2$ *for all* $x \in X$

is denoted by $LJX_2'(n)$. *(Note that if* $\left(X, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x\in X}\right)$ *is a* $LJX_2^*(n)$, *then* $\left(X, \mathcal{L}^{1'}, \mathcal{L}^{2'}, (\mathcal{L}_x)_{x\in X}\right)$, *is a* $LJX_2'(n)$, *where*

$$\mathcal{L}^{1'} = \left\{(x_0,x_1,x_2,x_3) : (x_0,x_1,x_2) \in \overline{\mathcal{L}^1} \text{ and } (x_3,x_1,x_2) \in \overline{\mathcal{L}^2}\right\}$$

and

$$\mathcal{L}^{2'} = \left\{(x_0,x_1,x_2,x_3) : (x_0,x_1,x_2) \in \overline{\mathcal{L}^2} \text{ and } (x_3,x_1,x_2) \in \overline{\mathcal{L}^1}\right\}.)$$

Example 6.7 A $LJX_2^*(4)$, a $LJX_2^*(5)$ and a $LJX_2^*(7)$ are shown in Tables 19, 20 and 21 respectively.

Lemma 6.2 (Teirlinck [22, Proposition 2]) *Let* $(X,\mathcal{B})$ *be a pairwise balanced design* $B[K,1;n]$ *with no blocks of size* 0, 1 *or* 2. *Let* $c_0 \in X$. *If, for each* $B \in \mathcal{B}$ *with* $c_0 \in B$, *there exists a* $LJX_2(|B|)$ *and, for each* $B \in \mathcal{B}$ *with* $c_0 \notin B$, *there exists a* $LJX_2'(|B|)$, *then there exists a* $LJX_2(n)$.

0000	2031	0000	2011	113	002	003	001
0133	2102	0123	2132	122	020	011	010
0211	2220	0231	2220	131	033	030	022
0322	2313	0312	2303	212	200	101	100
1023	3012	1033	3022	221	223	110	112
1110	3121	1110	3101	233	232	133	121
1232	3203	1202	3213	311	303	300	202
1301	3330	1321	3330	323	322	313	211
				332	330	331	220
$\mathcal{L}^1$		$\mathcal{L}^2$		$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$

Table 19 A $LJX_2^*(4)$

0000	2344	0000	2303	114	002	001	004	003
0124	2413	0133	2431	122	023	014	010	011
0243	3023	0211	3041	131	030	033	022	020
0312	3142	0344	3124	143	044	040	041	032
0431	3211	0422	3202	213	204	100	101	102
1041	3330	1032	3330	221	220	113	112	110
1110	3404	1110	3413	232	233	134	120	121
1234	4014	1243	4023	244	242	141	144	133
1303	4133	1321	4101	311	300	303	202	200
1422	4202	1404	4234	323	324	310	211	212
2032	4321	2014	4312	334	332	331	224	223
2101	4440	2142	4440	342	343	344	240	231
2220		2220		412	403	404	400	301
				424	422	411	414	313
				433	434	430	421	322
				441	440	443	442	330
$\mathcal{L}^1$		$\mathcal{L}^2$		$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$

Table 20 A $LJX_2^*(5)$

$\mathcal{L}^1$	$\mathcal{L}^2$
0000	0000
0135	0153
0263	0236
0321	0312
0456	0465
0514	0541
0642	0624
1052	1034
1110	1110
1245	1263
1303	1346
1431	1422
1566	1505
1624	1651
2034	2061
2162	2144
2220	2220
2355	2303
2413	2456
2541	2532
2606	2615
3016	3025
3144	3101
3202	3254
3330	3330
3465	3413
3523	3566
3651	3642
4061	4052
4126	4135
4254	4211
4312	4364
4440	4440
4505	4523
4633	4606
5043	5016
5101	5162
5236	5245
5364	5321
5422	5404
5550	5550
5615	5633
6025	6043
6153	6126
6211	6202
6346	6355
6404	6431
6532	6514
6660	6660

$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$	$\mathcal{L}_6$
112	005	003	004	006	002	001
121	024	014	016	012	011	010
136	036	030	022	021	020	025
144	042	041	040	035	034	033
155	050	056	055	053	043	044
163	063	065	061	060	066	052
211	204	104	106	102	101	100
224	223	116	114	115	110	113
233	232	135	125	120	123	122
242	240	140	141	133	132	131
256	255	153	152	151	146	145
265	266	161	160	166	164	154
316	306	300	202	201	200	205
323	322	315	215	210	213	212
332	335	334	220	225	226	221
345	344	343	246	236	231	234
354	353	351	251	252	244	243
361	360	366	264	263	262	250
414	402	401	400	305	304	303
422	420	410	411	313	312	311
435	434	433	426	326	321	324
441	443	446	445	331	336	330
453	456	455	454	350	340	342
466	465	464	462	362	363	355
515	500	506	505	503	403	404
526	525	513	512	511	416	415
534	533	531	521	522	424	423
543	546	545	544	530	430	432
551	552	554	550	556	442	440
562	564	560	566	565	461	451
613	603	605	601	600	606	502
625	626	611	610	616	614	514
631	630	636	624	623	622	520
646	645	644	642	632	633	535
652	654	650	656	655	641	541
664	662	663	665	661	660	553

Table 21 A $LJX_2^*(7)$

Proof. For each $B \in \mathcal{B}$ with $c_0 \in B$, let $\left(B, \mathcal{L}^1_B, \mathcal{L}^2_B, (\mathcal{L}_{Bx})_{x \in X}\right)$ be a $LJX_2(|B|)$ with $(x,x,x,c_0) \in \mathcal{L}^1_B \cap \mathcal{L}^2_B$ for all $x \in B$. For each $B \in \mathcal{B}$ with $c_0 \notin B$, let $\left(B, \mathcal{L}^1_{B'}, \mathcal{L}^2_{B'}, (\mathcal{L}_{Bx})_{x \in B}\right)$ be a $LJX_2(|B|)$. Let

$$\mathcal{L}^1_B = \mathcal{L}^1_{B'} \setminus \{(x,x,x,x) : x \in B\} \text{ and } \mathcal{L}^2_B = \mathcal{L}^2_{B'} \setminus \{(x,x,x,x) : x \in B\}.$$

Since there are no blocks of size 0, 1 or 2 in $(X, \mathcal{B})$ and $\lambda = 1$, identifying two elements, x and y, identifies a block, B, and hence an idempotent quasigroup $(B, \circ_{xy})$. Let

$$\mathcal{M} = \{(x_1,x_2,x_3) : x_1,x_2,x_3 \in X \setminus \{x\} \; ; \; \{x_1,x_2,x_3\} \not\subseteq B \in \mathcal{B} \text{ and } x_1 \circ_{x_1x_2} x_2 = x_3 \circ_{x_3x} x\}.$$

Then $\left(X, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x \in X}\right)$ is a $LJX_2(n)$, where $\mathcal{L}^1 = \bigcup_{B \in \mathcal{B}} \mathcal{L}^1_B$, $\mathcal{L}^2 = \bigcup_{B \in \mathcal{B}} \mathcal{L}^2_B$ and $\mathcal{L}_x = \left(\bigcup_{x \in B \in \mathcal{B}} \mathcal{L}_{Bx}\right) \cup \mathcal{M}$. □

We have seen that a pairwise balanced design with all blocks of size 3 and $\lambda = 1$ is in fact a Steiner triple system. More generally, a $B[\{K\}, \lambda; n]$ with $K = \{k\}$ is a *balanced incomplete block design* $B[k, \lambda; n]$. Such a design is said to be *resolvable* if its set of blocks can be decomposed into a family of parallel classes, where each parallel class covers each element of the design precisely once.

Lemma 6.3 (Teirlinck [22]) *A $LJX_2(n)$ exists for all $n \equiv 11 \bmod 12$, $n \geq 35$.*

Proof. Hanani, Ray–Chaudhuri and Wilson [6] have shown that a resolvable $SQS(v)$ exists for all $v \equiv 4 \bmod 12$. Hence there exists a resolvable $SQS(n-7)$, $(S, \mathcal{B})$, $n \geq 35$, with at least seven parallel classes. Let $\gamma_1, \gamma_2, \cdots, \gamma_7$ be seven parallel classes of $(S, \mathcal{B})$ and let

$$S' = S \cup \{\infty_1, \infty_2, \cdots, \infty_7\},$$

where $\infty_i \notin S$ for $i = 1, 2, \ldots, 7$. Then $\left(S', \mathcal{B}'\right)$ is a pairwise balanced design $B[\{4,5,7\}, 1; n]$, where

$$\begin{aligned} \mathcal{B}' = \; & \{B : B \in \mathcal{B} \text{ and } B \notin \gamma_i, i = 1, 2, \ldots, 7\} \cup \\ & \{B \cup \{\infty_i\} : B \in \gamma_i, i = 1, 2, \ldots, 7\} \cup \\ & \{\{\infty_1, \infty_2, \cdots, \infty_7\}\}, \end{aligned}$$

Hence, by Lemma 6.2, there exists a a $LJX_2(n)$ for all $n \equiv 11 \bmod 12$, $n \geq 35$. □

Construction 6.8 (Lu [10, Lemma 1]) Let n be a positive integer, $n \equiv 11 \bmod 12$. Let $I_n = \{0, 1, \cdots, n-1\}$ and let $\left(I_n, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x \in I_n}\right)$ be an auxiliary design of type 2, $LJX_2(n)$, based on I_n, such that condition (i) in the definition of an auxiliary design of type 2 is satisfied with $c_0 = 0$. Let $S' = \{\infty_1, \infty_2\} \cup I_n$ and let LS be a $LS(STS(n+2))$, based on S', where $LS = \{(S', \mathcal{B}_i) : i \in I_n\}$ and the triple $\{\infty_1, \infty_2, i\}$ belongs to $\mathcal{B}_i$. Finally, let $S = (GF[4] \setminus \{0\}) \times I_n$. Then, for any $i \in I_n$, the system of triples given in Table 31 at the end of this chapter form a $STS(3n)$ based on the set S. The Steiner triple system defined above will be denoted by $(S, \langle \mathcal{L}_i, \mathcal{B}_i \rangle_i)$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [10], Lemma 1.

Construction 6.9 (Lu [10, Lemma 2]) Let n be a positive integer, $n \equiv 11 \bmod 12$. Let $I_n = \{0, 1, \cdots, n-1\}$ and let $\left(I_n, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x \in I_n}\right)$ be an auxiliary design of type 2, $LJX_2(n)$, based on I_n, such that condition (i) in the definition of an auxiliary design of type 2 is satisfied with $c_0 = 0$. Let $S' = \{\infty_1, \infty_2\} \cup I_n$ and let LS be a $LS(STS(n+2))$, based on S', where $LS = \{(S', \mathcal{B}_i) : i \in I_n\}$ and the triple $\{\infty_1, \infty_2, i\}$ belongs to $\mathcal{B}_i$. Finally, let $S = (GF[4] \setminus \{0\}) \times I_n$. Then, for any $i \in I_n$, the system of triples given in Table 32 at the end of this chapter form a $STS(3n)$ based on the set S. The Steiner triple system defined above will be denoted by $(S, \langle \mathcal{L}^1, \mathcal{B}_i \rangle_i)$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [10], Lemma 2.

Construction 6.10 (Lu [10, Lemma 2]) Let n be a positive integer, $n \equiv 11 \bmod 12$. Let $I_n = \{0, 1, \cdots, n-1\}$ and let $\left(I_n, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x \in I_n}\right)$ be an auxiliary design of type 2, $LJX_2(n)$, based on I_n, such that condition (i) in the definition of an auxiliary design of type 2 is satisfied with $c_0 = 0$. Let $S' = \{\infty_1, \infty_2\} \cup I_n$ and let LS be a $LS(STS(n+2))$, based on S', where $LS = \{(S', \mathcal{B}_i) : i \in I_n\}$ and the triple $\{\infty_1, \infty_2, i\}$ belongs to $\mathcal{B}_i$. Finally, let $S = (GF[4] \setminus \{0\}) \times I_n$. Then, for any $i \in I_n$, the system of triples given in Table 33 at the end of this chapter form a $STS(3n)$ based on the set S. The Steiner triple system defined above will be denoted by $(S, \langle \mathcal{L}^2, \mathcal{B}_i \rangle_i)$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in Lu [10], Lemma 2.

6.4 The Proof of Lemma 4.3

The proof is by construction and is based on auxiliary designs of type 2. This construction uses specific examples given in Lu [10].

Theorem 6.11 (Lu [10, Theorem]) *If there exist both a* $LJX_2(n)$ *and a* $LS(STS(n+2))$*, then there exists a* $LS(STS(3n))$.

Construction 6.12 (Lu [10]) Let n be a positive integer, $n \equiv 11 \bmod 12$. Let $I_n = \{0, 1, \cdots, n-1\}$ and let $\left(I_n, \mathcal{L}^1, \mathcal{L}^2, (\mathcal{L}_x)_{x \in I_n}\right)$ be an auxiliary design of type 2, $LJX_2(n)$, based on I_n, such that condition (i) in the definition of an auxiliary design of type 2 is satisfied with $c_0 = 0$. Let $S' = \{\infty_1, \infty_2\} \cup I_n$ and let LS be a $LS(STS(n+2))$, based on S', where $LS = \{(S', \mathcal{B}_i) : i \in I_n\}$ and the triple $\{\infty_1, \infty_2, i\}$ belongs to $\mathcal{B}_i$. Finally, let $S = (GF[4] \setminus \{0\}) \times I_n$. Then a $LS(STS(3n))$ based on S can be constructed by taking the following collections of disjoint $STS(3n)$ based on S:

$$(S, \langle \mathcal{L}_i, \mathcal{B}_i \rangle_i) \qquad \text{for } i \in I_n;$$

$$(S, \langle \mathcal{L}^1, \mathcal{B}_i \rangle_i) \qquad \text{for } i \in \{1, 2, \ldots, n-1\};$$

$$(S, \langle \mathcal{L}^2, \mathcal{B}_i \rangle_i) \qquad \text{for } i \in \{1, 2, \ldots, n-1\}.$$

The counting arguments which show that each triple of elements of S occurs precisely once in this large set of Steiner triple systems can be found in Lu [10], Theorem 1.

Example 6.13 A $LJX_2^*(11)$ is shown in Table 22.

The proof of Lemma 4.3 follows from Lemma 6.3 and Theorem 6.11 for $n \geq 35$. For $n = 11$, the result follows from Example 6.13 and Theorem 6.11. The case $n = 23$ is avoided by direct construction of a $LS(STS(69))$. A construction for a $LS(STS(69))$ can be found in Denniston [3], using a starter $STS(69)$, $\mathcal{D}_0$, based on the set $\{0, 1, \ldots, 68\}$, $\mathcal{D}_i = \pi^i \mathcal{D}_0$, for $i = 0, 1, \ldots, 66$, where $\pi = (0, 1, ..., 66)$. $\mathcal{D}_0$ is listed in full in [18], Table A.7.

6.5 LJX–Partitions of Sets

Definition 6.14 *Let* X *be a set of* n *elements, where* n *is an integer. Let* $\mathcal{H} = \{H_{yx} : (y, x) \in (GF[2] \times X)\}$ *be a partitioning of the* $\binom{2n}{2}$ *unordered*

0000	37A1	74A2	0000	3784	7468
01A3	3894	7595	0141	3815	75A9
0296	3987	7688	0282	3956	763A
0389	3A7A	7770	0313	3A97	7770
0471	408A	7863	0454	40A7	7801
0564	4172	7956	0595	4138	7942
0657	4265	7A49	0626	4279	7A83
074A	4358	8059	0767	430A	8093
0832	4440	8141	08A8	4440	8124
0925	4533	8234	0939	4581	8265
0A18	4626	8327	0A7A	4612	83A6
1028	4719	841A	108A	4753	8437
1110	4801	8502	1110	4894	8578
1203	49A4	86A5	1251	4925	8609
13A6	4A97	8798	1392	4A66	874A
1499	50A7	8880	1423	5076	8880
1581	519A	8973	1564	5107	8911
1674	5282	8A66	16A5	5248	8A52
1767	5375	9076	1736	5389	9062
185A	5468	9169	1877	541A	91A3
1942	5550	9251	1908	5550	9234
1A35	5643	9344	1A49	5691	9375
2045	5736	9437	2059	5722	9406
2138	5829	952A	219A	5863	9547
2220	5911	9612	2220	59A4	9688
2313	5A04	9705	2361	5A35	9719
2406	6014	98A8	24A2	6045	985A
25A9	6107	9990	2533	6186	9990
2691	62AA	9A83	2674	6217	9A21
2784	6392	A093	2705	6358	A031
2877	6485	A186	2846	6499	A172
296A	6578	A279	2987	652A	A203
2A52	6660	A361	2A18	6660	A344
3062	6753	A454	3028	67A1	A485
3155	6846	A547	3169	6832	A516
3248	6939	A63A	32AA	6973	A657
3330	6A21	A722	3330	6A04	A798
3423	7031	A815	3471	7014	A829
3516	7124	A908	3502	7155	A96A
3609	7217	AAA0	3643	7296	AAA0
	730A			7327	
	$\mathcal{L}^1$			$\mathcal{L}^2$	

Table 22 A $LJX_2^*(11)$

117 614	009 605	00A 604	005 608	003 601	001 60A
123 62A	026 627	013 617	018 619	01A 610	010 616
132 635	038 636	031 63A	029 626	028 625	022 629
148 641	04A 649	046 640	044 645	030 633	033 634
156 653	055 650	058 659	050 654	057 652	047 643
165 667	062 663	064 668	067 662	066 66A	069 661
171 676	077 672	079 671	076 67A	075 677	074 678
184 688	083 68A	085 686	081 680	082 689	08A 687
19A 699	090 694	097 695	092 691	099 698	096 692
1A9 6A2	0A4 6A8	0A0 6A3	0AA 6A7	0A1 6A6	0A8 6A0
219 713	20A 703	105 709	101 704	100 708	102 707
226 722	228 725	11A 716	110 715	116 719	114 712
233 731	234 730	137 738	124 728	129 72A	120 721
245 749	243 746	149 747	142 740	13A 737	139 736
254 755	259 752	150 75A	157 751	155 756	141 744
262 768	267 764	166 761	169 76A	161 765	168 763
278 77A	276 778	173 774	175 779	178 773	177 770
287 786	282 787	188 783	18A 782	187 780	186 788
291 794	295 799	194 790	196 797	192 791	193 79A
2AA 7A7	2A0 7AA	1A1 7A5	1A8 7A6	1A3 7A2	1AA 7A9
312 811	300 808	301 800	202 806	209 807	204 803
321 828	32A 824	310 814	216 81A	212 815	211 819
33A 834	337 833	339 836	220 827	221 826	227 82A
347 843	344 842	345 841	248 849	235 839	23A 830
358 859	356 85A	354 857	25A 858	253 851	240 848
369 86A	365 866	36A 863	261 860	268 862	266 867
374 877	373 879	378 875	277 872	27A 870	272 876
385 882	389 880	387 889	284 885	286 88A	289 884
396 895	398 897	393 898	299 894	290 893	298 891
3A3 8A6	3A2 8A5	3A6 8AA	2A5 8A1	2A7 8A8	2A3 8A2
418 91A	404 907	403 906	407 900	306 902	308 909
429 924	423 922	411 919	412 911	313 917	31A 918
437 936	432 939	430 935	421 925	327 920	323 926
446 942	440 945	448 944	44A 947	331 938	332 937
45A 957	458 954	455 953	456 952	359 95A	346 94A
464 963	469 96A	467 960	465 968	360 969	364 962
473 975	47A 970	476 977	470 974	372 971	379 973
481 989	485 988	484 98A	489 986	388 983	380 981
492 998	496 993	49A 991	498 99A	395 996	397 990
4A5 9A1	4A7 9A6	4A9 9A8	4A4 9A9	3AA 9A5	3A1 9A4
516 A15	506 A02	508 A07	50A A09	505 A0A	400 A06
525 A27	529 A20	515 A18	514 A17	518 A11	417 A13
538 A39	53A A35	534 A33	522 A2A	523 A22	424 A28
54A A44	548 A47	543 A4A	541 A46	532 A36	438 A31
552 A51	557 A53	551 A56	559 A55	550 A58	442 A49
561 A66	560 A68	569 A65	566 A64	567 A63	46A A60
579 A72	575 A74	57A A70	578 A71	576 A79	471 A7A
583 A8A	584 A86	580 A81	587 A88	581 A85	483 A82
597 A93	592 A9A	596 A99	595 A90	59A A97	499 A94
5A4 AA8	5A3 AA9	5A7 AA4	5A0 AA2	5A9 AA0	4A6 AA7
$\mathcal{L}_0$	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$

Table 22 (cont'd) A $LJX_2^*(11)$

008	507	002	509	007	501	004	500	006	503
017	511	016	513	014	510	011	51A	012	519
024	528	020	52A	025	526	023	527	021	524
039	535	03A	531	036	537	032	533	037	530
042	549	048	545	041	544	040	546	045	542
05A	553	059	554	052	55A	056	558	054	556
071	570	061	568	063	562	065	564	060	565
080	582	084	586	070	573	07A	572	073	577
093	594	095	590	09A	599	088	585	089	588
0A5	5AA	0A3	5A2	0A9	5A5	0A7	5A1	098	591
109	701	106	600	104	603	10A	606	108	602
112	710	119	618	113	61A	118	612	115	611
121	727	128	622	127	624	125	621	122	620
133	73A	135	639	131	630	136	637	134	638
144	745	14A	646	140	642	147	648	143	644
158	754	153	65A	159	656	152	655	151	657
17A	772	160	664	16A	665	163	660	167	669
185	789	182	681	172	679	174	673	176	675
190	798	191	693	195	697	181	684	180	683
1A7	7A3	1A4	6A5	1A6	6A1	1A0	6AA	199	696
200	80A	208	804	205	706	207	702	201	700
213	818	21A	812	217	711	215	714	210	717
225	823	223	821	22A	729	224	720	229	723
231	832	232	838	239	733	238	735	236	732
24A	847	244	840	246	74A	242	741	247	748
252	855	255	856	250	757	251	753	258	759
279	874	269	865	264	760	26A	767	263	766
288	881	280	883	271	775	270	776	274	771
297	899	296	89A	293	792	283	78A	285	784
2A4	8A0	2A1	8A9	2A2	7A4	2A6	7A8	292	795
304	903	305	901	302	90A	303	805	307	809
315	914	311	910	319	915	316	817	318	813
322	92A	324	929	320	923	328	822	326	825
338	930	336	934	334	932	330	83A	335	831
340	941	342	943	343	949	34A	844	349	846
351	959	350	958	355	951	357	850	353	852
377	978	363	966	366	967	361	868	362	864
383	987	38A	985	37A	976	375	871	370	878
39A	995	399	992	391	994	382	886	381	887
3A9	9A2	3A8	9AA	3A7	9A0	3A4	8A3	394	890
402	A05	40A	A03	409	A00	408	A01	405	904
419	A1A	415	A14	416	A12	413	A10	414	916
420	A29	426	A25	422	A21	42A	A26	427	928
434	A37	433	A30	435	A3A	431	A34	439	933
443	A48	449	A41	447	A45	445	A43	441	940
457	A50	451	A52	453	A54	454	A5A	450	955
475	A73	462	A6A	461	A69	466	A62	468	961
48A	A84	488	A89	474	A77	477	A78	472	979
491	A92	494	A98	490	A96	480	A87	486	982
4A8	AA1	4A0	AA6	4AA	AA3	4A2	AA5	493	997
$\mathcal{L}_6$		$\mathcal{L}_7$		$\mathcal{L}_8$		$\mathcal{L}_9$		$\mathcal{L}_{10}$	

Table 22 (cont'd) A $LJX_2^*(11)$

pairs of elements of the set $GF[2] \times X$ *into* $2n$ *systems which satisfies the following conditions:*

(i) H_{0x} *contains* $n-1$ *mutually disjoint pairs not containing the elements* $(0,x)$ *and* $(1,x)$*, for each* $x \in X$ *; and*

(ii) H_{1x} *contains* n *mutually disjoint pairs, for each* $x \in X$ *.*

Such a partitioning based on the set X *will be referred to as a* LJX–partitioning of X *and will be denoted by* $\mathcal{H}[X]$.

Lemma 6.4 *Let* X *be a set of* n *elements, where* n *is an integer. Then a LJX–partitioning of* X, $\mathcal{H}[X]$, *exists for all integers* n.

Proof. The proof is by construction. Refer to Hanani [5] for details. □

6.6 Auxiliary Designs of Type 3

Definition 6.15 *Let* X *be a set of* $3m$ *elements, where* m *is an integer. Let* β_{01}, β_{10} *and* β_{11} *be sets of ordered triples of elements of* X *and let* β_{20} *and* β_{21} *be sets of ordered quadruples of elements of* X *which satisfy the following conditions:*

(i) β_{01}, β_{10}, β_{11}, β_{20} *and* β_{21} *are pairwise disjoint;*

(ii) Every unordered pair of elements of X *occurs in precisely one of* β_{01}, β_{10}, β_{11}, β_{20} *or* β_{21} *; and*

(iii) The set of ordered triples contained in $\beta_{10} \cup \beta_{11}$ *partition the set* X.

Then the set of five sets of blocks β_{01}, β_{10}, β_{11}, β_{20} *and* β_{21} *form an* auxiliary design of type 3, *based on the set* X, *and is denoted by* $LJX_3(m)$. *(Note that when the order of elements within the blocks of a* $LJX_3(m)$ *is ignored, a* $LJX_3(m)$ *is a pairwise balanced design* $B[\{3,4\},1;3m]$ *containing one parallel class of triples.)*

Lemma 6.5 *In any auxiliary design of type* 3, $LJX_3(m)$,

(i) $|\beta_{10} \cup \beta_{11}| = m$; *and*

(ii) $2|\beta_{20} \cup \beta_{21}| + |\beta_{01}| = \dfrac{3m(m-1)}{2}$.

Proof. The first result follows from the elements of β_{10} and β_{11} partitioning X and the second follows from the auxiliary design being a pairwise balanced design when the order of elements within the blocks is ignored. □

6.7 Large Sets of Auxiliary Designs of the First Class

Definition 6.16 *Let X be a set of $3m$ elements, where m is an integer. Let $I_{3m-1} = \{0, 1, \cdots, 3m-2\}$ and let $\left\{ LJX_3^j(m) : j \in I_{3m-1} \right\}$ be a collection of $3m-1$ auxiliary designs of type 3 based on the set X, which satisfy the following conditions:*

(i) Any unordered pair of elements of X is contained in β_{10}^j for precisely one $j \in I_{3m-1}$;

(ii) Any unordered triple of elements of X is contained in precisely one auxiliary design, $LJX_3^j(m)$, either as an element of β_{01}^j, β_{10}^j or β_{11}^j, or as a subset of an element of β_{20}^j or β_{21}^j;

(iii) The set of ordered triples of elements of X covered by the collection of all β_{10}^j, $j \in I_{3m-1}$, is the same as that covered by the collection of all β_{11}^j, $j \in I_{3m-1}$; and

(iv) The set of ordered quadruples of elements of X covered by the collection of all β_{20}^j, $j \in I_{3m-1}$, is the same as that covered by the collection of all β_{21}^j, $j \in I_{3m-1}$.

Then the collection of $3m-1$ auxiliary designs, based on the set X, forms a large set of auxiliary designs of the first class *and is denoted by $LS_1(LJX_3(m))$.*

Construction 6.17 (Lu [11, Lemma 1]) Let X be a set of $3m$ elements, where m is an integer. Let $LS_1(LJX_3(m))$ be a large set of auxiliary designs of the first class based on the set X. Let n be an integer, $n \neq 1$ or 3, so that a transversal design $T[4, 1; 2n]$ exists. Let $(GF[4] \times (GF[2] \times \mathbb{Z}_n), \mathcal{G}_2, \mathcal{A}_2)$

be a transversal design $T[4,1;2n]$, where $\mathcal{G}_2 = \{\{u\} \times (GF[2] \times \mathbb{Z}_n) : u \in GF[4]\}$. Let $\mathcal{H} = \{H_{yzw} : (y,z,w) \in (GF[2] \times GF[2] \times \mathbb{Z}_n)\}$ be a LJX–partitioning of $GF[2] \times \mathbb{Z}_n$, and let $S = \{\infty\} \cup (X \times GF[2] \times GF[2] \times \mathbb{Z}_n)$. Then, for any $(j, y, z, w) \in (I_{3m-1} \times GF[2] \times GF[2] \times \mathbb{Z}_n)$, the system of triples given in Table 34 at the end of this chapter form a $STS(12mn+1)$ based on the set S. The Steiner triple system defined above will be denoted by $\left(S, \left\langle LJX_3^j(m)\right\rangle_{yzw}\right)$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [11], Lemma 1.

Construction 6.18 (Lu [11, Lemma 2]) Let X be a set of $3m$ elements, where m is an integer. Let $LS_1(LJX_3(m))$ be a large set of auxiliary designs of the first class based on the set X. Let n be an integer, $n \neq 1$ or 3, so that a transversal design $T[4,1;2n]$ exists, and let $(GF[4] \times (GF[2] \times \mathbb{Z}_n), \mathcal{G}_2, \mathcal{A}_2)$ be such a transversal design, where $\mathcal{G}_2 = \{\{u\} \times (GF[2] \times \mathbb{Z}_n) : u \in GF[4]\}$. Let $S' = \{\infty\} \cup (GF[2] \times GF[2] \times \mathbb{Z}_n)$ and let (S', β) be a $STS(4n+1)$. Finally, let $S = \{\infty\} \cup (X \times GF[2] \times GF[2] \times \mathbb{Z}_n)$. Then, for any $(y, z, w) \in (GF[2] \times GF[2] \times \mathbb{Z}_n)$, the system of triples given in Table 35 at the end of this chapter form a $STS(12mn+1)$ based on the set S. The Steiner triple system defined above will be denoted by $(S, \langle LS_1(LJX_3(m)), \beta\rangle_{yzw})$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in Lu [11], Lemma 2.

6.8 The Proof of parts (i) and (iii) of Lemma 4.5

The proof is by construction and is based on large sets of auxiliary designs of the first class. This construction uses specific examples given in Lu [11].

Theorem 6.19 (Lu [11, Theorem 3]) *Let n be an integer, $n \neq 1$ or 3. If there exists a $LS_1(LJX_3(m))$ and a $LS(STS(4n+1))$ then there exists a $LS(STS(12mn+1))$.*

Construction 6.20 (Lu [11]) Let X be a set of $3m$ elements, where m is an integer. Let $LS_1(LJX_3(m))$ be a large set of auxiliary designs of the first class based on the set X. Let n be an integer, $n \neq 1$ or 3, so that a transversal design $T[4,1;2n]$ exists. Let $(GF[4] \times (GF[2] \times \mathbb{Z}_n), \mathcal{G}_2, \mathcal{A}_2)$ be such a transversal design $T[4,1;2n]$, where $\mathcal{G}_2 = \{\{u\} \times (GF[2] \times \mathbb{Z}_n) : u \in GF[4]\}$. Let

$\mathcal{H} = \{ H_{yzw} : (y,z,w) \in (GF[2] \times GF[2] \times \mathbb{Z}_n) \}$ be a LJX–partitioning of $GF[2] \times \mathbb{Z}_n$. Let $S' = \{\infty\} \cup (GF[2] \times GF[2] \times \mathbb{Z}_n)$ and let LS be a $LS(STS(4n+1))$, based on S', where

$$LS = \{ (S', \beta_{yzw}) : (y,z,w) \in (GF[2] \times GF[2] \times \mathbb{Z}_n) \setminus \{(0,0,0)\} \} .$$

Finally, let $S = \{\infty\} \cup (X \times GF[2] \times GF[2] \times \mathbb{Z}_n)$. Then we can construct a $LS(STS(12mn+1))$ based on S by taking the following collections of disjoint $STS(12mn+1)$ based on S: $\left(S, \left\langle LJX_3^j(m) \right\rangle_{yzw}\right)$ for $(j,y,z,w) \in \left(I_{3m-1} \times GF[2] \times GF[2] \times \mathbb{Z}_n\right)$, and $\left(S, \left\langle LS_1(LJX_3(m)), \beta_{yzw} \right\rangle_{yzw}\right)$ for $(y,z,w) \in (GF[2] \times GF[2] \times \mathbb{Z}_n) \setminus \{(0,0,0)\}$. The counting arguments which show that each triple of elements of S occurs precisely once in this large set of Steiner triple systems can be found in Lu [11], Theorem 3.

Example 6.21 (Lu [11]) A $LS_1(LJX_3(1))$ is shown in Table 23. Only non–empty blocks are listed.

β_{10}^0 and β_{11}^1	012

Table 23 A $LS_1(LJX_3(1))$

The proof of part (i) of Lemma 4.5 follows from Example 6.21 and Theorem 6.19, with $m = 1$, for all integers n, $n \neq 1$ or 3. For $n = 1$ and $n = 3$, the required transversal designs, $T[4,1;2]$ and $T[4,1;6]$, respectively, do not exist. As there does not exist a $LS(STS(5))$, the case $n = 1$ is vacuously satisfied. For $n = 3$, a $LS(STS(37))$ obtained from a construction given in [8] is listed in [18], Tables A.3 and A.4. Refer to [8] for details of the construction of this large set of Steiner triple systems.

Example 6.22 (Lu [11]) A $LS_1(LJX_3(5))$ is shown in Table 24. Only non–empty blocks are listed. Refer to [11] for details of the construction of this large set of auxiliary designs of the first class.

The proof of part (iii) of Lemma 4.5 follows from Example 6.22 and Theorem 6.19, with $m = 5$, for all integers n, $n \neq 1$ or 3. For $n = 1$ and $n = 3$, the required transversal designs, $T[4,1;2]$ and $T[4,1;6]$, respectively, do not

β_{10}^{0} and β_{11}^{7}	246 189 3CF 7AD 5BE
β_{20}^{0} and β_{21}^{7}	1357 8ACE 9BDF 23AB 45CD 67EF 12DE 478B 569A 16BC 258F 349E 14AF 279C 368D
β_{10}^{1} and β_{11}^{8}	347 28A 6BD 59C 1EF
β_{20}^{1} and β_{21}^{8}	1256 8BCF 9ADE 139B 46CE 57DF 17AC 249F 358E 148D 27BE 36AF 23CD 45AB 6789
β_{10}^{2} and β_{11}^{9}	356 48C 79E 1AB 2DF
β_{20}^{2} and β_{21}^{9}	1247 8BDE 9ACF 159D 26AE 37BF 168F 25BC 34AD 2389 45EF 67CD 13CE 469B 578A
β_{10}^{3} and β_{11}^{10}	167 38B 5AF 2CE 49D
β_{20}^{3} and β_{21}^{10}	2345 89EF ABCD 129A 47CF 56DE 14BE 278D 369C 13DF 468A 579B 158C 26BF 37AE
β_{10}^{4} and β_{11}^{11}	257 68E 1CD 4BF 39A
β_{20}^{4} and β_{21}^{11}	1346 8ADF 9BCE 179F 24AC 35BD 23EF 4589 67AB 15AE 269D 378C 128B 47DE 56CF
β_{10}^{5} and β_{11}^{12}	145 78F 29B 3DE 6AC
β_{20}^{5} and β_{21}^{12}	2367 89CD ABEF 169E 25AD 34BC 138A 46DF 57CE 12CF 479A 568B 17BD 248E 359F
β_{10}^{6} and β_{11}^{13}	123 58D 4AE 69F 7BC
β_{20}^{6} and β_{21}^{13}	4567 89AB CDEF 149C 27AD 36BE 15BF 268C 379D 178E 24BD 35AC 16AD 259E 348F

Table 24 A $LS_1\,(LJX_3\,(5))$

exist. The case $n = 1$ is avoided by direct construction of a $LS\,(STS\,(61))$. A construction for a $LS\,(STS\,(61))$ can be found in Denniston [3], using a starter $STS\,(61)$, $\mathcal{D}_0$, based on the set $\{0, 1, \ldots, 60\}$, $\mathcal{D}_i = \pi^i \mathcal{D}_0$, for $i = 0, 1, \ldots, 58$, where $\pi = (0, 1, ..., 58)$. $\mathcal{D}_0$ is listed in [18], Table A.5. The case $n = 3$ follows by taking $n = 15$ in part (i) of Lemma 4.5.

6.9 Large Sets of Auxiliary Designs of the Second Class

Definition 6.23 *Let X be a set of $3m$ elements, where m is an even integer. Let H be a set of ordered pairs of elements of X which partition X. Let $I_{3m-2} = \{0, 1, \cdots, 3m-2\}$ and let $\left\{ LJX_3^j(m) : j \in I_{3m-2} \right\}$ be a collection of $3m-2$ auxiliary designs of type 3, based on the set X, which satisfy the following conditions:*

(i) β_{11}^j is empty for all $j \in I_{3m-2}$;

(ii) Any unordered pair of distinct elements of X, which does not belong to H, is contained in precisely two ordered triples from the collection of all β_{10}^j, $j \in I_{3m-2}$, and neither element of the unordered pair can occupy the same position in the two ordered triples;

(iii) Any unordered triple of elements of X is contained in precisely one auxiliary design, $LJX_3^j(m)$, either as an element of β_{01}^j or β_{10}^j, or as a subset of an element of β_{20}^j or β_{21}^j;

(iv) The set of unordered quadruples of elements of X covered by the collection of all β_{20}^j, $j \in I_{3m-2}$, is the same as that covered by the collection of all β_{21}^j, $j \in I_{3m-2}$; and

(v) For any block, B, from the collection of all β_{10}^j, $j \in I_{3m-2}$, there is a set of mappings $\phi_{uB} : B \mapsto GF\,[2]$, $u \in GF\,[4]$, which satisfy the following conditions:

(a) if B_1 and B_2 are elements of the collection of all β_{10}^j, $j \in I_{3m-2}$, with $B_1 \cap B_2 = \{x_1, x_2\}$, then for any $u \in GF\,[4]$,

$$\phi_{uB_1}(x_1) + \phi_{uB_1}(x_2) + \phi_{uB_2}(x_1) + \phi_{uB_2}(x_2) = 1; \text{ and}$$

(b) if $u_1, u_2 \in GF\,[4]$, $u_1 \neq u_2$, and $B = \{x_0, x_1, x_2\}$ is an element of the collection of all β_{10}^j, $j \in I_{3m-2}$, then of the three pairs of values

$\{\phi_{u_1B}(x_0), \phi_{u_2B}(x_0)\}$, $\{\phi_{u_1B}(x_1), \phi_{u_2B}(x_1)\}$ *and* $\{\phi_{u_1B}(x_2), \phi_{u_2B}(x_2)\}$, *at least one pair has equal elements and at least one pair has unequal elements.*

Then the collection of $3m-2$ *auxiliary designs, based on the set* X, *forms a* large set of auxiliary designs of the second class *and is denoted by* $LS_2(LJX_3(m))$.

Lemma 6.6 *In any large set* $LS_2(LJX_3(m))$ *of auxiliary designs of the second class, the number of blocks in the collection of all* β_{10}^j, $j \in I_{3m-2}$, *is* $m(3m-2)$.

Proof. The proof is by a counting argument based on the number of pairs of elements contained in X and follows from β_{10}^j partitioning X for all $j \in I_{3m-2}$ after allowing for the pairs contained in H. □

Construction 6.24 (Lu [11, Lemma 3]) Let X be a set of $3m$ elements, where m is an even integer. Let $LS_2(LJX_3(m))$ be a large set of auxiliary designs of the second class based on the set X. Let n be an integer, $n \neq 2$ or 6, so that a transversal design $T[4,1;n]$ exists. Let $(GF[4] \times \mathbb{Z}_n, \mathcal{G}_1, \mathcal{A}_1)$ be such a transversal design $T[4,1;n]$, where $\mathcal{G}_1 = \{\{u\} \times \mathbb{Z}_n : u \in GF[4]\}$. Let $\mathcal{H} = \{H_{yw} : (y,w) \in (GF[2] \times \mathbb{Z}_n)\}$ be a LJX–partitioning of $\mathbb{Z}_n$. Finally, let $S = \{\infty\} \cup (X \times GF[2] \times \mathbb{Z}_n)$. Then the system of triples given in Table 36 at the end of this chapter form a $STS(6mn+1)$ based on the set S, for any $(j, y, w) \in (I_{3m-2} \times GF[2] \times \mathbb{Z}_n)$. The Steiner triple system defined above will be denoted by $\left(S, \left\langle LJX_3^j(m), \phi_{0B}\right\rangle_{yw}\right)$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found [11], Lemma 3.

Construction 6.25 (Lu [11, Lemma 4]) Let X be a set of $3m$ elements, where m is an even integer. Let $LS_2(LJX_3(m))$ be a large set of auxiliary designs of the second class based on the set X. Let n be an integer, $n \neq 2$ or 6, so that a transversal design $T[4,1;n]$ exists. Let $(GF[4] \times \mathbb{Z}_n, \mathcal{G}_1, \mathcal{A}_1)$ be such a transversal design $T[4,1;n]$, where $\mathcal{G}_1 = \{\{u\} \times \mathbb{Z}_n : u \in GF[4]\}$. Let $S' = \{\infty\} \cup (GF[2] \times GF[2] \times \mathbb{Z}_n)$ and let (S', β) be a $STS(4n+1)$. Finally, let $S = \{\infty\} \cup (X \times GF[2] \times \mathbb{Z}_n)$. Then the system of triples given in Table 37 at the end of this chapter form a $STS(6mn+1)$ based on the set S for any $(u, w) \in (GF[4] \times \mathbb{Z}_n)$. The Steiner triple system defined above will be denoted by $(S, \langle LS_2(LJX_3(m)), \phi_{uB}, \beta\rangle_{uw})$. The counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [11], Lemma 4.

6.10 The Proof of part (ii) of Lemma 4.5

The proof is by construction and is based on large sets of auxiliary designs of the second class. This construction uses specific examples given in [11].

Theorem 6.26 (Lu [11, Theorem 6]) *Let n be an integer. If there exist both a $LS_2(LJX_3(m))$ and a $LS(STS(4n+1))$, then a $LS(STS(6mn+1))$ also exists.*

Construction 6.27 (Lu [11]) Let X be a set of $3m$ elements, where m is an even integer. Let $LS_2(LJX_3(m))$ be a large set of auxiliary designs of the second class based on the set X. Let n be an integer, $n \neq 2$ or 6, so that a transversal design $T[4,1;n]$ exists. Let $(GF[4] \times \mathbb{Z}_n, \mathcal{G}_1, \mathcal{A}_1)$ be such a transversal design $T[4,1;n]$, where $\mathcal{G}_1 = \{\{u\} \times \mathbb{Z}_n : u \in GF[4]\}$. Let $\mathcal{H} = \{H_{yw} : (y,w) \in (GF[2] \times \mathbb{Z}_n)\}$ be a LJX–partitioning of $\mathbb{Z}_n$. Let $S' = \{\infty\} \cup (GF[2] \times GF[2] \times \mathbb{Z}_n)$ and let LS be a $LS(STS(4n+1))$, based on S', where

$$LS = \{(S, \beta_{uw}) : (u,w) \in (GF[4] \times \mathbb{Z}_n) \setminus \{(0,0)\}\}.$$

Finally, let $S = \{\infty\} \cup (X \times GF[2] \times \mathbb{Z}_n)$. Then a $LS(STS(6mn+1))$ based on S can be constructed by taking the following collections of disjoint $STS(6mn+1)$ based on S:
$\left(S, \left\langle LJX_3^j(m), \phi_{0B}\right\rangle_{yw}\right)$ for $(j,y,w) \in \left(I_{3m-2} \times GF[2] \times \mathbb{Z}_n\right)$, and
$(S, \langle LS_2(LJX_3(m)), \phi_{uB}, \beta_{uw}\rangle_{uw})$ for $(u,w) \in (GF[4] \times \mathbb{Z}_n) \setminus \{(0,0)\}$.
The counting arguments which show that each triple of elements of S occurs precisely once in this large set of Steiner triple systems can be found in [11], Theorem 6.

Example 6.28 (Lu [11]) A $LS_2(LJX_3(4))$ is shown in Table 25. Only non–empty blocks are listed. The mappings ϕ_{uB} associated with the blocks of each β_{10}^j of this $LS_2(LJX_3(4))$, $j \in \{0, 1, \ldots, 9\}$, are shown in Table 26.

The proof of part (ii) of Lemma 4.5 follows from Example 6.28 and Theorem 6.26, with $m = 4$, for all integers n, $n \neq 2$ or 6. For $n = 2$ and $n = 6$, the required transversal designs, $T[4,1;2]$ and $T[4,1;6]$, respectively, do not exist. The case $n = 2$ is avoided by direct construction of a $LS(STS(49))$. A construction for a $LS(STS(49))$ can be found in Denniston [3], using a starter $STS(49)$, $\mathcal{D}_0$, based on the set $\{0, 1, \ldots, 48\}$, $\mathcal{D}_i = \pi^i \mathcal{D}_0$, for

β_{01}^0	00 21 41 01 20 41 01 21 40 00 30 50 00 31 51 01 31 50 10 30 40 11 30 41 11 31 40 10 20 50 10 21 51 11 20 51
β_{10}^0	00 20 40 01 30 51 10 31 41 11 21 50
β_{20}^0	00 01 10 11 20 21 30 31 40 41 50 51
β_{01}^1	10 40 20 10 41 21 11 40 21 10 51 31 11 50 31 11 51 30 00 50 20 01 50 21 01 51 20 00 40 30 00 41 31 01 41 30
β_{10}^1	11 41 20 10 50 30 00 51 21 01 40 31
β_{21}^1	00 01 10 11 20 21 30 31 40 41 50 51
β_{01}^2	00 10 30 00 11 31 01 10 31 00 41 51 01 40 51 01 41 50 10 50 21 11 50 20 11 51 21 20 40 31 20 41 30 21 40 30
β_{10}^2	01 11 30 00 40 50 10 51 20 21 41 31
β_{20}^2	00 01 20 21 10 11 40 41 30 31 50 51
β_{01}^3	10 00 51 11 00 50 11 01 51 10 20 30 10 21 31 11 21 30 00 31 40 01 30 40 01 31 41 40 20 50 41 20 51 41 21 50
β_{10}^3	10 01 50 11 20 31 00 30 41 40 21 51
β_{21}^3	00 01 20 21 10 11 40 41 30 31 50 51
β_{01}^4	00 11 21 01 10 21 01 11 20 00 50 41 00 51 40 01 50 40 30 10 41 31 10 40 31 11 41 20 30 50 20 31 51 21 30 51
β_{10}^4	00 10 20 01 51 41 30 11 40 21 31 50
β_{20}^4	00 01 30 31 10 11 50 51 20 21 40 41

Table 25 A $LS_2(LJX_3(4))$

β_{01}^{5}	10 00 40 10 01 41 11 01 40 10 31 20 11 30 20 11 31 21 50 00 21 51 00 20 51 01 21 40 50 30 41 50 31 41 51 30
β_{10}^{5}	11 00 41 10 30 21 50 01 20 40 51 31
β_{21}^{5}	00 01 30 31 10 11 50 51 20 21 40 41
β_{01}^{6}	00 20 11 00 21 10 01 20 10 00 51 30 01 50 30 01 51 31 10 40 50 11 40 51 11 41 50 20 31 41 21 30 41 21 31 40
β_{10}^{6}	01 21 11 00 50 31 10 41 51 20 30 40
β_{20}^{6}	00 01 40 41 10 11 30 31 20 21 50 51
β_{01}^{7}	10 50 00 10 51 01 11 50 01 40 11 20 41 10 20 41 11 21 00 30 20 00 31 21 01 30 21 40 30 51 40 31 50 41 31 51
β_{10}^{7}	11 51 00 40 10 21 01 31 20 41 30 50
β_{21}^{7}	00 01 40 41 10 11 30 31 20 21 50 51
β_{01}^{8}	00 30 11 01 30 10 01 31 11 20 00 41 20 01 40 21 00 40 10 50 41 10 51 40 11 51 41 20 50 31 21 50 30 21 51 31
β_{10}^{8}	00 31 10 21 01 41 11 50 40 20 51 30
β_{20}^{8}	00 01 50 51 10 11 20 21 30 31 40 41
β_{01}^{9}	10 41 00 11 40 00 11 41 01 10 30 51 10 31 50 11 30 50 00 20 31 01 20 30 01 21 31 40 50 21 40 51 20 41 51 21
β_{10}^{9}	10 40 01 11 31 51 00 21 30 41 50 20
β_{21}^{9}	00 01 50 51 10 11 30 31 20 21 40 41

Table 25 (cont'd) A $LS_2\,(LJX_3\,(4))$

β_{10}^{0}	00	20	40	01	30	51	10	31	41	11	21	50
β_{10}^{1}	11	41	20	10	50	30	00	51	21	01	40	31
ϕ_{0B}	0	0	0	0	0	1	1	0	0	1	0	0
ϕ_{1B}	1	0	0	0	1	0	0	0	1	0	1	0
ϕ_{aB}	0	0	1	1	0	0	0	1	0	0	0	0
ϕ_{bB}	0	1	0	0	0	0	0	0	0	0	0	1
β_{10}^{2}	01	11	30	00	40	50	10	51	20	21	41	31
β_{10}^{3}	10	01	50	11	20	31	00	30	41	40	21	51
ϕ_{0B}	0	0	1	0	1	0	0	1	0	0	1	0
ϕ_{1B}	0	1	0	0	0	1	0	0	0	0	0	0
ϕ_{aB}	0	0	0	0	0	0	1	0	0	1	0	0
ϕ_{bB}	1	0	0	1	0	0	0	0	1	0	0	1
β_{10}^{4}	00	10	20	01	51	41	30	11	40	21	31	50
β_{10}^{5}	11	00	41	10	30	21	50	01	20	40	51	31
ϕ_{0B}	0	0	1	0	0	1	0	0	1	1	0	0
ϕ_{1B}	0	1	0	1	0	0	0	0	0	0	1	0
ϕ_{aB}	1	0	0	0	0	0	0	1	0	0	0	1
ϕ_{bB}	0	0	0	0	1	0	1	0	0	0	0	0
β_{10}^{6}	01	21	11	00	50	31	10	41	51	20	30	40
β_{10}^{9}	10	40	01	11	31	51	00	21	30	41	50	20
ϕ_{0B}	1	0	0	0	1	0	0	0	0	1	0	0
ϕ_{1B}	0	0	0	0	0	0	0	0	1	0	0	1
ϕ_{aB}	0	0	1	1	0	0	0	1	0	0	1	0
ϕ_{bB}	0	1	0	0	0	1	1	0	0	0	0	0
β_{10}^{8}	00	31	10	21	01	41	11	50	40	20	51	30
β_{10}^{7}	11	51	00	40	10	21	01	31	20	41	30	50
ϕ_{0B}	1	0	0	0	0	0	0	0	0	0	0	0
ϕ_{1B}	0	1	0	1	0	0	1	0	0	1	0	0
ϕ_{aB}	0	0	1	0	1	0	0	1	0	0	1	0
ϕ_{bB}	0	0	1	0	0	1	0	0	1	0	0	1

Table 26 The mappings ϕ_{uB} for the $LS_2\,(LJX_3\,(4))$ shown in Table 25

$i = 0, 1, \ldots, 46$, where $\pi = (0, 1, ..., 46)$. $\mathcal{D}_0$ is listed in [18], Table A.5. The case $n = 6$ is avoided by taking $n = 12$ in part (i) of Lemma 4.5.

6.11 Auxiliary Designs of Type 4

Definition 6.29 *Let X be a set of $3m$ elements, where m is an integer. Let β_{10}, β_{11}, β_{a0}, β_{a1} and β_{b1} be sets of ordered triples of elements of X which satisfy the following conditions:*

(i) β_{10}, β_{11}, β_{a0}, β_{a1} and β_{b1} are pairwise disjoint;

(ii) Every unordered pair of elements of X occurs in precisely one of β_{10}, β_{11}, β_{a0}, β_{a1} or β_{b1}; and

(iii) The set of ordered triples contained in $\beta_{10} \cup \beta_{a0} \cup \beta_{a1} \cup \beta_{b1}$ partition the set X.

Then the set of five sets of blocks β_{10}, β_{11}, β_{a0}, β_{a1} and β_{b1} form an auxiliary design of type 4, *based on the set X, and is denoted by $LJX_4(m)$. (Note that when the order of elements within the blocks of a $LJX_4(m)$ is ignored, a $LJX_4(m)$ is a Steiner triple system $STS(3m)$ containing one parallel class of triples.)*

Lemma 6.7 *In any auxiliary design of type 4, $LJX_4(m)$,*

(i) $|\beta_{10} \cup \beta_{a0} \cup \beta_{a1} \cup \beta_{b1}| = m$; and

(ii) $|\beta_{11}| = \dfrac{3m(m-1)}{2}$.

Proof. The first result follows from the elements of $\beta_{10} \cup \beta_{a0} \cup \beta_{a1} \cup \beta_{b1}$ partitioning X and the second follows from the auxiliary design being a balanced incomplete block design when the order of elements within the blocks is ignored. □

6.12 Large Sets of Auxiliary Designs of the Third Class

Definition 6.30 *Let X be a set of $3m$ elements, where m is an integer. Let $I_{3m-2} = \{0, 1, \cdots, 3m-2\}$ and let $\left\{ LJX_4^j(m) : j \in I_{3m-2} \right\}$ be a collection of $3m-2$ auxiliary designs of type 4 based on the set X, which satisfy the following conditions:*

(i) Every unordered pair, $\{x_1, x_2\}$, of elements of X satisfies precisely one of the following conditions:

(a) $\{x_1, x_2\} \subset B$ where $B \in \beta_{10}^j$, for some $j \in I_{3m-2}$;

(b) $\{x_1, x_2\} = B_1 \cap B_2$ where $B_1 \in \beta_{a0}^{j_1}$ and $B_2 \in \beta_{a1}^{j_2}$, for some $j_1, j_2 \in I_{3m-2}$. In this case, if x_1 is in position p in B_1 and x_2 is in position q in B_1 then x_1 is in position q in B_2 and x_2 is in position p in B_2; or

(c) $\{x_1, x_2\} = B_1 \cap B_2$ where $B_1 \in \beta_{a0}^{j_1}$ and $B_2 \in \beta_{b1}^{j_2}$, for some $j_1, j_2 \in I_{3m-2}$. In this case, if x_1 is in position p in B_1 and x_2 is in position q in B_1 then x_1 is in position p in B_2 and x_2 is in position q in B_2;

(ii) Any unordered triple of elements of X is contained in precisely one auxiliary design, $LJX_4^j(m)$, as an element of β_{10}^j, β_{11}^j, β_{a0}^j, β_{a1}^j or β_{b1}^j ; and

(iii) The set of ordered triples of elements of X contained in the collection of all β_{10}^j, $j \in I_{3m-2}$, partition the set X .

Then the collection of $3m-2$ auxiliary designs, based on the set X, forms a large set of auxiliary designs of the third class *and is denoted by $LS_3(LJX_4(m))$.*

Lemma 6.8 *In any large set $LS_3(LJX_4(m))$ of auxiliary designs of the third class:*

(i) $|\bigcup_{j \in I_{3m-2}} \beta_{10}^j| = m$; and

(ii) $|\bigcup_{j \in I_{3m-2}} \left(\beta_{a0}^j \cup \beta_{a1}^j \cup \beta_{b1}^j\right)| = 3m(m-1)$.

Proof. The first result follows from the elements of $\bigcup_{j \in I_{3m-2}} \beta_{10}^j$ partitioning X and the second result is obtained by a counting argument based on the first result and the results of Lemma 6.7. □

Construction 6.31 (Lu [11, Lemma 5]) Let X be a set of $3m$ elements, where m is an integer. Let $LS_3(LJX_4(m))$ be a large set of auxiliary designs of the third class based on the set X. Let n be an odd integer, so that a transversal design $T[4,1;n]$ exists. Let $(GF[4] \times \mathbb{Z}_n, \mathcal{G}_1, \mathcal{A}_1)$ be such a transversal design $T[4,1;n]$, where $\mathcal{G}_1 = \{\{u\} \times \mathbb{Z}_n : u \in GF[4]\}$. Let

$$S' = \{\infty\} \cup ((GF[4] \setminus \{0\}) \times GF[4] \times \mathbb{Z}_n)$$

and (S', β) be a $STS(12n+1)$. Finally, let $S = \{\infty\} \cup (X \times GF[4] \times \mathbb{Z}_n)$. Then, for any $(j, u, w) \in (I_{3m-2} \times GF[4] \times \mathbb{Z}_n)$, the system of triples given in Table 38 at the end of this chapter form a $STS(12mn+1)$ based on the set S, and $\left(S, \left\langle LJX_4^j(m), \beta \right\rangle_{uw}\right)$ denotes the Steiner triple system defined above. Counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [11], Lemma 5.

Construction 6.32 (Lu [11, Lemma 6]) Let X be a set of $3m$ elements, where m is an integer. Let $LS_3(LJX_4(m))$ be a large set of auxiliary designs of the third class based on the set X. Let n be an odd integer, so that a transversal design $T[4,1;n]$ exists. Let $(GF[4] \times \mathbb{Z}_n, \mathcal{G}_1, \mathcal{A}_1)$ be such a transversal design $T[4,1;n]$, where $\mathcal{G}_1 = \{\{u\} \times \mathbb{Z}_n : u \in GF[4]\}$. Let

$$S' = \{\infty\} \cup ((GF[4] \setminus \{0\}) \times GF[4] \times \mathbb{Z}_n)$$

and (S', β) be a $STS(12n+1)$. Finally, let $S = \{\infty\} \cup (X \times GF[4] \times \mathbb{Z}_n)$. Then, for any $(y, u, w) \in (GF[2] \times GF[4] \times \mathbb{Z}_n)$, the system of triples given in Table 39 at the end of this chapter form a $STS(12mn+1)$ based on the set S, and $(S, \langle LS_3(LJX_4(m)), \beta\rangle_{yuw})$ denotes the Steiner triple system defined above. Counting arguments which show that each pair of elements of S occurs precisely once in this Steiner triple system can be found in [11], Lemma 6.

6.13 The Proof of parts (i) and (ii) of Lemma 4.6

The proof is by construction and is based on large sets of auxiliary designs of the third class. This construction uses specific examples given in Lu [11].

Theorem 6.33 (Lu [11, Theorem 8]) *Let n be an odd integer. If there exist both a $LS_3(LJX_4(m))$ and a $LS(STS(12n+1))$ then a $LS(STS(12mn+1))$ also exists.*

Construction 6.34 (Lu [11]) Let X be a set of $3m$ elements, where m is an integer. Let $LS_3(LJX_4(m))$ be a large set of auxiliary designs of the third class based on the set X. Let n be an odd integer, so that a transversal design $T[4,1;n]$ exists. Let $(GF[4] \times \mathbb{Z}_n, \mathcal{G}_1, \mathcal{A}_1)$ be such a transversal design $T[4,1;n]$, where $\mathcal{G}_1 = \{\{u\} \times \mathbb{Z}_n : u \in GF[4]\}$. Let

$$S' = \{\infty\} \cup ((GF[4] \setminus \{0\}) \times GF[4] \times \mathbb{Z}_n)$$

and let LS be a $LS(STS(12n+1))$, based on S', where

$$\begin{aligned} LS = \; & \{(S', \beta_{uw}) : (u,w) \in (GF[4] \times \mathbb{Z}_n)\} \cup \\ & \{(S', \beta_{yuw}) : (y,u,w) \in (GF[2] \times GF[4] \times \mathbb{Z}_n) \setminus \{(0,0,0)\}\}. \end{aligned}$$

Finally, let $S = \{\infty\} \cup (X \times GF[4] \times \mathbb{Z}_n)$. Then a $LS(STS(12mn+1))$ based on S can be constructed by taking the following collections of disjoint $STS(12mn+1)$ based on S:
$\left(S, \left\langle LJX_4^j(m), \beta_{uw}\right\rangle_{uw}\right)$ for $(j,u,w) \in \left(I_{3m-2} \times GF[4] \times \mathbb{Z}_n\right)$, and
$\left(S, \left\langle LS_3(LJX_4(m)), \beta_{yuw}\right\rangle_{yuw}\right)$ for
$(y,u,w) \in (GF[2] \times GF[4] \times \mathbb{Z}_n) \setminus \{(0,0,0)\}$. The counting arguments which show that each triple of elements of S occurs precisely once in this large set of Steiner triple systems can be found in [11], Theorem 8.

Example 6.35 (Lu [11]) A $LS_3(LJX_4(7))$ is shown in Table 27. Only non-empty blocks are listed. Refer to [11] for details of the construction of this large set of auxiliary designs of the third class.

The proof of part (i) of Lemma 4.6 follows from Example 6.35 and Theorem 6.33.

Example 6.36 (Lu [11]) A $LS_3(LJX_4(11))$ is shown in Table 28. Only non-empty blocks are listed. Refer to [11] for details of the construction of this large set of auxiliary designs of the third class.

The proof of part (ii) of Lemma 4.6 follows from Example 6.36 and Theorem 6.33.

β_{10}^{0}	10 a0 b0
β_{10}^{1}	11 a1 b1

β_{10}^{2}	12 a2 b2
β_{10}^{3}	13 a3 b3
β_{10}^{4}	14 a4 b4

β_{10}^{5}	15 a5 b5
β_{10}^{6}	16 a6 b6

β_{a0}^{0}	11 12 14 a1 a2 a4 b1 b2 b4
β_{a1}^{0}	16 15 13 a6 a5 a3 b6 b5 b3
β_{a0}^{1}	12 13 15 a2 a3 a5 b2 b3 b5
β_{a1}^{1}	10 16 14 a0 a6 a4 b0 b6 b4
β_{a0}^{2}	13 14 16 a3 a4 a6 b3 b4 b6
β_{a1}^{2}	11 10 15 a1 a0 a5 b1 b0 b5
β_{a0}^{3}	14 15 10 a4 a5 a0 b4 b5 b0

β_{a1}^{3}	12 11 16 a2 a1 a6 b2 b1 b6
β_{a0}^{4}	15 16 11 a5 a6 a1 b5 b6 b1
β_{a1}^{4}	13 12 10 a3 a2 a0 b3 b2 b0
β_{a0}^{5}	16 10 12 a6 a0 a2 b6 b0 b2
β_{a1}^{5}	14 13 11 a4 a3 a1 b4 b3 b1
β_{a0}^{6}	10 11 13 a0 a1 a3 b0 b1 b3
β_{a1}^{6}	15 14 12 a5 a4 a2 b5 b4 b2

β_{a0}^{01}	14 a0 b6 15 a1 b0 16 a2 b1 10 a3 b2 11 a4 b3 12 a5 b4 13 a6 b5
β_{a0}^{02}	11 a0 b5 12 a1 b6 13 a2 b0 14 a3 b1 15 a4 b2 16 a5 b3 10 a6 b4
β_{a0}^{03}	15 a0 b4 16 a1 b5 10 a2 b6 11 a3 b0 12 a4 b1 13 a5 b2 14 a6 b3
β_{a0}^{04}	12 a0 b3 13 a1 b4 14 a2 b5 15 a3 b6 16 a4 b0 10 a5 b1 11 a6 b2
β_{a0}^{05}	16 a0 b2 10 a1 b3 11 a2 b4 12 a3 b5 13 a4 b6 14 a5 b0 15 a6 b1
β_{a0}^{06}	13 a0 b1 14 a1 b2 15 a2 b3 16 a3 b4 10 a4 b5 11 a5 b6 12 a6 b0
β_{b1}^{11}	12 a0 b6 13 a1 b0 14 a2 b1 15 a3 b2 16 a4 b3 10 a5 b4 11 a6 b5
β_{b1}^{12}	14 a0 b5 15 a1 b6 16 a2 b0 10 a3 b1 11 a4 b2 12 a5 b3 13 a6 b4
β_{b1}^{13}	16 a0 b4 10 a1 b5 11 a2 b6 12 a3 b0 13 a4 b1 14 a5 b2 15 a6 b3
β_{b1}^{14}	11 a0 b3 12 a1 b4 13 a2 b5 14 a3 b6 15 a4 b0 16 a5 b1 10 a6 b2
β_{b1}^{15}	13 a0 b2 14 a1 b3 15 a2 b4 16 a3 b5 10 a4 b6 11 a5 b0 12 a6 b1
β_{b1}^{16}	15 a0 b1 16 a1 b2 10 a2 b3 11 a3 b4 12 a4 b5 13 a5 b6 14 a6 b0

Table 27 A $LS_3\,(LJX_4\,(7))$

β_{10}	∞ 000 001 010 0a0 0b0 011 0a1 0b1 100 a00 b00 101 a01 b01 110 ab0 ba0 111 ab1 ba1 1a0 a10 bb0 1a1 a11 bb1 1b0 aa0 b10 1b1 aa1 b11
$\beta_{a0}^{\infty 10}$	100 a10 b10 110 aa0 bb0 1a0 a00 ba0 1b0 ab0 b00 ∞ 0a0 0b1 000 010 011 001 0a1 0b0
$\beta_{b1}^{\infty 10}$	101 a11 b11 111 aa1 bb1 1a1 a01 ba1 1b1 ab1 b01
$\beta_{a0}^{\infty 11}$	101 a10 b11 111 aa0 bb1 1a1 a00 ba1 1b1 ab0 b01
$\beta_{b1}^{\infty 11}$	100 a11 b10 110 aa1 bb0 1a0 a01 ba0 1b0 ab1 b00 ∞ 0a1 0b0 001 010 011 000 0a0 0b1
$\beta_{a0}^{\infty a0}$	100 aa0 ba0 110 a10 b00 1a0 ab0 b10 1b0 a00 bb0 ∞ 011 0b0 001 010 0b1 000 0a0 0a1
$\beta_{b1}^{\infty a0}$	101 aa1 ba1 111 a11 b01 1a1 ab1 b11 1b1 a01 bb1
$\beta_{a0}^{\infty a1}$	101 aa0 ba1 111 a10 b01 1a1 ab0 b11 1b1 a00 bb1
$\beta_{b1}^{\infty a1}$	100 aa1 ba0 110 a11 b00 1a0 ab1 b10 1b0 a01 bb0 ∞ 010 0b1 000 011 0b0 001 0a0 0a1
$\beta_{a0}^{\infty b0}$	100 ab0 bb0 110 a00 b10 1a0 aa0 b00 1b0 a10 ba0 ∞ 010 0a1 001 011 0a0 000 0b0 0b1
$\beta_{b1}^{\infty b0}$	101 ab1 bb1 111 a01 b11 1a1 aa1 b01 1b1 a11 ba1
$\beta_{a0}^{\infty b1}$	101 ab0 bb1 111 a00 b11 1a1 aa0 b01 1b1 a10 ba1
$\beta_{b1}^{\infty b1}$	100 ab1 bb0 110 a01 b10 1a0 aa1 b00 1b0 a11 ba0 ∞ 011 0a0 000 010 0a1 001 0b0 0b1

Table 28 A $LS_3(LJX_4(11))$

β_{a0}^{010}	011	1a1	1b0	0a1	a11	ab0	0b1	b11	ba0	111	aa1	bb0
	∞	a00	b01	000	100	101	001	a01	b00			
β_{b1}^{010}	010	1a0	1b1	0a0	a10	ab1	0b0	b10	ba1	110	aa0	bb1
β_{a0}^{011}	010	1a1	1b1	0a0	a11	ab1	0b0	b11	ba1	110	aa1	bb1
β_{b1}^{011}	011	1a0	1b0	0a1	a10	ab0	0b1	b10	ba0	111	aa0	bb0
	∞	a01	b00	001	100	101	000	a00	b01			
β_{a0}^{0a0}	011	ba1	bb0	0a1	111	1b0	0b1	a11	aa0	1a1	ab1	b10
	∞	101	b00	001	100	b01	000	a00	a01			
β_{b1}^{0a0}	010	ba0	bb1	0a0	110	1b1	0b0	a10	aa1	1a0	ab0	b11
β_{a0}^{0a1}	010	ba1	bb1	0a0	111	1b1	0b0	a11	aa1	1a0	ab1	b11
β_{b1}^{0a1}	011	ba0	bb0	0a1	110	1b0	0b1	a10	aa0	1a1	ab0	b10
	∞	100	b01	000	101	b00	001	a00	a01			
β_{a0}^{0b0}	011	aa1	ab0	0a1	b11	bb0	0b1	111	1a0	1b1	a11	ba0
	∞	100	a01	001	101	a00	000	b00	b01			
β_{b1}^{0b0}	010	aa0	ab1	0a0	b10	bb1	0b0	110	1a1	1b0	a10	ba1
β_{a0}^{0b1}	010	aa1	ab1	0a0	b11	bb1	0b0	111	1a1	1b0	a11	ba1
β_{b1}^{0b1}	011	a00	ab0	0a1	b10	bb0	0b1	110	1a0	1b1	a10	ba0
	∞	101	a00	000	100	a01	001	b00	b01			
β_{a0}^{110}	010	1a0	1b0	0a0	a00	aa0	0b1	b01	bb0	101	a11	b10
	∞	ab0	ba1	000	110	111	001	ab1	ba0			
β_{b1}^{110}	011	1a1	1b1	0a1	a01	aa1	0b0	b00	bb1	100	a10	b11
β_{a0}^{111}	011	1a0	1b1	0a1	a00	aa1	0b0	b01	bb1	100	a11	b11
β_{b1}^{111}	010	1a1	1b0	0a0	a01	aa0	0b1	b00	bb0	101	a10	b10
	∞	ab1	ba0	001	110	111	000	ab0	ba1			

Table 28 (cont'd) A LS_3 (LJX_4 (11))

β_{a0}^{1a0}	010	b00	b10	0a1	101	1a0	0b0	a10	aa0	1b1	a01	bb0
	∞	111	ba0	001	110	ba1	000	ab0	ab1			
β_{b1}^{1a0}	011	b01	b11	0a0	100	1a1	0b1	a11	aa1	1b0	a00	bb1
β_{a0}^{1a1}	011	b00	b11	0a0	101	1a1	0b1	a10	aa1	1b0	a01	bb1
β_{b1}^{1a1}	010	b01	b10	0a1	100	1a0	0b0	a11	aa0	1b1	a00	bb0
	∞	110	ba1	000	111	ba0	001	ab0	ab1			
β_{a0}^{1b0}	011	a01	a10	0a0	b10	bb0	0b0	100	1b0	1a1	aa1	b00
	∞	110	ab1	001	111	ab0	000	ba0	ba1			
β_{b1}^{1b0}	010	a00	a11	0a1	b11	bb1	0b1	101	1b1	1a0	aa0	b01
β_{a0}^{1b1}	010	a01	a11	0a1	b10	bb1	0b1	100	1b1	1a0	aa1	b01
β_{b1}^{1b1}	011	a00	a10	0a0	b11	bb0	0b0	101	1b0	1a1	aa0	b00
	∞	111	ab0	000	110	ab1	001	ba0	ba1			
β_{a0}^{a10}	011	b01	b10	0a0	110	1b0	0b0	a00	ab0	101	aa1	ba0
	∞	a10	bb1	000	1a0	1a1	001	a11	bb0			
β_{b1}^{a10}	010	b00	b11	0a1	111	1b1	0b1	a01	ab1	100	aa0	ba1
β_{a0}^{a11}	010	b01	b11	0a1	110	1b1	0b1	a00	ab1	100	aa1	ba1
β_{b1}^{a11}	011	b00	b10	0a0	111	1b0	0b0	a01	ab0	101	aa0	ba0
	∞	a11	bb0	001	1a0	1a1	000	a10	bb1			
β_{a0}^{aa0}	010	aa0	ab0	0a0	b00	ba0	0b1	101	1b0	111	a01	b10
	∞	1a1	bb0	001	1a0	bb1	000	a10	a11			
β_{b1}^{aa0}	011	aa1	ab1	0a1	b01	ba1	0b0	100	1b1	110	a00	b11
β_{a0}^{aa1}	011	aa0	ab1	0a1	b00	ba1	0b0	101	1b1	110	a01	b11
β_{b1}^{aa1}	010	aa1	ab0	0a0	b01	ba0	0b1	100	1b0	111	a00	b10
	∞	1a0	bb1	000	1a1	bb0	001	a10	a11			

Table 28 (cont'd) A $LS_3\,(LJX_4\,(11))$

β_{a0}^{ab0}	010	100	110	0a1	a01	aa0	0b0	b10	ba0	1b1	ab1	b00
	∞	1a0	a11	001	1a1	a10	000	bb0	bb1			
β_{b1}^{ab0}	011	101	111	0a0	a00	aa1	0b1	b11	ba1	1b0	ab0	b01
β_{a0}^{ab1}	011	100	111	0a0	a01	aa1	0b1	b10	ba1	1b0	ab1	b01
β_{b1}^{ab1}	010	101	110	0a1	a00	aa0	0b0	b11	ba0	1b1	ab0	b00
	∞	1a1	a10	000	1a0	a11	001	bb0	bb1			
β_{a0}^{b10}	010	a00	a10	0a1	b01	ba0	0b0	110	1a0	101	ab1	bb0
	∞	aa0	b11	000	1b0	1b1	001	aa1	b10			
β_{b1}^{b10}	011	a01	a11	0a0	b00	ba1	0b1	111	1a1	100	ab0	bb1
β_{a0}^{b11}	011	a00	a11	0a0	b01	ba1	0b1	110	1a1	100	ab1	bb1
β_{b1}^{b11}	010	a01	a10	0a1	b00	ba0	0b0	111	1a0	101	ab0	bb0
	∞	aa1	b10	001	1b0	1b1	000	aa0	b11			
β_{a0}^{ba0}	011	101	110	0a0	a10	ab0	0b0	b00	bb0	1a1	a01	ba0
	∞	1b1	b10	001	1b0	b11	000	aa0	aa1			
β_{b1}^{ba0}	010	100	111	0a1	a11	ab1	0b1	b01	bb1	1a0	a00	ba1
β_{a0}^{ba1}	010	101	111	0a1	a10	ab1	0b1	b00	bb1	1a0	a01	ba1
β_{b1}^{ba1}	011	100	110	0a0	a11	ab0	0b0	b01	bb0	1a1	a00	ba0
	∞	1b0	b11	000	1b1	b10	001	aa0	aa1			
β_{a0}^{bb0}	010	ba0	bb0	0a0	100	1a0	0b1	a01	ab0	111	a11	b00
	∞	1b0	aa1	001	1b1	aa0	000	b10	b11			
β_{b1}^{bb0}	011	ba1	bb1	0a1	101	1a1	0b0	a00	ab1	110	a10	b01
β_{a0}^{bb1}	011	ba0	bb1	0a1	100	1a1	0b0	a01	ab1	110	a11	b01
β_{b1}^{bb1}	010	ba1	bb0	0a0	101	1a0	0b1	a00	ab0	111	a10	b00
	∞	1b1	aa0	000	1b0	aa1	001	b10	b11			

Table 28 (cont'd) A LS_3 (LJX_4 (11))

LJX_1	$STS(12n+1)$
$(i_1, i_2, i_3) \in \widetilde{\mathcal{B}}$	$\{i_1j_1, i_2j_2, i_3j_3\}$ where $\{i_3j_3, \mathtt{C}s, \mathtt{D}j_4, \cdot\} \in \mathcal{A}$; $j \neq s$ $\{i_1j_1, i_2j_2, \mathtt{C}j, \mathtt{D}j_4\} \in \mathcal{A}$
$(i_1, i_2) \in \rho^1$	$\{\infty, i_1j_1, i_1j_2\}$ where $\{\infty, \mathtt{0}j_1, \mathtt{0}j_2\} \in \mathcal{B}'$
	$\{\infty, i_1j_1, i_2j_2\}$ where $\{\infty, \mathtt{0}j_1, \mathtt{1}j_2\} \in \mathcal{B}'$
	$\{\infty, i_2j_1, i_2j_2\}$ where $\{\infty, \mathtt{1}j_1, \mathtt{1}j_2\} \in \mathcal{B}'$
	$\{i_1j_1, i_1j_2, i_1j_3\}$ where $\{\mathtt{0}j_1, \mathtt{0}j_2, \mathtt{0}j_3\} \in \mathcal{B}'$
	$\{i_1j_1, i_1j_2, i_2j_3\}$ where $\{\mathtt{0}j_1, \mathtt{0}j_2, \mathtt{1}j_3\} \in \mathcal{B}'$
	$\{i_2j_1, i_2j_2, i_1j_3\}$ where $\{\mathtt{1}j_1, \mathtt{1}j_2, \mathtt{0}j_3\} \in \mathcal{B}'$
	$\{i_2j_1, i_2j_2, i_2j_3\}$ where $\{\mathtt{1}j_1, \mathtt{1}j_2, \mathtt{1}j_3\} \in \mathcal{B}'$
$(i_1, i_2) \in \rho^2$	$\{i_1j_1, i_1j_2, i_2j_3\}$ where $\{i_1j, i_2j_3, \mathtt{C}s, \cdot\} \in \mathcal{A}$ $j_1 \neq j_2$; $2j \equiv j_1 + j_2 \bmod n$
$(i_1, i_2) \notin \rho^1$ and $(i_2, i_1) \notin \rho^1$	$\{\infty, i_1j_1, i_2j_2\}$ where $\{i_1j_1, i_2j_2, \mathtt{C}s, \cdot\} \in \mathcal{A}$ $\{\infty, i_1, i_2\} \in \mathcal{B}$
	$\{i_1j_1, i_2j_2, i_3j_3\}$ where $\{i_1j_1, i_2j_2, i_3j_3, \mathtt{C}s\} \in \mathcal{A}$ $\{i_1, i_2, i_3\} \in \mathcal{B}$

Table 29 Constructing a $STS(12n+1)$ from a LJX_1 in which $\rho^2 \neq \emptyset$

LJX_1	$STS(12n+1)$
$(i_1, i_2, i_3) \in \widetilde{\mathcal{B}}$	$\{i_1 j_1, i_2 j_2, i_3 j_3\}$ where $\{i_1 j_1, i_2 j_2, i_3 j_4, \cdot\} \in \mathcal{A}$ $j_3 = j_4 + s \bmod n$
$(i_1, i_2) \in \rho^1$	$\{\infty, i_1 j_1, i_1 j_2\}$ where $\{\infty, 0 j_1, 0 j_2\} \in \mathcal{B}'$
	$\{\infty, i_1 j_1, i_2 j_2\}$ where $\{\infty, 0 j_1, 1 j_2\} \in \mathcal{B}'$
	$\{\infty, i_2 j_1, i_2 j_2\}$ where $\{\infty, 1 j_1, 1 j_2\} \in \mathcal{B}'$
	$\{i_1 j_1, i_1 j_2, i_1 j_3\}$ where $\{0 j_1, 0 j_2, 0 j_3\} \in \mathcal{B}'$
	$\{i_1 j_1, i_1 j_2, i_2 j_3\}$ where $\{0 j_1, 0 j_2, 1 j_3\} \in \mathcal{B}'$
	$\{i_2 j_1, i_2 j_2, i_1 j_3\}$ where $\{1 j_1, 1 j_2, 0 j_3\} \in \mathcal{B}'$
	$\{i_2 j_1, i_2 j_2, i_2 j_3\}$ where $\{1 j_1, 1 j_2, 1 j_3\} \in \mathcal{B}'$

Table 30 Constructing a $STS(12n+1)$ from a LJX_1 in which $\rho^2 = \emptyset$

$LJX_2(n)$	$STS(3n)$
	$\{1i, \mathtt{a}i, \mathtt{b}i\}$
$(x_0, x_1, x_2) \in \mathcal{L}_i$	$\{1x_0, \mathtt{a}x_1, \mathtt{b}x_2\}$
	$\{1i_0, 1i_1, 1i_2\}$ where $\{i_0, i_1, i_2\} \in \mathcal{B}_i$
	$\{\mathtt{a}i_0, \mathtt{a}i_1, \mathtt{a}i_2\}$ where $\{i_0, i_1, i_2\} \in \mathcal{B}_i$
	$\{\mathtt{b}i_0, \mathtt{b}i_1, \mathtt{b}i_2\}$ where $\{i_0, i_1, i_2\} \in \mathcal{B}_i$
	$\{1i, \mathtt{a}i_1, \mathtt{a}i_2\}$ where $\{\infty_1, i_1, i_2\} \in \mathcal{B}_i$
	$\{\mathtt{a}i, \mathtt{b}i_1, \mathtt{b}i_2\}$ where $\{\infty_1, i_1, i_2\} \in \mathcal{B}_i$
	$\{\mathtt{b}i, 1i_1, 1i_2\}$ where $\{\infty_1, i_1, i_2\} \in \mathcal{B}_i$
	$\{1i, \mathtt{b}i_1, \mathtt{b}i_2\}$ where $\{\infty_2, i_1, i_2\} \in \mathcal{B}_i$
	$\{\mathtt{a}i, 1i_1, 1i_2\}$ where $\{\infty_2, i_1, i_2\} \in \mathcal{B}_i$
	$\{\mathtt{b}i, \mathtt{a}i_1, \mathtt{a}i_2\}$ where $\{\infty_2, i_1, i_2\} \in \mathcal{B}_i$

Table 31 Constructing a $STS(3n)$ from a $LJX_2(n)$ using $\mathcal{L}_i$

$LJX_2(n)$	$STS(3n)$
$(x_0, x_1, x_2, i) \in \mathcal{L}^1$	$\{1x_0, \mathtt{a}x_1, \mathtt{b}x_2\}$
	$\{1x_0, \mathtt{a}i_1, \mathtt{a}i_2\}$ where $\{\infty_1, i_1, i_2\} \in \mathcal{B}_{x_1}$
	$\{\mathtt{a}x_0, \mathtt{b}i_1, \mathtt{b}i_2\}$ where $\{\infty_1, i_1, i_2\} \in \mathcal{B}_{x_2}$
	$\{\mathtt{b}x_0, 1i_1, 1i_2\}$ where $\{\infty_1, i_1, i_2\} \in \mathcal{B}_{x_0}$

Table 32 Constructing a $STS(3n)$ from a $LJX_2(n)$ using $\mathcal{L}^1$

$LJX_2(n)$	$STS(3n)$
$(x_0, x_1, x_2, i) \in \mathcal{L}^2$	$\{1x_0, \mathtt{a}x_1, \mathtt{b}x_2\}$
	$\{1x_0, \mathtt{b}i_1, \mathtt{b}i_2\}$ where $\{\infty_2, i_1, i_2\} \in \mathcal{B}_{x_2}$
	$\{\mathtt{a}x_0, 1i_1, 1i_2\}$ where $\{\infty_2, i_1, i_2\} \in \mathcal{B}_{x_0}$
	$\{\mathtt{b}x_0, \mathtt{a}i_1, \mathtt{a}i_2\}$ where $\{\infty_2, i_1, i_2\} \in \mathcal{B}_{x_1}$

Table 33 Constructing a $STS(3n)$ from a $LJX_2(n)$ using $\mathcal{L}^2$

$LJX_1^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{01}^j$	$\{x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2\}$ where $y_0 + y_1 + y_2 = y$; $z_0 + z_1 + z_2 = z$ $w_0 + w_1 + w_2 = w$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{\infty, x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1\}$ where $\{0\,z\,w, 1\,z_0 w_0, \mathtt{a}\,z_1 w_1, \mathtt{b}\,z_2 w_2\} \in \mathcal{A}_2$ $y_0 = y$; $y_1 = y + z_2 + 1$
	$\{\infty, x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2\}$ where $\{0\,z\,w, 1\,z_0 w_0, \mathtt{a}\,z_1 w_1, \mathtt{b}\,z_2 w_2\} \in \mathcal{A}_2$ $y_1 = y + z_2$; $y_2 = y + 1$
	$\{\infty, x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0\}$ where $\{0\,z\,w, 1\,z_0 w_0, \mathtt{a}\,z_1 w_1, \mathtt{b}\,z_2 w_2\} \in \mathcal{A}_2$ $y_2 = y$; $y_0 = y + 1$
$(x_0, x_1, x_2) \in \beta_{11}^j$	$\{\infty, x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1\}$ where $\{0\,z\,w, 1\,z_0 w_0, \mathtt{a}\,z_1 w_1, \mathtt{b}\,z_2' w_2\} \in \mathcal{A}_2$ $y_0 = y + 1$; $y_1 = y + z_2$ $z_2' = z_2 + 1$
	$\{\infty, x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2\}$ where $\{0\,z\,w, 1\,z_0 w_0, \mathtt{a}\,z_1 w_1, \mathtt{b}\,z_2' w_2\} \in \mathcal{A}_2$ $y_1 = y + z_2 + 1$; $y_2 = y + 1$ $z_2' = z_2 + 1$
	$\{\infty, x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0\}$ where $\{0\,z\,w, 1\,z_0 w_0, \mathtt{a}\,z_1 w_1, \mathtt{b}\,z_2' w_2\} \in \mathcal{A}_2$ $y_2 = y$; $y_0 = y$ $z_2' = z_2 + 1$

Table 34 Constructing a $STS(12mn+1)$ from a $LJX_3^j(m)$

$LJX_1^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1, x_1 y_3 z_3 w_3\}$ where $\{y_1 z_1 w_1, y_3 z_3 w_3\} \in H_{y'z'w'}[GF[2] \times \mathbb{Z}_n]$ $\{0 z w, 1 z_0 w_0, \mathtt{a} z' w', \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_0 = y + y'$
	$\{x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2, x_2 y_3 z_3 w_3\}$ where $\{y_2 z_2 w_2, y_3 z_3 w_3\} \in H_{y'z'w'}[GF[2] \times \mathbb{Z}_n]$ $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z' w'\} \in \mathcal{A}_2$ $y_1 = y + y' + z'$
	$\{x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0, x_0 y_3 z_3 w_3\}$ where $\{y_0 z_0 w_0, y_3 z_3 w_3\} \in H_{y'z'w'}[GF[2] \times \mathbb{Z}_n]$ $\{0 z w, 1 z' w', \mathtt{a} z_1 w_1, \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_2 = y + y'$
$(x_0, x_1, x_2) \in \beta_{11}^j$	$\{x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1, x_1 y_3 z_3 w_3\}$ where $\{y_1 z_1 w_1, y_3 z_3 w_3\} \in H_{y'z'w'}[GF[2] \times \mathbb{Z}_n]$ $\{0 z w, 1 z_0' w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_0 = y + y' + 1$; $z_0' = z_0 + 1$
	$\{x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2, x_2 y_3 z_3 w_3\}$ where $\{y_2 z_2 w_2, y_3 z_3 w_3\} \in H_{y'z'w'}[GF[2] \times \mathbb{Z}_n]$ $\{0 z w, 1 z_0' w_0, \mathtt{a} z_1 w_1, \mathtt{b} z' w'\} \in \mathcal{A}_2$ $y_1 = y + y' + z_0$; $z_0' = z_0 + 1$
	$\{x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0, x_0 y_3 z_3 w_3\}$ where $\{y_0 z_0 w_0, y_3 z_3 w_3\} \in H_{y'z'w'}[GF[2] \times \mathbb{Z}_n]$ $\{0 z w, 1 z_0' w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_2 = y + y'$; $z_0' = z' + 1$

Table 34 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_3^j(m)$

$LJX_1^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2\}$ where $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_0 = y$; $y_1 = y + z_2$ $y_2 = y$
	$\{x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0\}$ where $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_1 = y + z_2$; $y_2 = y$ $y_0 = y$
	$\{x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1\}$ where $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2 w_2\} \in \mathcal{A}_2$ $y_2 = y$; $y_0 = y$ $y_1 = y + z_2$
$(x_0, x_1, x_2) \in \beta_{11}^j$	$\{x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2\}$ where $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2' w_2\} \in \mathcal{A}_2$ $y_0 = y$; $y_1 = y + z_2$ $y_2 = y + 1$; $z_2' = z_2 + 1$
	$\{x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0\}$ where $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2' w_2\} \in \mathcal{A}_2$ $y_1 = y + z_2$; $y_2 = y + 1$ $y_0 = y$; $z_2' = z_2 + 1$
	$\{x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1\}$ where $\{0 z w, 1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2' w_2\} \in \mathcal{A}_2$ $y_2 = y + 1$; $y_0 = y$ $y_1 = y + z_2$; $z_2' = z_2 + 1$

Table 34 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_3^j(m)$

$LJX_1^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2, x_3) \in \beta_{20}^j$	$\{x_0y_0z_0w_0, x_1y_1z_1w_1, x_2y_2z_2w_2\}$ where $y_0+y_1=y$; $y_0+y_2=0$ $z_0+z_1+z_2=z$ $w_0+w_1+w_2=w$
	$\{x_0y_0z_0w_0, x_1y_1z_1w_1, x_3y_3z_3w_3\}$ where $y_0+y_1=y+1$ $y_0+y_3=y$ $z_0+z_1+z_3=z$ $w_0+w_1+w_3=w$
	$\{x_2y_2z_2w_2, x_0y_0z_0w_0, x_3y_3z_3w_3\}$ where $y_2+y_0=1$ $y_0+y_3=y+1$ $z_2+z_0+z_3=z$ $w_2+w_0+w_3=w$
	$\{x_1y_1z_1w_1, x_2y_2z_2w_2, x_3y_3z_3w_3\}$ where $y_1+y_2=y+1$ $y_1+y_3=0$ $z_0+z_1+z_2=z$ $w_0+w_1+w_2=w$
$(x_0, x_1, x_2, x_3) \in \beta_{21}^j$	$\{x_0y_0z_0w_0, x_1y_1z_1w_1, x_2y_2z_2w_2\}$ where $y_0+y_1=y$; $y_0+y_2=1$ $z_0+z_1+z_2=z$ $w_0+w_1+w_2=w$
	$\{x_0y_0z_0w_0, x_1y_1z_1w_1, x_3y_3z_3w_3\}$ where $y_0+y_1=y+1$ $y_0+y_3=y+1$ $z_0+z_1+z_3=z$ $w_0+w_1+w_3=w$

Table 34 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_3^j(m)$

$LJX_1^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2, x_3) \in \beta_{21}^j$	$\{x_2 y_2 z_2 w_2, x_0 y_0 z_0 w_0, x_3 y_3 z_3 w_3\}$ where $y_2 + y_0 = 0$; $y_0 + y_3 = y$ $z_2 + z_0 + z_3 = z$ $w_2 + w_0 + w_3 = w$
	$\{x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2, x_3 y_3 z_3 w_3\}$ where $y_1 + y_2 = y$; $y_1 + y_3 = 1$ $z_0 + z_1 + z_2 = z$ $w_0 + w_1 + w_2 = w$

Table 34 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_3^j(m)$

$LS_1(LJX_1(m))$	$STS(12mn+1)$
	$\{\infty, x y_0 z_0 w_0, x y_1 z_1 w_1\}$ where $x \in X$ $\{\infty, y_0 z_0 w_0, y_1 z_1 w_1\} \in \beta$
	$\{x y_0 z_0 w_0, x y_1 z_1 w_1, x y_2 z_2 w_2\}$ where $x \in X$ $\{y_0 z_0 w_0, y_1 z_1 w_1, y_2 z_2 w_2\} \in \beta$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-1}} \beta_{10}^j$	$\{x_0 y_0 z_0 w_0, x_1 y_1 z_1 w_1, x_2 y_2 z_2 w_2\}$ where $\{1 z_0 w_0, \mathtt{a} z_1 w_1, \mathtt{b} z_2' w_2', \cdot\} \in \mathcal{A}_2$ $z_2 = y_0 + y_1 + z$ $z_2' = y + y_1 + y_2$ $w_2' = w + w_2$

Table 35 Constructing a $STS(12mn+1)$ from a $LS_1(LJX_3(m))$

$LJX_1^j(m)$	$STS(6mn+1)$
$(x_0, x_1, x_2) \in \beta_{01}^j$	$\{x_0 y_0 w_0, x_1 y_1 w_1, x_2 y_2 w_2\}$ where $y_0 + y_1 + y_2 = y$; $w_0 + w_1 + w_2 = w$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{\infty, x_0 y_0 w_0, x_1 y_1 w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $y_0 = \phi_{0B}(x_0) + y$; $y_1 = \phi_{0B}(x_1) + y + 1$
	$\{\infty, x_1 y_1 w_1, x_2 y_2 w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $y_1 = \phi_{0B}(x_1) + y$; $y_2 = \phi_{0B}(x_2) + y + 1$
	$\{\infty, x_2 y_2 w_2, x_0 y_0 w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $y_2 = \phi_{0B}(x_2) + y$; $y_0 = \phi_{0B}(x_0) + y + 1$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{x_0 y_0 w_0, x_1 y_1 w_1, x_1 y_2 w_2\}$ where $\{y_1 w_1, y_2 w_2\} \in H_{y'w'}[\mathbb{Z}_n]$ $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $y_0 = \phi_{0B}(x_0) + y + y'$
	$\{x_1 y_1 w_1, x_2 y_2 w_2, x_2 y_0 w_0\}$ where $\{y_2 w_2, y_0 w_0\} \in H_{y'w'}[\mathbb{Z}_n]$ $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $y_1 = \phi_{0B}(x_1) + y + y'$
	$\{x_2 y_2 w_2, x_0 y_0 w_0, x_0 y_1 w_1\}$ where $\{y_0 w_0, y_1 w_1\} \in H_{y'w'}[\mathbb{Z}_n]$ $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $y_2 = \phi_{0B}(x_2) + y + y'$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{x_0 y_0 w_0, x_1 y_1 w_1, x_2 y_2 w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $y_0 = \phi_{0B}(x_0) + y$; $y_1 = \phi_{0B}(x_1) + y$ $y_2 = \phi_{0B}(x_2) + y$

Table 36 Constructing a $STS(6mn+1)$ from a $LJX_3^j(m)$

$LJX_1^j(m)$	$STS(6mn+1)$
$(x_0, x_1, x_2, x_3) \in \beta_{20}$	$\{x_0y_0w_0, x_1y_1w_1, x_2y_2w_2\}$ where $y_0+y_1=y$; $y_0+y_2=0$ $w_0+w_1+w_2=w$
	$\{x_0y_0w_0, x_1y_1w_1, x_3y_3w_3\}$ where $y_0+y_1=y+1$; $y_0+y_3=y$ $w_0+w_1+w_3=w$
	$\{x_2y_2w_2, x_0y_0w_0, x_3y_3w_3\}$ where $y_2+y_0=1$; $y_0+y_3=y+1$ $w_2+w_0+w_3=w$
	$\{x_1y_1w_1, x_2y_2w_2, x_3y_3w_3\}$ where $y_1+y_2=y+1$; $y_1+y_3=0$ $w_0+w_1+w_2=w$
$(x_0, x_1, x_2, x_3) \in \beta_{21}$	$\{x_0y_0w_0, x_1y_1w_1, x_2y_2w_2\}$ where $y_0+y_1=y$; $y_0+y_2=1$ $w_0+w_1+w_2=w$
	$\{x_0y_0w_0, x_1y_1w_1, x_3y_3w_3\}$ where $y_0+y_1=y+1$; $y_0+y_3=y+1$ $w_0+w_1+w_3=w$
	$\{x_2y_2w_2, x_0y_0w_0, x_3y_3w_3\}$ where $y_2+y_0=0$; $y_0+y_3=y$ $w_2+w_0+w_3=w$
	$\{x_1y_1w_1, x_2y_2w_2, x_3y_3w_3\}$ where $y_1+y_2=y$; $y_1+y_3=1$ $w_0+w_1+w_2=w$

Table 36 (cont'd) Constructing a $STS(6mn+1)$ from a $LJX_3^j(m)$

$LS_2(LJX_1(m))$	$STS(6mn+1)$
	$\{\infty, x_0y_0w_0, x_1y_1w_1\}$ where $\{x_0, x_1\} \subseteq B \in H$ $\{\infty, 0y_0w_0, 1y_1w_1\} \in \beta$
	$\{x_0y_0w_0, x_1y_1w_1, x_1y_2w_2\}$ where $\{x_0, x_1\} \subseteq B \in H$ $\{0y_0w_0, 1y_1w_1, 1y_2w_2\} \in \beta$
	$\{x_0y_0w_0, x_0y_1w_1, x_1y_2w_2\}$ where $\{x_0, x_1\} \subseteq B \in H$ $\{0y_0w_0, 0y_1w_1, 1y_2w_2\} \in \beta$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-1}} \beta_{10}^j$	$\{x_0y_0w_0, x_1y_1w_1, x_2y_2w_2\}$ where $\{1w_0, \mathtt{a}w_1, \mathtt{b}w_2', \cdot\} \in \mathcal{A}_1$ $y_0 = \phi_{uB}(x_0)$ $y_1 = \phi_{uB}(x_1)$ $y_2 = \phi_{uB}(x_2)$ $w_2' = w + w_2$
	$\{x_0y_0w_0, x_1y_1w_1, x_2y_2w_2\}$ where $\{1w_0, \mathtt{a}w_1, \mathtt{b}w_2', \cdot\} \in \mathcal{A}_1$ $y_0 = \phi_{uB}(x_0) + 1$ $y_1 = \phi_{uB}(x_1) + 1$ $y_2 = \phi_{uB}(x_2) + 1$ $w_2' = w + w_2$

Table 37 Constructing a $STS(6mn+1)$ from a $LS_2(LJX_3(m))$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{10}^j$	$\{\infty, x_0u_0w_0, x_1u_1w_1\}$ where $\{\infty, \mathtt{1}u_0w_0, \mathtt{a}u_1w_1\} \in \beta$
	$\{\infty, x_1u_1w_1, x_2u_2w_2\}$ where $\{\infty, \mathtt{a}u_1w_1, \mathtt{b}u_2w_2\} \in \beta$
	$\{\infty, x_2u_2w_2, x_0u_0w_0\}$ where $\{\infty, \mathtt{b}u_2w_2, \mathtt{0}u_0w_0\} \in \beta$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{\mathtt{1}u_0w_0, \mathtt{a}u_1w_1, \mathtt{b}u_2w_2\} \in \beta$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $u_0+u_1+u_2=u$ $w_0+w_1+w_2=w$
$(x_0, x_1, x_2) \in \beta_{a0}^j$	$\{\infty, x_0u_0w_0, x_1u_1w_1\}$ where $\{\mathtt{0}w, \mathtt{1}w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0 = u$; $u_1 = u+1$
	$\{\infty, x_0u_0w_0, x_1u_1w_1\}$ where $\{\mathtt{0}w, \mathtt{1}w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0 = u+1$; $u_1 = u+\mathtt{a}$
	$\{\infty, x_1u_1w_1, x_2u_2w_2\}$ where $\{\mathtt{0}w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1 = u$; $u_2 = u+\mathtt{a}$
	$\{\infty, x_1u_1w_1, x_2u_2w_2\}$ where $\{\mathtt{0}w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1 = u+\mathtt{a}$; $u_2 = u+1$

Table 38 Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS\,(12mn+1)$
$(x_0\,,\,x_1\,,\,x_2) \in \beta_{a0}^j$	$\{\infty\,,\,x_2\,u_2\,w_2\,,\,x_0\,u_0\,w_0\}$ where $\{0\,w\,,\,\mathtt{b}\,w_2\,,\,1\,w_0\,,\,\cdot\} \in \mathcal{A}_1$ $u_2 = u\;;\;u_0 = u + \mathtt{b}$
	$\{\infty\,,\,x_2\,u_2\,w_2\,,\,x_0\,u_0\,w_0\}$ where $\{0\,w\,,\,\mathtt{b}\,w_2\,,\,1\,w_0\,,\,\cdot\} \in \mathcal{A}_1$ $u_2 = u + \mathtt{b}\;;\;u_0 = u + \mathtt{a}$
$(x_0\,,\,x_1\,,\,x_2) \in \beta_{a1}^j$	$\{\infty\,,\,x_0\,u_0\,w_0\,,\,x_1\,u_1\,w_1\}$ where $\{0\,w\,,\,1\,w_0\,,\,\mathtt{a}\,w_1\,,\,\cdot\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{b}\;;\;u_1 = u$
	$\{\infty\,,\,x_0\,u_0\,w_0\,,\,x_1\,u_1\,w_1\}$ where $\{0\,w\,,\,1\,w_0\,,\,\mathtt{a}\,w_1\,,\,\cdot\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{a}\;;\;u_1 = u + \mathtt{a}$
	$\{\infty\,,\,x_1\,u_1\,w_1\,,\,x_2\,u_2\,w_2\}$ where $\{0\,w\,,\,\mathtt{a}\,w_1\,,\,\mathtt{b}\,w_2\,,\,\cdot\} \in \mathcal{A}_1$ $u_1 = u + 1\;;\;u_2 = u$
	$\{\infty\,,\,x_1\,u_1\,w_1\,,\,x_2\,u_2\,w_2\}$ where $\{0\,w\,,\,\mathtt{a}\,w_1\,,\,\mathtt{b}\,w_2\,,\,\cdot\} \in \mathcal{A}_1$ $u_1 = u + \mathtt{b}\;;\;u_2 = u + \mathtt{b}$
	$\{\infty\,,\,x_2\,u_2\,w_2\,,\,x_0\,u_0\,w_0\}$ where $\{0\,w\,,\,\mathtt{b}\,w_2\,,\,1\,w_0\,,\,\cdot\} \in \mathcal{A}_1$ $u_2 = u + \mathtt{a}\;;\;u_0 = u$
	$\{\infty\,,\,x_2\,u_2\,w_2\,,\,x_0\,u_0\,w_0\}$ where $\{0\,w\,,\,\mathtt{b}\,w_2\,,\,1\,w_0\,,\,\cdot\} \in \mathcal{A}_1$ $u_2 = u + 1\;;\;u_0 = u + 1$

Table 38 (cont'd) Constructing a $STS\,(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{b1}^j$	$\{\infty, x_0u_0w_0, x_1u_1w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0 = u+1$; $u_1 = u+1$
	$\{\infty, x_0u_0w_0, x_1u_1w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0 = u$; $u_1 = u+\mathtt{b}$
	$\{\infty, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1 = u+\mathtt{a}$; $u_2 = u+\mathtt{a}$
	$\{\infty, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1 = u$; $u_2 = u+1$
	$\{\infty, x_2u_2w_2, x_0u_0w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_2 = u+\mathtt{b}$; $u_0 = u+\mathtt{b}$
	$\{\infty, x_2u_2w_2, x_0u_0w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_2 = u$; $u_0 = u+\mathtt{a}$
$(x_0, x_1, x_2) \in \beta_{a0}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_1u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w', \cdot\} \in \mathcal{A}_1$ $u_0+u_1+u_2 = u$ $w_1 \neq w_2$; $2w' = w_1+w_2$
	$\{x_1u_1w_1, x_2u_2w_2, x_2u_0w_0\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w', \cdot\} \in \mathcal{A}_1$ $u_1+u_2+u_0 = u$ $w_2 \neq w_0$; $2w' = w_2+w_0$

Table 38 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{a0}^j$	$\{x_2u_2w_2, x_0u_0w_0, x_0u_1w_1\}$ where $\{0w, \mathtt{b}w_2, 1w', \cdot\} \in \mathcal{A}_1$ $u_2+u_0+u_1=u$ $w_0 \neq w_1$; $2w' = w_0+w_1$
$(x_0, x_1, x_2) \in \beta_{a1}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_1u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w', \cdot\} \in \mathcal{A}_1$ $u_0+u_1+u_2=u$ $w_1 \neq w_2$; $2w' = w_1+w_2$
	$\{x_1u_1w_1, x_2u_2w_2, x_2u_0w_0\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w', \cdot\} \in \mathcal{A}_1$ $u_1+u_2+u_0=u$ $w_2 \neq w_0$; $2w' = w_2+w_0$
	$\{x_2u_2w_2, x_0u_0w_0, x_0u_1w_1\}$ where $\{0w, \mathtt{b}w_2, 1w', \cdot\} \in \mathcal{A}_1$ $u_2+u_0+u_1=u$ $w_0 \neq w_1$; $2w' = w_0+w_1$
$(x_0, x_1, x_2) \in \beta_{b1}^j$	$\{x_0u_0w_0, x_2u_2w_2, x_2u_1w_1\}$ where $\{0w, 1w_0, \mathtt{b}w', \cdot\} \in \mathcal{A}_1$ $u_0+u_2+u_1=u$ $w_2 \neq w_1$; $2w' = w_2+w_1$
	$\{x_1u_1w_1, x_0u_0w_0, x_0u_2w_2\}$ where $\{0w, \mathtt{a}w_1, 1w', \cdot\} \in \mathcal{A}_1$ $u_1+u_0+u_2=u$ $w_0 \neq w_2$; $2w' = w_0+w_2$
	$\{x_2u_2w_2, x_1u_1w_1, x_1u_0w_0\}$ where $\{0w, \mathtt{b}w_2, \mathtt{a}w', \cdot\} \in \mathcal{A}_1$ $u_2+u_1+u_0=u$ $w_1 \neq w_0$; $2w' = w_1+w_0$

Table 38 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{a0}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_1u_2w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0+u_1+u_2=u\ ;\ u_1 \neq u_2$ $\{u_1, u_2\} \neq \{u+\mathtt{b}, u+\mathtt{a}\}$
	$\{x_0u_0w_0, x_1u_1w_1, x_1u_2w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0=u$ $\{u_1, u_2\} = \{u+\mathtt{b}, u+\mathtt{a}\}$
	$\{x_1u_1w_1, x_2u_2w_2, x_2u_0w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1+u_2+u_0=u\ ;\ u_2 \neq u_0$ $\{u_2, u_0\} \neq \{u+1, u+\mathtt{b}\}$
	$\{x_1u_1w_1, x_2u_2w_2, x_2u_0w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1=u$ $\{u_2, u_0\} = \{u+1, u+\mathtt{b}\}$
	$\{x_2u_2w_2, x_0u_0w_0, x_0u_1w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_2+u_0+u_1=u\ ;\ u_0 \neq u_1$ $\{u_0, u_1\} \neq \{u+\mathtt{a}, u+1\}$
	$\{x_2u_2w_2, x_0u_0w_0, x_0u_1w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_2=u$ $\{u_0, u_1\} = \{u+\mathtt{a}, u+1\}$

Table 38 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{a1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_1 u_2 w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0 + u_1 + u_2 = u + \mathtt{b}$ $u_1 \neq u_2$ $\{u_1, u_2\} \neq \{u + \mathtt{b}, u + \mathtt{a}\}$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_1 u_2 w_1\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{b}$ $\{u_1, u_2\} = \{u + \mathtt{b}, u + \mathtt{a}\}$
	$\{x_1 u_1 w_1, x_2 u_2 w_2, x_2 u_0 w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1 + u_2 + u_0 = u + 1$ $u_2 \neq u_0$ $\{u_2, u_0\} \neq \{u + 1, u + \mathtt{b}\}$
	$\{x_1 u_1 w_1, x_2 u_2 w_2, x_2 u_0 w_2\}$ where $\{0w, \mathtt{a}w_1, \mathtt{b}w_2, \cdot\} \in \mathcal{A}_1$ $u_1 = u + 1$ $\{u_2, u_0\} = \{u + 1, u + \mathtt{b}\}$
	$\{x_2 u_2 w_2, x_0 u_0 w_0, x_0 u_1 w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_2 + u_0 + u_1 = u + \mathtt{a}$ $u_0 \neq u_1$ $\{u_0, u_1\} \neq \{u + \mathtt{a}, u + 1\}$
	$\{x_2 u_2 w_2, x_0 u_0 w_0, x_0 u_1 w_0\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_2 = u + \mathtt{a}$ $\{u_0, u_1\} = \{u + \mathtt{a}, u + 1\}$

Table 38 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{b1}^j$	$\{x_0u_0w_0, x_2u_2w_2, x_2u_1w_2\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_0 + u_2 + u_1 = u + \mathtt{b}$ $u_2 \neq u_1$ $\{u_2, u_1\} \neq \{u+1, u\}$
	$\{x_0u_0w_0, x_2u_2w_2, x_2u_1w_2\}$ where $\{0w, \mathtt{b}w_2, 1w_0, \cdot\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{b}$ $\{u_2, u_1\} = \{u+1, u\}$
	$\{x_1u_1w_1, x_0u_0w_0, x_0u_2w_0\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_1 + u_0 + u_2 = u + 1$ $u_0 \neq u_2$ $\{u_0, u_2\} \neq \{u + \mathtt{a}, u\}$
	$\{x_1u_1w_1, x_0u_0w_0, x_0u_2w_0\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_1 = u + 1$ $\{u_0, u_2\} = \{u + \mathtt{a}, u\}$
	$\{x_2u_2w_2, x_1u_1w_1, x_1u_0w_1\}$ where $\{0w, \mathtt{b}w_2, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_2 + u_1 + u_0 = u + \mathtt{a}$ $u_1 \neq u_0$ $\{u_1, u_0\} \neq \{u + \mathtt{b}, u\}$
	$\{x_2u_2w_2, x_1u_1w_1, x_1u_0w_1\}$ where $\{0w, \mathtt{b}w_2, \mathtt{a}w_1, \cdot\} \in \mathcal{A}_1$ $u_2 = u + \mathtt{a}$ $\{u_1, u_0\} = \{u + \mathtt{b}, u\}$

Table 38 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LJX_2^j(m)$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \beta_{a0}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $u_0 = u$; $u_1 = u$; $u_2 = u$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $u_0 = u + 1$; $u_1 = u + \mathtt{a}$ $u_2 = u + \mathtt{b}$
$(x_0, x_1, x_2) \in \beta_{a1}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{b}$; $u_1 = u + 1$ $u_2 = u + \mathtt{a}$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{a}$; $u_1 = u + \mathtt{b}$ $u_2 = u + 1$
$(x_0, x_1, x_2) \in \beta_{b1}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{b}$; $u_1 = u + 1$ $u_2 = u + \mathtt{a}$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{0w, 1w_0, \mathtt{a}w_1, \mathtt{b}w_2\} \in \mathcal{A}_1$ $u_0 = u + \mathtt{a}$; $u_1 = u + \mathtt{b}$ $u_2 = u + 1$

Table 38 (cont'd) Constructing a $STS(12mn+1)$ from a $LJX_4^j(m)$

$LS_3(LJX_2(m))$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \bigcup_{j\in I_{3m-2}} \beta_{10}^j$	$\{\infty, x_0u_0w_0, x_1u_1w_1\}$ where $\{\infty, 1u_0w_0, \mathtt{a}u_1w_1\} \in \beta$
	$\{\infty, x_0u_0w_0, x_2u_2w_2\}$ where $\{\infty, 1u_0w_0, \mathtt{b}u_2w_2\} \in \beta$
	$\{\infty, x_1u_1w_1, x_2u_2w_2\}$ where $\{\infty, \mathtt{a}u_1w_1, \mathtt{b}u_2w_2\} \in \beta$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{1u_0w_0, \mathtt{a}u_1w_1, \mathtt{b}u_2w_2\} \in \beta$
$(x_0, x_1, x_2) \in \bigcup_{j\in I_{3m-2}} \beta_{a0}^j$	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{1w_0, \mathtt{a}w_1, \mathtt{b}w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y$; $u_1 = 1y$ $u_2 = \mathtt{a}y + u$ $w_2' = w + w_2$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{1w_0, \mathtt{a}w_1, \mathtt{b}w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + 1$; $u_1 = 1y + 1$ $u_2 = \mathtt{a}y + u + 1$ $w_2' = w + w_2$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{1w_0, \mathtt{a}w_1, \mathtt{b}w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{a}$; $u_1 = 1y + \mathtt{a}$ $u_2 = \mathtt{a}y + u + \mathtt{a}$ $w_2' = w + w_2$
	$\{x_0u_0w_0, x_1u_1w_1, x_2u_2w_2\}$ where $\{1w_0, \mathtt{a}w_1, \mathtt{b}w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{b}$; $u_1 = 1y + \mathtt{b}$ $u_2 = \mathtt{a}y + u + \mathtt{b}$ $w_2' = w + w_2$

Table 39 Constructing a $STS(12mn+1)$ from a $LS_3(LJX_4(m))$

$LS_3(LJX_2(m))$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{a0}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + 1$; $u_1 = 1y + \mathtt{a}$ $u_2 = \mathtt{a}y + u + \mathtt{b}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y$; $u_1 = 1y + \mathtt{b}$ $u_2 = \mathtt{a}y + u + \mathtt{a}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{b}$; $u_1 = 1y$ $u_2 = \mathtt{a}y + u + 1$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{a}$; $u_1 = 1y + 1$ $u_2 = \mathtt{a}y + u$; $w_2' = w + w_2$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{a1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{b}$; $u_1 = 1y + 1$ $u_2 = \mathtt{a}y + u + \mathtt{a}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{a}$; $u_1 = 1y$ $u_2 = \mathtt{a}y + u + \mathtt{b}$ $w_2' = w + w_2$

Table 39 (cont'd) Constructing a $STS(12mn+1)$ from a $LS_3(LJX_4(m))$

$LS_3(LJX_2(m))$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{a1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{1}$; $u_1 = \mathtt{1}y + \mathtt{b}$ $u_2 = \mathtt{a}y + u$; $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y$; $u_1 = \mathtt{1}y + \mathtt{a}$ $u_2 = \mathtt{a}y + u + \mathtt{1}$ $w_2' = w + w_2$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{a1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{a}$; $u_1 = \mathtt{1}y + \mathtt{b}$ $u_2 = \mathtt{a}y + u + \mathtt{1}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{b}$; $u_1 = \mathtt{1}y + \mathtt{a}$ $u_2 = \mathtt{a}y + u$; $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y$; $u_1 = \mathtt{1}y + \mathtt{1}$ $u_2 = \mathtt{a}y + u + \mathtt{b}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{1}$; $u_1 = \mathtt{1}y$ $u_2 = \mathtt{a}y + u + \mathtt{a}$ $w_2' = w + w_2$

Table 39 (cont'd) Constructing a $STS(12mn+1)$ from a $LS_3(LJX_4(m))$

$LS_3(LJX_2(m))$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{b1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{b}$; $u_1 = 1y + 1$ $u_2 = \mathtt{a}y + u + \mathtt{a}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{a}$; $u_1 = 1y$ $u_2 = \mathtt{a}y + u + \mathtt{b}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + 1$; $u_1 = 1y + \mathtt{b}$ $u_2 = \mathtt{a}y + u$; $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y$; $u_1 = 1y + \mathtt{a}$ $u_2 = \mathtt{a}y + u + 1$ $w_2' = w + w_2$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{b1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{a}$; $u_1 = 1y + \mathtt{b}$ $u_2 = \mathtt{a}y + u + 1$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{1 w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{b}$; $u_1 = 1y + \mathtt{a}$ $u_2 = \mathtt{a}y + u$; $w_2' = w + w_2$

Table 39 (cont'd) Constructing a $STS(12mn+1)$ from a $LS_3(LJX_4(m))$

$LS_3(LJX_2(m))$	$STS(12mn+1)$
$(x_0, x_1, x_2) \in \bigcup_{j \in I_{3m-2}} \beta_{b1}^j$	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{\mathtt{1} w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y$; $u_1 = \mathtt{1}y + \mathtt{1}$ $u_2 = \mathtt{a}y + u + \mathtt{b}$ $w_2' = w + w_2$
	$\{x_0 u_0 w_0, x_1 u_1 w_1, x_2 u_2 w_2\}$ where $\{\mathtt{1} w_0, \mathtt{a} w_1, \mathtt{b} w_2', \cdot\} \in \mathcal{A}_1$ $u_0 = \mathtt{b}y + \mathtt{1}$; $u_1 = \mathtt{1}y$ $u_2 = \mathtt{a}y + u + \mathtt{a}$ $w_2' = w + w_2$

Table 39 (cont'd) Constructing a $STS(12mn+1)$ from a $LS_3(LJX_4(m))$

REFERENCES

[1] R. C. Bose, On the construction of balanced incomplete block designs, *Ann. Eugenics* 9 (1939), 353–399.

[2] A. Cayley, On the triadic arrangements of seven and fifteen things, *London, Edinburgh & Dublin Phil. Mag. J. Sci.* 37 (1850), 50–53; *Collected Mathematical Papers* I, 481–484.

[3] R. H. F. Denniston, Some packings with Steiner triple systems, *Discrete Math.* 9 (1973), 213–227.

[4] Jean Doyen, Constructions of disjoint Steiner triple systems, *Proc. Amer. Math. Soc.* 32 (1972), 409–416.

[5] Haim Hanani, On quadruple systems, *Canadian J. Math.* 12 (1960), 145–157.

[6] Haim Hanani, D. K. Ray-Chaudhuri and Richard M. Wilson, On resolvable designs, *Discrete Math.* 3 (1972), 343–357.

[7] Thomas P. Kirkman, On a problem in combinations, *Cambridge and Dublin Math. J.* 2 (1847), 191–204.

[8] Lu Jia-Xi, On large sets of disjoint Steiner triple systems I, *J. Combinatorial Theory* 34A (1983), 140–146.

[9] Lu Jia-Xi, On large sets of disjoint Steinner triple systems II, *J. Combinatorial Theory* 34A (1983), 147–155.

[10] Lu Jia-Xi, On large sets of disjoint Steiner triple systems III, *J. Combinatorial Theory* 34A (1983), 156–182.

[11] Lu Jia-Xi, On large sets of disjoint Steiner triple systems IV, *J. Combinatorial Theory* 37A (1984), 136–163.

[12] Lu Jia-Xi, On large sets of disjoint Steiner triple systems V, *J. Combinatorial Theory* 37A (1984), 164–188.

[13] Lu Jia-Xi, On large sets of disjoint Steiner triple systems VI, *J. Combinatorial Theory* 37A (1984), 189–192.

[14] Lu Jia-Xi, On large sets of disjoint Steiner triple systems VII, unfinished manuscript.

[15] Eric Mendelsohn and Alexander Rosa, One-factorizations of the complete graph — a survey, *J. Graph Theory* 9 (1985), 43–65.

[16] D. K. Ray–Chaudhuri and Richard M. Wilson, Solution of Kirkman's school girl problem, *Proceedings of Symposia in Pure Mathematics*, American Mathematical Society, Providence, Rhode Island, 19 (1971), 187–203.

[17] Alexander Rosa, A theorem on the maximum number of disjoint Steiner triple systems, *J. Combinatorial Theory* 18A (1975), 305–312.

[18] Martin J. Sharry, *Large and overlarge sets of block designs*, Ph. D. thesis, The University of Queensland, Queensland, Australia, 1992.

[19] D. R. Stinson, A short proof of the non–existence of a pair of orthogonal Latin squares of order six, *J. Combinatorial Theory* 36A (1984), 373–376.

[20] Anne Penfold Street and Deborah J. Street, *Combinatorics of Experimental Design*, Clarendon Press, Oxford, New York (1987).

[21] Luc Teirlinck, On the maximum number of disjoint Steiner triple systems, *Discrete Math.* 6 (1973), 299–300.

[22] Luc Teirlinck, A completion of Lu's determination of the spectrum for large sets of disjoint Steiner triple systems, *J. Combinatorial Theory* 57A (1991), 302–305.

11

$(22, 33, 12, 8, 4)$-BIBD, AN UPDATE

G. H. J. van Rees

Department of Computer Science,
University of Manitoba,
Winnipeg, Manitoba, Canada R3T 2N2
vanrees@cs.umanitoba.ca

IN MEMORY OF MARSHALL HALL JR.

ABSTRACT

A (v, b, r, k, λ)-balanced incomplete block design is a family of b sets (called blocks) of size k whose elements (varieties) are from a v-set, $v > k$, such that every element occurs exactly r times and every pair exactly λ times. A $(22, 33, 12, 8, 4)$-BIBD is the set of parameters with the smallest v for which it is not known whether a BIBD exists or not. A survey of what is known about such a design is given.

1 INTRODUCTION

A (v, b, r, k, λ)-balanced incomplete block design is a family of b sets (called blocks) of size k whose elements (varieties) are from a v-set, $k < v$, such that every element occurs exactly r times and every pair exactly λ times. The (v, b, r, k, λ) are called the parameters of the design. Designs are usually thought of as a list of blocks but can also be thought of when programming or for theoretical work as an incidence matrix. This is a v by b matrix, A, whose rows are indexed by the element names, usually 1 to v, and whose columns are indexed by the block names, usually 1 to b, where element $a_{i,j}$ is 1 if element i is in block j and is 0 otherwise. Also useful is the matrix S, called the block structure matrix, that is defined to be equal to the matrix $A^t A$.

W. D. Wallis (ed.), Computational and Constructive Design Theory, 337–357.
© 1996 *Kluwer Academic Publishers.*

Statisticians in the thirties needed BIBDs to design experiments in which computations did not get too profuse. Even today, BIBDs are still useful to them. Today there are many other applications, eg. coding theory, cryptography and pure mathematics. Fisher and Yates [4] in their thirties Statistical tables listed all known small BIBDs. Their smallest undecided case was for the parameters (22,33,12,8,4) and it is still undecided today. Theory developed for it will apply to all BIBDs.

There are four well-known necessary conditions for the existence of a BIBD:

1. $rv = bk$
2. $\lambda(v-1) = r(k=1)$.
3. Fisher's Inequality: $b \geq v$
4. Bruck-Ryser-Chowla Condition: If $v = b$, then
   (a) if v is even, $k - \lambda$ is a square.
   (b) if v is odd, $z^2 = (k-\lambda)\,x^2 + (-1)^{(v-1)/2}\lambda y^2$ has a solution in integers x, y, z not all equal to 0.

However these are not sufficient conditions. Mathon and Rosa [12] have published tables of all the small parameter sets which obey the first three necessary conditions. Their smallest (in v) undecided parameter set is the same one that Fisher and Yates had 40 years before.

Over the last ten years, much effort and computation has gone into settling this question. This work will be reviewed including the unpublished work of Greig. Greig's work allows us to prove a lot of the early brute force counting arguments in a much more concise and systematic way. The group theory approach is reviewed. The results from coding theory used in two different ways are explained. Finally, a new approach by Van Rees is described and new exciting computer results from McKay and Radziszowski are announced.

2 COUNTING

The first paper specifically on the parameter set (22,33,12,8,4) appeared in JCT in 1978 by Hamada and Kobayashi [8]. First, they examined the block

intersection numbers. They gave the blocks an arbitrary numbering from 1 to b and defined a_i as the number of blocks that intersect block B_1 in i varieties with $i < k$. Using simple counting arguments, they obtained the following three equations:

$$\sum_{i=0}^{k} a_i = b - 1 = 32 \tag{11.1}$$

$$\sum_{i=0}^{k} i a_i = k(r-1) = 88 \tag{11.2}$$

$$\sum_{i=0}^{k} \binom{i}{2} a_i = \binom{k}{2} (\lambda - 1) = 84 \tag{11.3}$$

Solving these equations, they obtained nine solutions. They reduced these nine solutions to the four solutions in Table 1. They eliminated the other solutions with elementary but ad hoc counting arguments. These solutions are often called block intersection patterns. None of these patterns have been eliminated in the last 25 years. We record the result in Theorem 1.

Theorem 1 *The 4 possible block intersection patterns in Table 1 are the only possible block intersection patterns for a BIBD with parameters (22,33,12,8,4).*

Table 1 Intersection Patterns

TYPE	n_0	n_1	n_2	n_3	n_4
1	0	0	12	16	4
2	0	1	9	19	3
3	0	2	6	22	2
4	1	0	6	24	1

The next important paper to appear was the 4-author paper, *On Designs (22,33,12,8,4)* by Hall, Roth, van Rees, Vanstone [7]. This paper and the previous paper both used Connor's Criterion to good effect. If we let I be the identity matrix, J be all the 1's matrix and let $S = A^t A$, then first part of Connor's Criterion can be stated as the following:

Theorem 2 *The matrix $C = r(r-\lambda)I + \lambda k J - rS$ is positive semi-definite.*

We do not give the proof of this here but instead will skip a few years ahead to give the proof of the generalized Connor Criterion given by Greig [5]. Unfortunately, it has not been published but some design theorists have copies of his preprint. He gave an integer form of the Connor Criterion. This theorem has wide application in design theory including our problem.

His basic idea is that if y is a real column vector of length p then $y^t(y - 1_b) \geq -p/4$ and this leads to Connor's original result. But if y is an integral vector then $y^t(y - 1_p) \geq 0$ and this leads to the following general theorem.

Theorem 3 *Let B be a p by q matrix, let x be a q by 1 vector, let j be a p by 1 vector of all 1's and let d be a scalar then if $y = Bx - JBx/p + \left(\frac{1}{2} - d\right)j$ then $x^tB^tBx - x^tB^tJBx/p - p\left(\frac{1}{4} - d^2\right) \geq 0$ if d is chosen to make y integral.*

Proof Since d has been chosen to make y integral, $y^t(y - 1_p) >= 0$. So

$$\begin{aligned} y^t(y - 1_p) &= \left(Bx - JBx/p + \left(\frac{1}{2} - d\right)j\right)^t \left(Bx - JBx/p + \left(-\frac{1}{2} - d\right)j\right) \\ &= x^tB^tBx - x^tB^tJBx/p - \left(\frac{1}{2} + d\right)x^tB^tj - x^tB^tJBx/p \\ &\quad + x^tB^tJ^2Bx/p^2 + \left(\frac{1}{2} + d\right)x^tB^tJj/p + \left(\frac{1}{2} - d\right)j^tBx \\ &\quad - \left(\frac{1}{2} - d\right)j^tJBx/p - \left(\frac{1}{4} - d^2\right)j^tj \\ &= x^tB^tBx - x^tB^tJBx/p - p\left(\frac{1}{4} - d^2\right) \geq 0. \end{aligned}$$

□

If y has only t integral coordinates, then we get Theorem 4.

Theorem 4 *If the conditions of Theorem 3 hold but d is chosen to make only t coordinates of y integral then $x^tB^tBx - x^tB^tJBx/p - p\left(\frac{1}{4} - d^2\right) \geq \frac{1}{4}(t - p)$.*

Both of these theorems are specializations of one of Greig's theorems. Now, by choosing three appropriate B matrices, Greig establishes three useful corollaries that we will then use to easily prove much of the hard work of the first two papers mentioned.

In Theorem 3, let $B = ((r - \lambda) I_b - S)$ to get the following corollary. It is the integer version of the Connor Criterion.

Corollary 5 *If S is the block structure matrix of a $BIBD(v, b, r, k, \lambda)$ and if $((r - \lambda) I_b - S) x$ is integral and d is chosen to make $-\lambda v j^t x / b - \left(\frac{1}{2} - d\right)$ integral then $x^t ((r - \lambda) I_b - S) x \geq b \left(\frac{1}{4} - d^2\right) / (r - \lambda) - \lambda v x^t J x / b$.*

Let $B = S$ in Theorem 3 to get the following corollary.

Corollary 6 *If S is the block structure matrix of a $BIBD(v, b, r, k, \lambda)$ and if Sx is integral and d is chosen such that $k^2 x^t 1_b / v - \frac{1}{2} + d$ is integral then $x^t S x \geq k^2 x^t J x / v + b \left(\frac{1}{4} - d^2\right) / (r - \lambda)$.*

Finally, let $B = A$ in Theorem 3 to get this last corollary. It depends on circumstances which of Corollary 6 or Corollary 7 is the stronger.

Corollary 7 *If A is the incidence matrix of a $BIBD(v, b, r, k, \lambda)$, and Ax is integral and d is chosen such that $k x^t j / v - \frac{1}{2} + d$ is integral then $x^t S x \geq k^2 x^t J x / v + v \left(\frac{1}{4} - d^2\right)$.*

With these three corollaries, Greig easily reproved some of the results from the previous two papers. The notation $0 * 31$ means a list of thirty-one 0's. The first result is two lemmas in [8].

Theorem 8 *There are no block intersections of size larger than 4.*

Proof Let s be the block intersection of block 1 and block 2. Use $x^t = \left(1, 1, 0^{*31}\right)$ and $d = -\frac{1}{6}$ in Corollary 5. Then one only needs to know the submatrix

$$\begin{bmatrix} 8 & s \\ s & 8 \end{bmatrix}$$

of S to calculate that $-2s \geq \frac{33}{36} - \frac{32}{3}$ which simplifies to $s \leq \frac{117}{24}$ or $s \leq 4$. □

The next theorem replaces several ad hoc arguments in [8] and [7].

Theorem 9 *If*

$$\begin{bmatrix} 8 & a & b \\ a & 8 & c \\ b & c & 8 \end{bmatrix}$$

is a submatrix of S *then* $a+b \geq c$.

Proof Use Corollary 5 with $x^t = \left(-1, 1, 1, 0^{*30}\right)$ and $d = \frac{1}{6}$ to get $2a+2b-2c \geq \frac{33}{36} - \frac{8}{3}$ which simplifies to $c \leq \frac{7}{8} + a + b$ or $c \leq a + b$ □

Theorem 10 *If*

$$\begin{bmatrix} 8 & a & b & c \\ a & 8 & d & e \\ b & d & 8 & f \\ c & e & f & 8 \end{bmatrix}$$

is a submatrix of S *then* $a+b+c+d+e+f \geq 10$.

Proof Pub $x^t = \left(1, 1, 1, 1, 0^{0*29}\right)$ in Corollary 7 with $d = \frac{1}{22}$ to get $32+2(a+b+c+d+e+f) \geq \frac{512}{11} + \frac{18}{4}$ which simplifies to $a+b+c+d+e+f \geq 9 + \frac{23}{44}$. □

Corollary 11 *For any block,* $b_1 \leq 2$.

Proof Assume $b_1 \geq 3$. Put $a = b = c = 1$ in Theorem 10, then one of d, e or f must be greater than 3 which contradicts Theorem 9. □

Corollary 12 *There are no blocks with* $b_0 = 1$, $b_1 = 1$, *and* $b_2 = 1$.

Proof Put $a = 0$, $b = 1$, and $c = 2$. Then using Theorem 9 and Theorem 10 we obtain $d = 1$. Then $e = 2$ is obtained. Finally $w = 4$ by Theorem 10 and $w \neq 4$ by Theorem 9. □

Theorem 1 can now be proved using equations (1), (2) and (3) along with Theorem 8 and Corollaries 10 and 11.

3 CODING THEORY

Hall, Roth, van Rees and Vanstone [7] were the first to use coding theory on this problem. They considered the rows of the incidence matrix to be binary vectors. The span of these rows form a vector space, say C, over $GF(2)$. This is often called the point code of the design. The orthogonal complement of this vector space, defined to be the set of vectors that have dot product of 0 with all vectors in C, is again a vector space, denoted by $C^{\perp}$. If the weight of a vector is defined to be the number of non-zero positions, we can note that the weight of any row in the incidence matrix is always a multiple of 4 and that any two rows of the incidence matrix have an even number of positions where both rows contain a one. From this it can be proved that the weight of any codeword is 0 mod 4. From this it follows that C is self-orthogonal ie. $C \subseteq C^{\perp}$. From elementary linear algebra, it is known that dimension of C + dimension of $C^{\perp}$ = dimension of the space so the dimension of C, say s, is less than or equal to 16.

Let A_i be the number of codewords of weight i in the code, C, and B_i be the number of codewords of weight i in the orthogonal complement $C^{\perp}$. If the weight enumerator of a code of length n is defined to be $W_c(x, y) = \sum A_i x^{n-i} y^i$ the MacWilliams' identities [11] say that $W_{C^{\perp}}(x, y) = W_c(x + y, x - y)/|C|$. Hall et al. applied this to this problem and obtained the following 17 equations by equating coefficients:

$$2^s C_i = \sum_{j=0}^{8} \sum_{k=0}^{i} (-1)^{(i-k)} \binom{33-4j}{k} \binom{4j}{i-k} A_{4j}, i = 0, 1, \ldots 16$$

A few values may be filled in. Clearly $C_0 = 1$. Also, $C_1 = 0$ as there are no empty blocks and $C_2 = 0$ as there are no pair of identical blocks. It is also useful to note and easy to prove that all 1's vector, j, is in $C^{\perp}$ but not C. Now these equations can be manipulated to show that various C_i are positive. Hall et al. only used the coding theory to show that C_5 was positive.

Theorem 13 $C_5 \geq 6$.

Proof These equations have a kind of symmetry which is revealed when successive equations are subtracted as shown in Table 2. There we list the coefficients b_{ij} in the equation

$$(C_i - C_{i-1})\, 2^{s-3} = b_{i1} A_0 + b_{i2} (A_4 - A_{28}) + b_{i3} (A_8 - A_{24}) + b_{i4} (A_{12} - A_{20})$$

Table 2 Coefficients, b_{ij}, of the Manipulated MacWilliams' Identity

i	b_{i1}	b_{i2}	b_{i3}	b_{i4}	Equation #
1	4	3	2	1	1
3	616	238	52	-6	2
5	24552	4078	-12	-6	3
7	395560	19198	-1132	170	4
9	3085368	-2262	2588	-754	5
11	12621960	-475020	1860	1794	6
13	27293640	-20010	-13340	-2574	7
15	27293640	300150	16100	2002	8

Now if equation 3 is added to 6 times equation 1, then $2^{s-9}C_5 - 2^{s-9}C_4 = 6.2^{s-9}$ +384 - $64A_{28}$ + $64A_4$. Since all the codewords in C have even weight and all the 1's vector is in $C^{\perp}$, then $C_5 = C_{28}$. But $C_{28} \geq A_{28}$ so $C_5 \geq A_{28}$. Also $C_4 \geq A_4$. Using these small inequalities, the equation becomes $\left(2^{s-9} + 64\right) C_5 \geq \left(2^{s-9} + 64\right) C_4 + 6\left(2^{s-9} + 64\right)$ which gives us $C_5 \geq A_4 + 6$. □

Similarly, it can be proved that $C_9 \geq A_8 + 754$ and that $C_{13} \geq A_{12} + 6670$. Also it can be proved that C_8, C_{16} and of course C_{12} are greater than 0. But the most useful result is the one where the subscript is smallest. What does $C_5 > 0$ imply? It implies that some 5 columns, say the first 5, in the incidence matrix of any such design has the property that all the rows in those 5 columns have even weight. Hence Hall et al. went on to prove that there were 108 ways that those 5 columns could exist. Many were clearly impossible. So after much tedious work, they eliminated these possibilities to 13 possible cases. Rather than duplicate this approach, we will follow Greig's work that proved that there were 7 possible arrangements (unique up to row rearrangement).

Greig first examined, in the incidence matrix, the 5 columns corresponding to a codeword of weight 5 in $C^{\perp}$. We will suppose that these columns are the first 5 columns of A and will designate them as A_5. By examining these columns, the following lemma about the corresponding 5 by 5 submatrix of S will be proved. Designate this submatrix as S_5. Let s be the sum of the off-diagonal entries in S_5, let r_i be the sum of the off-diagonal entries in row i in S_5 and R_j be the sum of the first 5 entries of a row of S that is not in S_5.

Lemma 14 *$s/4, r_i$ and R_j are even.*

Proof Let x_i be the number of rows in A_5 that have weight i. Clearly $\sum x_i = 22$ and $\sum i x_i = 40$. Solving, we get $x_2 = 20 - 2x_4$ and $x_0 = 2 + x_4$. Since a weight i row of A_5 contributes $i(i-1)$ to s, then $s = 12x_4 + 40 - 4x_4 + 0 = 40 + 8x_4$.

The a weight j rows in A_5 that have a 1 in column i contribute $j - 1$ to the off-diagonal sum of row i in S_5. So $r_i = 3b + (8 - b) = 8 + 2b$.

Let x_i be the number of one's in column j that occur in a row which has weight i in A_5. Since such a row contributes i to the sum of the first 5 entries of R_j, $R_j = \sum i x_i$. Since $x_2 = 8 - x_0 - x_4$, $R_j = 4x_4 + 2(8 - x_0 c - x_4 d) = 16 + 2x_4 - 2x_0$. □

Since these various quantities are even the entry $\frac{1}{2}$ can be used in the vector x^t in Corollaries 5, 6 and 7 without violating the integrality constraints.

Lemma 15 *$s = 40, 48$ or 56.*

Proof In Corollary 5, use $d = -\frac{1}{6}$ and $x^t = \left(\left(\frac{1}{2}\right)^{*5}, 0^{*28}\right)$ to get $15.75 \geq \frac{s}{4}$. In corollary 7, use $d = -\frac{9}{22}$ and the same x^t to get $\frac{s}{4} \geq 10$. Since s is a multiple of 8, the lemma follows.

□

Lemma 16 *$r_i \leq \frac{s}{4} + 2$.*

Proof We will prove the result for r_5. In corollary 7, use $x^t = \left(\frac{1}{2}^{*4}, -\frac{1}{2}, 0^{*28}\right)$ and $d = \frac{-1}{22}$. □

Lemmas 15 and 16 severely restrict the possibilities for S_5. Also noting that r_i is greater than 8 and that r_i is even leads to the following theorem. We assume that the r_i's are in decreasing order and let $\hat{r}$ be the vector of r_i's.

Theorem 17 *If $s = 40$ then $r^+ = 8^{*5}$. If $s = 48$, then $r^+ = 10^{*4}8^{*1}$. If $s = 56$, then $r^+ = 12^4 8^{*1}$ or $12^{*3}10^{*2}$.*

Lemma 18 *The possible values for s_{ij} given r_i and r_j are given in Table 3.*

Proof We will prove the lemma for $i = 1$ and $j = 2$. Take $x^t = \left(\left(-\frac{1}{2}\right)^{*2}, \frac{1}{2}^{*3}, 0^{*28}\right)$ and $d = \frac{1}{6}$ in Corollary 5 to obtain that $\frac{1}{4}s - r_1 - r_2 + 2s_{12} \leq -\frac{1}{4}$ and hence the upper bound. Take the same x_t and $d = \frac{7}{22}$ in Corollary 7 to obtain that $10 + \frac{s}{4} - r_1 - r_2 + s_{12} \geq 4$ and hence the lower bound. □

Table 3 Intersection Numbers in S_5

s	r_i	r_j	s_{ij}
40	8	8	0,1,2
48	8	10	0,1,2
48	10	10	1,2,3
56	8	12	0,1,2
56	10	10	0,1,2
56	10	12	1,2,3
56	12	12	2,3,4

With the two previous lemmas and a bit of case work, it can be shown that there are 7 possible S_5's. They are listed in Table 4 using the numbering notation from Hall et al. [7].

If a (22,33,12,8,4)-BIBD exists, then at least six of these S_5's must be embedded in the structure matrix, possibly the same one six times. Also note that $H64$ and $H65$ have 0 determinant. It is also the case that each of the 7 corresponding A_5 are uniquely determined by the S_5. Now the same process can be done for S_3 and S_4 except that the design's structure matrix need not contain these submatrices. Greig [5] was the first to list these tables as shown in Table 5.

Coding theory was used in a different fashion by Bate, Hall, and Van Rees [1]. They use the fact that the incidence matrix has 22 rows but has rank at most 16. The easiest way to think of their results is to consider the other code obtainable from the incidence matrix, the so-called block code. The columns of the matrix are treated as binary vectors and the block code is the binary code which is the span of the column vectors. This code then has maximum dimension 16 and its dual code has dimension at least 6. Hence the dual block code has at least 2^6 vectors in it. The ones in a codeword of the dual block code pick out a set of rows from incidence matrix that sum to 0. The ones

Table 4 Possible S_5's

H1	H38	H64	H65
82222	83322	84422	84332
28222	38232	48242	48332
22822	32832	42842	33842
22282	23382	24482	33482
22228	22228	22228	22228

H74	H81	H82
83333	83333	82433
38333	38432	28433
33833	34823	44822
33381	33282	33282
33318	32328	33228

Table 5 Possible S_3's and Possible S_4's

844	8233	8431	8422
484	2833	4813	4822
448	3382	3184	2284
	3328	1348	2248

in the rows picked out from the incidence matrix intersect each column of the incidence matrix an even number of times.

In terms of our BIBD, this means that a codeword of the dual block code corresponds to s elements of V that occur an even number of times in each block of the design. This sub-structure may be called an even sub-(r, λ)-design. An (r, λ)-design is a BIBD except that k does not have to be constant, but the even refers to the fact that the block sizes are even.

Now if an even sub-(r, λ)-design has s elements and b_i blocks of size i, by simple counting we get $\sum b_i = b$, $\sum ib_i = rs$ and $\sum i(i-1)b_i = \lambda s(s-1)$. In order to have solutions to these equations, $s \geq 4$ and s is even. There are 12 sets of solutions (called B-vectors) to these equations of which two of the solutions are easily eliminated. The B-vectors are listed in Table 6 and we use the numbering

system from Bate, Hall and van Rees [1]. This gives us the following theorem.

Theorem 19 *There are at least* 2^6 *even sub-*(r, λ)*-designs in any (22,33,12,8,4)-BIBD and each one must correspond to the 10 B-vectors listed in Table 6.*

Table 6 Admissible B-vectors for Even Sub-(r, λ)-designs

type	s	b_0	b_2	b_4	b_6	b_8
1	4	9	24	0	0	0
2	6	3	24	6	0	0
3	6	2	27	3	1	0
6	8	1	16	16	0	0
7	8	0	19	13	1	0
8	10	3	0	30	0	0
9	10	2	3	27	1	0
10	10	1	6	24	2	0
11	10	0	9	21	3	0
12	10	0	8	24	0	1

If the BIBD has a type 8 even sub-(r, λ)-design in it, then the design splits into a (10,30,12,4,4) design and a slightly modified (12,33,12,4,4) design. Marshall Hall Jr. [6] investigated the case where the (10,30,12,4,4) is the 3-design obtained by adjoining a tenth point to the Steiner triple system on 9 points. Although this would be a beautiful construction it does not work. Tonchev has told me one of his students has tried tens of different (10,30,12,4,4) designs and none embed in 'the' design.

Finally, we mention results obtained by Demeng, Greig and van Rees [3] that were not published. By using a linear programming package, they showed that there were many solutions to the MacWilliams' Identity especially when s was less than 16. If $s = 16$ then the dimension of $C^{\perp}$ is 17. Now, $\sum A_{4i} = 2^{16}$ and since $A_i \subset C_i$, $\sum C_{4i} = 2^{16}$. But then $\sum C_{33-4i} = 2^{16}$. This implies that $A_{4i} = C_{4i} = C_{33-4i}$ and that $C_i = 0$ for $i = 1, 2$ modulo 4. If this is substituted back into the MacWilliams' Identity, this gives $A_0 = 1$, $A_4 = t$, $A_8 = 411 + 10t$, $A_{12} = 10886 - 49t$, $A_{16} = 35511 + 76t$, $A_{20} = 17556 - 49t$, $A_{24} = 1165 + 10t$ and $A_{28} = 6 + t$. There are 222 values of t that leave all the variables as non-negative integers. Many of these solutions may not correspond to any code. Some will correspond to codes but not codes that come from the

incidence matrix of a (22,33,12,8,4)-BIBD. As an example, when $t = 45$, there are 2 self-orthogonal codes that have the required number of codewords with the required weights. Using the notation from Pless [14], they are $A_8 \bigoplus A_8 \bigoplus \bar{I}_{17}^{(3)}$ and $E_{16} \bigoplus \bar{I}_{17}^{(3)}$, but neither comes from an incidence matrix as will be proved in Theorem 26.

Since $C_9 > 0$ and $C_{13} > 0$ lead to large but very numerous starting configuration, they tried to prove that there were a small number of columns in A that had all odd weight rows. They did this by considering a matrix, B, which is the incidence matrix of the design, A, with all the 1's column adjoined to it making a 22 by 34 matrix. Let B' be the code generated by the rows of B and let D' be the orthogonal complement of B'. Let B_i be the number of codewords of weight i in B' and let D_i be the number of codewords of weight i in D'. Note that the weights of the codewords in B' are 0 or 1 modulo 4. If we let $E_{i-1} = D_i - C_i$ $(E_{-1} = 0)$ then E_i's are the number of odd weight sets of i columns in A, the original matrix. One can express the MacWilliams' Identity in terms of these E_i's in the hope of proving that $E_4 > 0$. But the best result achieved is that one of E_4, E_5, E_6 or $E7$ is positive or that E_8 is positive. But these kind of results are not very useful in obtaining a large unique structure that must be embedded in the design.

4 GROUP METHODS

An entirely different approach to finding this design was taken by Landgev and Tonchev [10] and by Kapralov [9]. They wanted to use the automorphism group of the design to severely limit the search space they had to examine. To explain this several definitions are needed. Two BIBDs, (V_1, B_1) and (V_2, B_2), are isomorphic if there exists a bijection $a: V_1 \rightarrow V_2$ such that $B_1 a = B_2$. An automorphism of a design is an isomorphism of the design onto itself. The set of all automorphisms form a group called the full automorphism group of the design. Any subgroup is called an automorphism group of the design. Using these definitions, we will outline their results.

Given a BIBD, D, with an automorphism group G, the orbit matrix $M = (m_{ij})$ of D with respect to G is defined as a v' by b' matrix whose rows and columns are indexed by the element orbits and block orbits of D under G respectively, where m_{ij} is the number of elements from the i'th element orbit contained in a block from the j'th block orbit. M is often referred to as a tactical decomposition of D defined by the action of G.

M must obey the following constraints. Let r_j be the length of j'th block orbit and k_i be the length of the i'th element orbit.

$$1) \quad \sum_{i=1}^{v'} m_{ij} = k$$

$$2) \quad \sum_{j=1}^{b'} r_j m_{ij} = k_i r$$

$$3) \quad \sum_{j=1}^{b'} r_j m_{ij} (m_{ij} - 1) = k_i (k_i - 1) \lambda$$

$$4) \quad \sum_{j=1}^{b'} r_j m_{ij} m) i'_{,j} = k_i k_{i'}, \lambda, \quad i \neq i'$$

In the rest of this paper, we will be assuming that an automorphism group of the BIBD will have order p, a prime. So all orbits will have length p or length 1. If the length of the orbit is p (or 1), then we call a block or element in it a moving (fixed) block or element. The following lemma from Landgev and Tonchev [10] tells us which p are possible.

Lemma 20 *If p, a prime, is the order of an automorphism group of a BIBD (v, b, r, k, λ), then either p divides v or $p \leq r$.*

Proof Suppose p does not divide v. Then there must be a fixed element, x, in the design. If x occurs in a moving block it occurs p times and $p \leq r$, the number of times that any element must occur. If $p > r$, then the fixed element must occur only in fixed blocks. Every other element of the design must occur with x in a fixed block. If one moving element is in a fixed block then the whole orbit containing that moving element must be in the fixed block. Since $p > r \geq k$, every element of a fixed block must be fixed. Then there are no moving elements or $p = 1$ - contradiction. □

Applying Lemma 20 to this design gives us the following result.

Lemma 21 *The only primes which might be the orders of automorphism groups of a (22,33,12,8,4)-BIBD are 11,7,5,3, or 2.*

The idea now is to examine the primes one at a time, either to eliminate them or find the design. Now if $p = 11$, and if the automorphism group has no fixed point, then equations (1), (2) and (3) have no solution. If $p = 11$ and the automorphism has a fixed point then it must have 8 fixed points. But there must also be fixed blocks because if a fixed element were only in moving blocks then it would occur 0 mod 11 times but $r = 12$. But if there is one fixed block there must be at least 11 of them. Since no block can contain an all elements of a point orbit when $p = 11$, the fixed block contains only fixed points and a quick count shows that the fixed element pairs occur too frequently.

If $p = 7$, then there are 5 mod 7 fixed blocks. Since a complete orbit of elements can be contained in a block only once, as otherwise the intersection would be larger than 4, there must be at least 2 fixed blocks containing only fixed points. Therefore there must be, at least 8+8-4=12 fixed points. Therefore the number of fixed points is exactly 15. In a moving block there must be at least 2 fixed points, but then these fixed points occur together 7 times which contradicts $\lambda = 4$.

If $p = 5$, then there are at least 3 fixed blocks. A fixed block must contain at least 3 fixed points and since the number of fixed popints is 2 mod 5, there are at least 7 fixed points and at most 3 moving element orbits. Since the moving blocks cannot contain fixed element pairs and since the three fixed blocks cannot contain all the fixed element pairs, there must be at least 8 fixed blocks. Then there are at least 5 fixed blocks containing only fixed points. By checking block intersections we see that these 5 blocks must contain at least 13 fixed points. Therefore there are exactly 17 fixed points and one moving element orbit. Therefore a moving block orbit contains at least 2 fixed point contradicting $\lambda = 4$.

If $p = 3$, the situation is more tedious. We will not give all the details here but they can be found in the two previous papers mentioned. There are many cases depending on the number of fixed elements and blocks. The proof of the following well-known lemma can be found in a paper by Block [2].

Lemma 22 *If α is an automorphism of a BIBD that has t element orbits and s block orbits then $0 \leq s - t \leq b - v$.*

Also helpful in eliminating these cases is the following lemma of Landgev and Tonchev's [10]. The proof uses just elementary counting arguments.

Lemma 23 *An automorphism of order 3 in a BIBD can fix at most* $b-3r+3\lambda$ *blocks.*

With these last two lemmas as a starting point it can be shown that there are 5 cases to investigate in detail. The first case occurs when there are exactly 10 fixed elements and 9 fixed blocks. One can write down the orbit matrix for the moving element orbits. However these cannot be extended to the full orbit matrix. If one assumes that there are exactly 7 fixed points, then the next three cases occur when there are 3,6, or 9 fixed blocks. The case of 3 fixed blocks is easily ruled out and the case of 6 fixed blocks can also be ruled out with some more work. This leaves the case of 9 fixed blocks and 7 fixed elements. But the 5 rows of the orbit matrix corresponding to the moving element orbits cannot be filled in. Penultimately, we have the case of 4 fixed elements. With a bit of work, this can be shown to imply that there are 6 fixed blocks. This enables one to determine three possibilities for the rows of the orbit matrix corresponding to the fixed elements. This leaves 7 possibilities for each of the other rows. With much case work, each of the 3 possibilities can be slowly eliminated. It was not possible to obtain a viable orbit matrix in any of these cases. This is where Landgev and Tonchev stop.

Finally, we have the single left over case when the automorphism group has order 3 ie. exactly 1 fixed element. This case is the sole contents of Kapralov [9]. The only feasible way to proceed was to use computers to tackle the very many subcases involved. First, since a block does not have size 0 or 1 mod 3, there are no fixed blocks. Solving the equations (1) to (4) for this subcase leads to 53 non-isomorphic orbit matrices. We list the first of the 53 possible orbit matrices.

$$\begin{bmatrix} 00011112222 \\ 01122220011 \\ 02211112200 \\ 12200110122 \\ 20202112011 \\ 21122000211 \\ 22020112011 \\ 10000110100 \end{bmatrix}$$

At this point, Kapralov had to expand each orbit matrix into an incidence matrix by taking each integer in the orbit matrix and replacing it by a 3 by 3 matrix in the 7 moving element orbits and a 1 by 3 matrix in the fixed element orbit. Kapralov did this using a fairly lengthy backtrack program. When he

did this, none of the orbit matrix expansions were successful. We sum up this section with the following theorem.

Theorem 24 *The full automorphism group of a (22,33,12,8,4)-BIBD must either be a 2-group or trivial.*

To conclude this section, there are still people eliminating some of the 2-groups. For instance, Siftar [16] has eliminated the possibility of a cyclic group of order 8 in the automorphism group of this design. However, it seems unlikely that this will be a fruitful avenue of discovery because even if 2-groups are ruled out, there is still the possibility of the trivial automorphism group. All the previous group work will be of no help in this situation.

5 NEW WORK

There has been a great deal of work done by coding theorists on self-orthogonal codes. They have classified all self-dual codes up to length 32. A self-dual code is a code that is equal to its orthogonal complement. Now most self-orthogonal codes of length 32, even with the correct parameters, are not going to contain 22 codewords that would be the rows of an incidence matrix of our BIBD. But if the BIBD exists, the code must exist.

Since the self-orthogonal codes of length 33 have not been enumerated, one can try to get at the self-dual codes of length 32 and dimension 16 which have been tabulated. This can be done as follows. Let C be the binary code generated by the incidence matrix of a (22,33,12,8,4)-BIBD. Assume that C has dimension 16. Let D be the code obtained from C by choosing the codewords from C that have a 0 in the first coordinate position (any coordinate position will do). Since the heaviest weight codeword in C is 28, then D does not contain all the 1's codeword, j. Then the code generated by $E = D \bigoplus j$ is a (32,16) self-dual linear code. If E has minimum weight 8, then there are just 5 codes to check. However, if the minimum weight is 4, then there are 80 to check. That is not to say that the checking will be easy. It may require huge amounts of computing to search these known codes. This approach has been thought about but nobody has seriously attempted it to my knowledge.

Another similar approach to the problem is to use our knowledge about the structure of the design to eliminate possible codes as codes that come from

the incidence matrix of this design. The idea is to examine the enumeration of self-orthogonal codes in Pless [14] and Pless and Sloane [15] and eliminate codes that could not possibly come from the incidence matrix of our design and hope very few codes are left over. In this vein we give the following theorem. But first some necessary terminology from coding theory. An (in)decomposable code is (not) a direct sum of its subcodes. We will assume that the generator matrices for the codes in this section have the property the matrices contain the least weight codewords possible. See Pless [14] for more details.

Theorem 25 *The self-orthogonal code that is generated by the incidence matrix of a (22,33,12,8,4)-BIBD is indecomposable if the rank of the incidence matrix is 16.*

Proof The self-orthogonal codes of dimension 16 and length 33 have not been determined but the literature [14] and [15] contains enough information for us to find out if the code that comes from the incidence matrix of our design can be expressed as a direct sum of two codes, $A_i \bigoplus A_j$. Clearly, one of the codes must be a self-dual code of dimension t and length $2t$ and the other must be a self-orthogonal code of dimension t' and length $2t'+1$. The weight enumerator of $A_i \bigoplus A_j$ must be the product of the weight enumerators of A_i and A_j. This means both A_i and A_j must only contain codewords of weight 0 modulo 4. Also since the heaviest weight word in our code is suppose to be 28 and since self-dual codes contain the all 1's codeword, the odd length $(2t'+1)$ code must have heaviest codeword of weight $2t'-4$. By checking the two papers cited, there are only 2 possibilities. They are $A = A_8 \bigoplus A_8 \bigoplus \bar{I}_{17}^{(3)}$ and $B = E_{16} \bigoplus \bar{I}_{17}^{(3)}$ where the generator matrices of the subcodes are

$$A_8 \text{ is } \begin{bmatrix} 11110000 \\ 00111100 \\ 00001111 \\ 10101010 \end{bmatrix},$$

$$E_{16} \text{ is } \begin{bmatrix} 1111000000000000 \\ 0011110000000000 \\ 0000111100000000 \\ 0000001111000000 \\ 0000000011110000 \\ 0000000000111100 \\ 1010101010101010 \end{bmatrix} \text{ and } \bar{I}_{17}{}^{(3)} \text{ is } \begin{bmatrix} 1111000000000000 \\ 0011110000000000 \\ 0000111100000000 \\ 0000001111000000 \\ 1010101010111111 \\ 0000000000111000 \\ 0000000000011110 \\ 0000000000101010 \end{bmatrix}$$

It is easy to show that any code that contains A_8 as a subcode is decomposable into $A_8 \bigoplus A_j$. Assume that A contains A_8 as a subcode and that in the generator for A, A_8, occurs in the first 8 columns. Then the first 8 columns of each row of the incidence matrix of our design must be a linear combination of the four rows of A_8. Let the first 8 columns of the incidence matrix be designated by Z. The rows of Z must then have weight 0, 4 or 8. Let x_i be the number of rows of weight i in Z. Then $x_8 + x_4 + x_0 = 22$ and $8x_8 + 4x_4 = 64$ implies that $x_0 = x_8 + 6$. This implies that incidence matrix has a 6 by 8 rectangle of 0's.

If we now let y_i be the number of columns of weight i in the incidence matrix rows that correspond to the 0 weight rows of S, we can state that $\sum y_i = 25$, $\sum iy_i = 72$ and $\sum i(i-1)y_i = 120$. Since there are no non-negative solutions to these equations, this situation is impossible. Hence A is not the code generated by the incidence matrix of our design.

Let us next consider a code containing the first 10 columns of $\bar{I}_{17}^{(3)}$. Let us say that these columns occur in the first 10 coordinate positions. Further, let us assume that the code is generated by the incidence matrix of our design. Then the incidence matrix for our design in the first 10 columns has rows of weight 0, 4, 5 or 8. Let x_i be the number of rows in these columns with weight i. Then we can state that $x_0 + x_4 + x_5 + x_8 = 22$ and $4x_4 + 5x_5 + 8x_8 = 80$. This implies that $x_0 \geq 2$. If $x_0 \geq 3$, then we let y_i be the number of columns of weight i in the incidence matrix rows that correspond to the 0 weight rows. We can state that $\sum y_i = 23$, $\sum iy_i = 36$ and $\sum i(i-1)y_i = 24$. Since this has no non-negative solutions, this situation is impossible so B is not the code generated by the incidence matrix of our design. □

This means that the code generated by a rank 16 incidence matrix of our design is a truncated code of a self-dual code of dimension 17 and length 34. Unfortunately, these have not been enumerated as yet.

Now there are many other configurations that occur in self-orthogonal codes that cannot occur in the code that is generated from an incidence of a (22, 33, 12, 8, 4)-BIBD. It seems likely if there is such a BIBD then the code generated by its incidence matrix will be indecomposable no matter what the rank of the incidence matrix. But it is more likely that the design does not exist.

Finally we announce the exciting results of McKay and Radziszowski [13] who have used at least 2 years of CPU time to make major breakthroughs. They prove the following theorems.

Theorem 26 *Any two blocks in a (22,33,12,8,4)-BIBD must have non-empty intersection.*

Theorem 27 *Every quadruple of points in a (22,33,12,8,4)-BIBD can occur in at most two blocks.*

Theorem 28 *No three blocks in a (22,33,12,8,4)-BIBD can pairwise intersect in one point.*

Theorem 26 eliminates type 4 intersection patterns. This is the first type ruled out since the paper by Hamada and Kobayashi. This is a huge advance. The other two theorems rule out configurations that have teased researchers for years.

McKay and Radziszowski have also found what they call near designs. These have 22 elements, 33 blocks, block size 8, element replication 12, allowable block intersection and 227 of the 231 pairs of elements in the design occur 4 times. This is as close to being the BIBD as you can get without actually being the BIBD. This indicates that the non-existence of this design will be quite hard to prove or will rely on a great deal of computing.

6 ACKNOWLEDGEMENTS

I have had many useful discussions in various media with the following people in the writing of this paper. I wish to thank them for their help. Malcolm Greig, Brendan McKay, Vladimir Tonchev, Ed Assmus, Cheng Demeng, and Stanislaw Radziszowski. I also wish to thank the editor Wal Wallis for his encouragement to write this paper and his patience in waiting for me to finally finish it.

REFERENCES

[1] J.A. Bate, M. Hall Jr., and G.H.J. van Rees. Structures within (22,33,12,8,4)-designs. *J. Comb. Math. Comb. Comp.*, 4 (1988), 115–122.

[2] R.E. Block. On the orbits of collineation groups. *Math. Zeitschr.*, 96 (1967), 33–49.

[3] C. Demeng, M. Greig, and G.H.J. van Rees. non-existent preprint.

[4] R.A. Fisher and F. Yates. *Statistical Tables for Biological, Agricultural and Medical Research*, volume 93. Hafner, New York, 6th edition edition, 1963.

[5] M. Greig. An improvement to connor's criterion. preprint.

[6] M. Hall Jr. Constructive methods for designs. *Cong. Numer.*, 66 (1988), 141–144.

[7] M. Hall Jr., R. Roth, G.H.J. van Rees, and S.A. Vanstone. On designs (22,33,12,8,4). *J. Combinatorial Theory*, 47 (1988), 157–175.

[8] N. Hamada and Y. Kobayshi. On the block structure of BIB designs with parameters $v = 22, b = 33, r = 12, k = 8$ and $\lambda = 4$. *J. Combinatorial Theory*, 24A (1978), 75–83.

[9] S. Kapralov. Combinatorial 2-(22,8,4) designs with automorphisms of order 3 fixing one point. In *Math. and Education in Math.*, Proc. of the XVI Spring Conference of Union of Bulgarian Mathematicians, pages 453–458. Sunny Beach, 1987.

[10] I. Landgev and V. Tonchev. Automorphisms of 2-(22,8,4) designs. *Discrete Math.*, 77 (1989), 177–189.

[11] F.J. MacWilliams. A theorem on the distribution of weights in a systematic code. *Bell System Tech. J.*, 42 (1963), 79–94.

[12] R. Mathon and A. Rosa. Tables of parameters of BIBDs with $r <= 41$ including existence, enumeration and resolvability results: An update. *Ars Combin.*, 30 (1991), 65–96.

[13] B.D. McKay and S.P. Radziszowski. Towards deciding the existence of 2-(22,8,4) designs. *J. Comb. Math. Comb. Comp.*. submitted.

[14] V. Pless. A classification of self-orthogonal codes over GF(2). *Discrete Math.*, 3 (1972), 209–246.

[15] V. Pless and N.J.A. Sloane. On the classification and enumeration of self-dual codes. *J. Combinatorial Theory*, 18A (1975), 313–335.

[16] J. Siftar. On 2-groups operating on a 2-(22,8,4) design. *Rad.-Mat.*, 7(2) (1991), 217–229.

Other *Mathematics and Its Applications* titles of interest:

P.H. Sellers: *Combinatorial Complexes. A Mathematical Theory of Algorithms.* 1979, 200 pp. ISBN 90-277-1000-7

P.M. Cohn: *Universal Algebra.* 1981, 432 pp.
ISBN 90-277-1213- 1 (hb), ISBN 90-277-1254-9 (pb,

J. Mockor: *Groups of Divisibility.* 1983, 192 pp. ISBN 90-277-1539-4

A. Wwarynczyk: *Group Representations and Special Functions.* 1986, 704 pp.
ISBN 90-277-2294-3 (pb), ISBN 90-277-1269-7 (hb)

I. Bucur: *Selected Topics in Algebra and its Interrelations with Logic, Number Theory and Algebraic Geometry.* 1984, 416 pp. ISBN 90-277-1671-4

H. Walther: *Ten Applications of Graph Theory.* 1985, 264 pp.
ISBN 90-277-1599-8

L. Beran: *Orthomodular Lattices. Algebraic Approach.* 1985, 416 pp.
ISBN 90-277-1715-X

A. Pazman: *Foundations of Optimum Experimental Design.* 1986, 248 pp.
ISBN 90-277-1865-2

K. Wagner and G. Wechsung: *Computational Complexity.* 1986, 552 pp.
ISBN 90-277-2146-7

A.N. Philippou, G.E. Bergum and A.F. Horodam (eds.): *Fibonacci Numbers and Their Applications.* 1986, 328 pp. ISBN 90-277-2234-X

C. Nastasescu and F. van Oystaeyen: *Dimensions of Ring Theory.* 1987, 372 pp.
ISBN 90-277-2461-X

Shang-Ching Chou: *Mechanical Geometry Theorem Proving.* 1987, 376 pp.
ISBN 90-277-2650-7

D. Przeworska-Rolewicz: *Algebraic Analysis.* 1988, 640 pp. ISBN 90-277-2443-1

C.T.J. Dodson: *Categories, Bundles and Spacetime Topology.* 1988, 264 pp.
ISBN 90-277-2771-6

V.D. Goppa: *Geometry and Codes.* 1988, 168 pp. ISBN 90-277-2776-7

A.A. Markov and N.M. Nagorny: *The Theory of Algorithms.* 1988, 396 pp.
ISBN 90-277-2773-2

E. Kratzel: *Lattice Points.* 1989, 322 pp. ISBN 90-277-2733-3

A.M.W. Glass and W.Ch. Holland (eds.): *Lattice-Ordered Groups. Advances and Techniques.* 1989, 400 pp. ISBN 0-7923-0116-1

N.E. Hurt: *Phase Retrieval and Zero Crossings: Mathematical Methods in Image Reconstruction.* 1989, 320 pp. ISBN 0-7923-0210-9

Du Dingzhu and Hu Guoding (eds.): *Combinatorics, Computing and Complexity.* 1989, 248 pp. ISBN 0-7923-0308-3

Other *Mathematics and Its Applications* titles of interest:

A.Ya. Helemskii: *The Homology of Banach and Topological Algebras.* 1989, 356 pp. ISBN 0-7923-0217-6

J. Martinez (ed.): *Ordered Algebraic Structures.* 1989, 304 pp. ISBN 0-7923-0489-6

V.I. Varshavsky: *Self-Timed Control of Concurrent Processes. The Design of Aperiodic Logical Circuits in Computers and Discrete Systems.* 1989, 428 pp. ISBN 0-7923-0525-6

E. Goles and S. Martinez: *Neural and Automata Networks. Dynamical Behavior and Applications.* 1990, 264 pp. ISBN 0-7923-0632-5

A. Crumeyrolle: *Orthogonal and Symplectic Clifford Algebras. Spinor Structures.* 1990, 364 pp. ISBN 0-7923-0541-8

S. Albeverio, Ph. Blanchard and D. Testard (eds.): *Stochastics, Algebra and Analysis in Classical and Quantum Dynamics.* 1990, 264 pp. ISBN 0-7923-0637-6

G. Karpilovsky: *Symmetric and G-Algebras. With Applications to Group Representations.* 1990, 384 pp. ISBN 0-7923-0761-5

J. Bosak: *Decomposition of Graphs.* 1990, 268 pp. ISBN 0-7923-0747-X

J. Adamek and V. Trnkova: *Automata and Algebras in Categories.* 1990, 488 pp. ISBN 0-7923-0010-6

A.B. Venkov: *Spectral Theory of Automorphic Functions and Its Applications.* 1991, 280 pp. ISBN 0-7923-0487-X

M.A. Tsfasman and S.G. Vladuts: *Algebraic Geometric Codes.* 1991, 668 pp. ISBN 0-7923-0727-5

H.J. Voss: *Cycles and Bridges in Graphs.* 1991, 288 pp. ISBN 0-7923-0899-9

V.K. Kharchenko: *Automorphisms and Derivations of Associative Rings.* 1991, 386 pp. ISBN 0-7923-1382-8

A.Yu. Olshanskii: *Geometry of Defining Relations in Groups.* 1991, 513 pp. ISBN 0-7923-1394-1

F. Brackx and D. Constales: *Computer Algebra with LISP and REDUCE. An Introduction to Computer-Aided Pure Mathematics.* 1992, 286 pp. ISBN 0-7923-1441-7

N.M. Korobov: *Exponential Sums and their Applications.* 1992, 210 pp. ISBN 0-7923-1647-9

D.G. Skordev: *Computability in Combinatory Spaces. An Algebraic Generalization of Abstract First Order Computability.* 1992, 320 pp. ISBN 0-7923-1576-6

E. Goles and S. Martinez: *Statistical Physics, Automata Networks and Dynamical Systems.* 1992, 208 pp. ISBN 0-7923-1595-2

Other *Mathematics and Its Applications* titles of interest:

M.A. Frumkin: *Systolic Computations.* 1992, 320 pp. ISBN 0-7923-1708-4

J. Alajbegovic and J. Mockor: *Approximation Theorems in Commutative Algebra.* 1992, 330 pp. ISBN 0-7923-1948-6

I.A. Faradzev, A.A. Ivanov, M.M. Klin and A.J. Woldar: *Investigations in Algebraic Theory of Combinatorial Objects.* 1993, 516 pp. ISBN 0-7923-1927-3

I.E. Shparlinski: *Computational and Algorithmic Problems in Finite Fields.* 1992, 266 pp. ISBN 0-7923-2057-3

P. Feinsilver and R. Schott: *Algebraic Structures and Operator Calculus.* Vol. I. Representations and Probability Theory. 1993, 224 pp. ISBN 0-7923-2116-2

A.G. Pinus: *Boolean Constructions in Universal Algebras.* 1993, 350 pp. ISBN 0-7923-2117-0

V.V. Alexandrov and N.D. Gorsky: *Image Representation and Processing. A Recursive Approach.* 1993, 200 pp. ISBN 0-7923-2136-7

L.A. Bokut' and G.P. Kukin: *Algorithmic and Combinatorial Algebra.* 1994, 384 pp. ISBN 0-7923-2313-0

Y. Bahturin: *Basic Structures of Modern Algebra.* 1993, 419 pp. ISBN 0-7923-2459-5

R. Krichevsky: *Universal Compression and Retrieval.* 1994, 219 pp. ISBN 0-7923-2672-5

A. Elduque and H.C. Myung: *Mutations of Alternative Algebras.* 1994, 226 pp. ISBN 0-7923-2735-7

E. Goles and S. Martínez (eds.): *Cellular Automata, Dynamical Systems and Neural Networks.* 1994, 189 pp. ISBN 0-7923-2772-1

A.G. Kusraev and S.S. Kutateladze: *Nonstandard Methods of Analysis.* 1994, 444 pp. ISBN 0-7923-2892-2

P. Feinsilver and R. Schott: *Algebraic Structures and Operator Calculus.* Vol. II. Special Functions and Computer Science. 1994, 148 pp. ISBN 0-7923-2921-X

V.M. Kopytov and N. Ya. Medvedev: *The Theory of Lattice-Ordered Groups.* 1994, 400 pp. ISBN 0-7923-3169-9

H. Inassaridze: *Algebraic K-Theory.* 1995, 438 pp. ISBN 0-7923-3185-0

C. Mortensen: *Inconsistent Mathematics.* 1995, 155 pp. ISBN 0-7923-3186-9

R. Abłamowicz and P. Lounesto (eds.): *Clifford Algebras and Spinor Structures.* A Special Volume Dedicated to the Memory of Albert Crumeyrolle (1919–1992). 1995, 421 pp. ISBN 0-7923-3366-7

W. Bosma and A. van der Poorten (eds.), *Computational Algebra and Number Theory.* 1995, 336 pp. ISBN 0-7923-3501-5

Other *Mathematics and Its Applications* titles of interest:

A.L. Rosenberg: *Noncommutative Algebraic Geometry and Representations of Quantized Algebras*. 1995, 316 pp. ISBN 0-7923-3575-9

L. Yanpei: *Embeddability in Graphs*. 1995, 400 pp. ISBN 0-7923-3648-8

B.S. Stechkin and V.I. Baranov: *Extremal Combinatorial Problems and Their Applications*. 1995, 205 pp. ISBN 0-7923-3631-3

Y. Fong, H.E. Bell, W.-F. Ke, G. Mason and G. Pilz (eds.): *Near-Rings and Near-Fields*. 1995, 278 pp. ISBN 0-7923-3635-6

A. Facchini and C. Menini (eds.): *Abelian Groups and Modules*. (Proceedings of the Padova Conference, Padova, Italy, June 23–July 1, 1994). 1995, 537 pp. ISBN 0-7923-3756-5

D. Dikranjan and W. Tholen: *Categorical Structure of Closure Operators*. With Applications to Topology, Algebra and Discrete Mathematics. 1995, 376 pp. ISBN 0-7923-3772-7

A.D. Korshunov (ed.): *Discrete Analysis and Operations Research*. 1996, 351 pp. ISBN 0-7923-3866-9

P. Feinsilver and R. Schott: *Algebraic Structures and Operator Calculus*. Vol. III: Representations of Lie Groups. 1996, 238 pp. ISBN 0-7923-3834-0

M. Gasca and C.A. Micchelli (eds.): *Total Positivity and Its Applications*. 1996, 528 pp. ISBN 0-7923-3924-X

W.D. Wallis (ed.): *Computational and Constructive Design Theory*. 1996, 368 pp. ISBN 0-7923-4015-9