Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 70
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 70
Espoo 2002 HUT-TCS-A70

ISOMORPH-FREE EXHAUSTIVE GENERATION OF COMBINATORIAL DESIGNS

Petteri Kaski

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

Helsinki University of Technology Laboratory for Theoretical Computer Science
Research Reports 70
Teknillisen korkeakoulun tietojenkäsittelyteorian laboratorion tutkimusraportti 70
Espoo 2002 HUT-TCS-A70

ISOMORPH-FREE EXHAUSTIVE GENERATION OF COMBINATORIAL DESIGNS

Petteri Kaski

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science

Teknillinen korkeakoulu
Tietotekniikan osasto
Tietojenkäsittelyteorian laboratorio

Distribution:
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O.Box 5400
FIN-02015 HUT
Tel. +358-0-451 1
Fax. +358-0-451 3369
E-mail: lab@tcs.hut.fi

ISBN 951-22-5815-3
ISSN 1457-7615

Picaset Oy
Helsinki 2002

ABSTRACT: This report investigates algorithmic isomorph-free exhaustive generation of balanced incomplete block designs (BIBDs), resolvable balanced incomplete block designs (RBIBDs), and the corresponding resolutions. In particular, three algorithms for isomorph-free exhaustive generation of (v, k, λ)-BIBDs and resolutions are described and applied to settle existence and classification problems.

The first algorithm generates BIBDs point by point using an orderly algorithm in combination with a maximum clique algorithm. The second and third algorithm generate resolutions of BIBDs by utilizing a correspondence between resolutions and certain optimal q-ary error-correcting codes. The second algorithm generates the corresponding codes codeword by codeword, and is analogous in structure to the first algorithm. The third algorithm generates codes coordinate by coordinate, and is based on the recent isomorph-free exhaustive generation framework of Brendan McKay.

The main result of this report is a proof of nonexistence for a $(15, 5, 4)$ RBIBD by exhaustive computer search. Other new classification results include the classifications of the $(13, 6, 5)$- and $(14, 7, 6)$-BIBDs and the $(16, 4, 2)$-, $(14, 7, 12)$-, and $(24, 12, 11)$-RBIBDs together with their resolutions, which correspond to classifications of the $(10, 16, 8)_4$, $(26, 14, 14)_2$, and $(23, 24, 12)_2$ error-correcting $(n, M, d)_q$-codes, respectively. Additionally, a number of earlier classification results are corroborated.

KEYWORDS: balanced incomplete block design, error-correcting code, exhaustive search, group action, isomorph rejection, orderly algorithm, resolution, resolvable design

CONTENTS

1 INTRODUCTION

Combinatorial design theory is a branch of discrete mathematics which originated in the design of statistical experiments for agriculture and as the generalization of various recreational problems. Modern combinatorial design theory is a field of active research with connections to coding theory [75], graph theory [16], finite group theory [33], and computer science and cryptography [21, 23] in addition to the traditional applications in statistical experiment design [101, 114].

A *combinatorial design* can be described as an arrangement of a finite set of *points* into a finite collection of *blocks* so that the arrangement satisfies some prescribed properties. As an example, consider the following famous recreational problem posed by Thomas P. Kirkman in the *Lady's and Gentleman's Diary* of 1850.

> Fifteen young ladies in a school walk out three abreast for seven days in succession: it is required to arrange them daily, so that no two walk twice abreast.

A solution is given below.

Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
A B C	A H I	A J K	A D E	A F G	A L M	A N O
D J N	B E G	B M O	B L N	B H J	B I K	B D F
E H M	C M N	C E F	C I J	C L O	C D G	C H K
F I O	D K O	D H L	F K M	D I M	E J O	E I L
G K L	F J L	G I N	G H O	E K N	F H N	G J M

(A generalization of this problem to an arbitrary number of girls and days was unsolved for well over a hundred years until it was finally settled by Ray-Chaudhuri and Wilson [102] in 1968.)

Kirkman's problem and its generalizations are examples of an *existence problem* in design theory: Given a collection of properties, decide whether there exists a design realizing these properties. A problem related to the existence problem is the *classification problem*, in which one is asked to describe, up to some criterion of equivalence, all the designs that have the desired properties. An easier version of the classification problem is the *counting problem*, in which the task is to count, up to equivalence, the number of distinct designs meeting the properties.

Solving such classification problems is of interest for both practical and theoretical reasons. A complete catalogue of configurations of a particular type can be searched for the best configuration meeting some additional requirements, or for counterexamples to a conjecture. Additionally, the study of a classification often provides new insights in the form of theorems concerning the structure of a larger class of objects.

Algorithmic methods are widely used in design theory for solving existence and classification problems [42]. The success of algorithmic methods is based on the fact that the configurations studied are finite and thus amenable to local and exhaustive search methods (see [69, Ch. 4–5]). *Local* *search* methods are typically incomplete in the sense that they are not guaranteed to find a solution to a problem even if one exists. Thus, local search methods are primarily applicable to solving existence problems by explicit construction, where they typically outperform exhaustive methods. *Exhaustive search* considers all candidate solutions in turn and is guaranteed to find a solution if one exists. Exhaustive search can thus be applied to generate all combinatorial structures with particular properties, or to settle the nonexistence of such a structure. Indeed, often an exhaustive computer search has been the only instrument for demonstrating the nonexistence of a particular design. The nonexistence of finite projective planes of order 10 is perhaps the most notable such result to date [71]. Other examples are provided by [61, 85].

A central issue in the design of practical exhaustive search algorithms for the generation of combinatorial configurations is the detection and elimination of *equivalent*, or *isomorphic*, copies of configurations from consideration. This is because failure to do so results in redundant work proportional to the number of such copies—which is typically exponential in the size of the configuration—making the search very inefficient or altogether infeasible to conduct with available computational resources. Furthermore, from a mathematical point of view isomorphic configurations are by definition identical in the structure of interest, so it is desirable to eliminate all but one of such generated structures.

In this report we study in a group-theoretic framework the problem of isomorph-free exhaustive generation of two families of combinatorial designs, namely (balanced incomplete) block designs and their resolutions.

The organization of the report is as follows. The first two chapters focus on introducing the terminology and the necessary theoretical background on block designs and error-correcting codes. Chapter 2 introduces block designs and their resolutions, and derives representations for isomorphism equivalence classes of both as orbits of a suitable group action. Chapter 3 introduces the necessary coding-theoretic concepts and develops an alternative representation for resolutions of block designs as error-correcting codes based on a bijective correspondence between the two discovered by Semakov and Zinov'ev [109].

The next two chapters discuss isomorph-free exhaustive generation. Chapter 4 describes generic algorithmic frameworks developed for the problem of generating a collection of orbit representatives. As a new result we demonstrate that the McKay framework requires an additional axiom for correct operation of the associated orbit transversal algorithm. Chapter 5 develops three isomorph-free exhaustive generation algorithms for block designs and their resolutions based on the Read–Faradžev and McKay frameworks described in Chapter 4. The two Read–Faradžev-type algorithms for generation of block designs and resolutions apply a combination of clique search and standard orderly generation with alternating steps of generation and lexicographic maximality testing. The McKay-type algorithm generates resolutions of block designs parallel class by parallel class and uses the canonical labelling package *nauty* to compute canonical placement.

Chapter 6 describes the new classification results obtained and earlier results corroborated. The main result of this report is a nonexistence proof for a resolvable (15,5,4) balanced incomplete block design by exhaustive search. The other classification results are conveniently summarized in Tables 6.1 and 6.2 on page 66, so we shall not repeat them here.

Chapter 7 concludes the report by discussing the reliability of the classification results and the choices made in algorithm design. Additionally, several suggestions for future research are given.

The mathematical prerequisites appear in Appendix B. Appendix C discusses briefly the computational complexity of problems related to classification of block designs and the efficiency of algorithms that generate all solutions to a given problem.

2 COMBINATORIAL DESIGNS

This chapter gives a brief introduction to the theory of combinatorial designs. In particular, we discuss block designs, their resolvability, resolutions, and isomorphism of block designs and their resolutions. We assume familiarity with basic concepts and notation related to sets, finite groups, and group actions. Appendix B covers the presupposed mathematical definitions and notation not discussed in the main subject matter.

For an in-depth introduction to design theory the reader is encouraged to consult the books [4, 54]. A thorough survey of existence and classification results for block designs is given in [20, Part I]. A recent comprehensive treatment of existence theory and constructions for resolvable designs is [37].

2.1 SET SYSTEMS

We shall define block designs as a subfamily of a more generic family of combinatorial objects, namely *set systems.* Informally, a set system is simply a collection of subsets of a given set X, however, a rigorous definition is somewhat more involved because we want it to be possible for a subset of X to occur more than once in a set system. For this purpose, we use the following standard formalism for *multisets,* that is, sets in which an element may occur more than once.

Definition 2.1.1 For a nonempty set X, we define a *multiset* over X to be a mapping $\mathcal{E} : X \to \mathbb{N}$, which associates to an element $x \in X$ its *multiplicity* $\mathcal{E}(x)$.

We use the following additional terminology in the context of multisets. A multiset $\mathcal{E}$ over X is *finite* if $\mathcal{E}(x) \neq 0$ for only finitely many $x \in X$; otherwise $\mathcal{E}$ is *infinite*. The *cardinality* of a finite multiset $\mathcal{E}$ over X is defined by $|\mathcal{E}| = \sum_{x\in X} \mathcal{E}(x)$. The *empty* multiset is the multiset of cardinality zero. We say that $x \in X$ is an *element* of $\mathcal{E}$ if $\mathcal{E}(x) \geq 1$, and indicate this by writing $x \in \mathcal{E}$. Conversely, if $\mathcal{E}(x) = 0$, then we write $x \notin \mathcal{E}$.

Definition 2.1.2 We write $\mathscr{P}[X]$ for the set of all subsets of X, and $\mathscr{M}[X]$ for the set of all multisets over X. The restriction to subsets (respectively, multisets) of cardinality b is denoted by $\mathscr{P}_b[X]$ (respectively, $\mathscr{M}_b[X]$).

We use the standard brace bracket notation used to describe a set by listing its elements for multisets as well:

Example 2.1.3 Suppose $\mathcal{B}$ is a multiset over $\{0, 1, 2, 3\}$. Then we write, say, $\mathcal{B} = \{0, 0, 1, 2\}$ to indicate that $\mathcal{B}(0) = 2$, $\mathcal{B}(1) = 1$, $\mathcal{B}(2) = 1$, and $\mathcal{B}(3) = 0$. $\diamondsuit$

To avoid confusion between sets and multisets, we denote multisets always by uppercase calligraphic letters $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots, \mathcal{Z}$ and standard sets by uppercase roman letters $A, B, C, \ldots, Z$. Sometimes it is convenient to implicitly identify an $E \in \mathscr{P}[X]$ with the multiset $\mathcal{E} \in \mathscr{M}[X]$ satisfying, for all $x \in X$,

$$x \in E \Leftrightarrow \mathcal{E}(x) = 1, \qquad x \notin E \Leftrightarrow \mathcal{E}(x) = 0. \tag{2.1}$$

(We write "$\Leftrightarrow$" as a shorthand for "if and only if.")

Using the multiset formalism, we can now give a precise definition for set systems.

Definition 2.1.4 A *set system* $\mathcal{B}$ over a nonempty set X is a multiset over the set of subsets of X, or briefly $\mathcal{B} \in \mathscr{M}[\mathscr{P}[X]]$.

The following terminology is used in the context of set systems. The elements of X are called *points* of the set system, and the elements $B \in \mathcal{B}$ are called *blocks* of the set system. Two distinct points $x, y \in X$ are *adjacent* if there exists a $B \in \mathcal{B}$ such that $x, y \in B$. For a point $x \in X$ and a block $B \in \mathcal{B}$ we say that x is *incident* to B if $x \in B$.

Informally, two set systems are considered equivalent in structure, or *isomorphic*, if they have an identical structure of point-block incidences irrespective of the concrete identities of the points in the blocks. *Graphs* provide a convenient illustration of this concept before we give the formal definition.

Definition 2.1.5 If each block in a set system has multiplicity one, then the set system is *simple*. A *graph* is a simple set system in which all blocks have cardinality two. It is customary to call the blocks of a graph *edges* and the points *vertices*.

Example 2.1.6 Below is a graph with vertex set $\{0, 1, \ldots, 9\}$.

$$\begin{aligned} G = \{&\{0,1\}, \{0,4\}, \{0,5\}, \{1,2\}, \{1,6\}, \{2,3\}, \{2,7\}, \{3,4\}, \\ &\{3,8\}, \{4,9\}, \{5,7\}, \{5,8\}, \{6,8\}, \{6,9\}, \{7,9\}\}. \end{aligned} \tag{2.2}$$

$\diamondsuit$

Figure 2.1 shows the graph (2.2) in its *labelled* and *unlabelled* form.

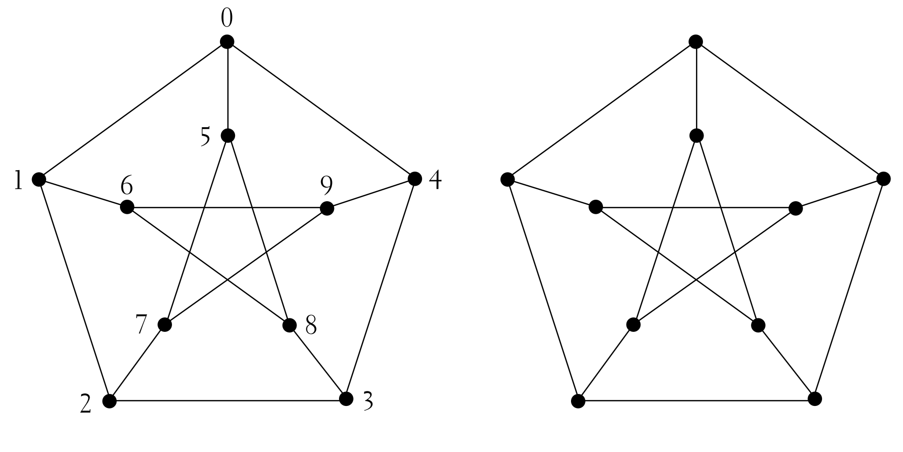


Figure 2.1: A labelled graph and the corresponding unlabelled graph.

The vertex-labelled graph on the left hand side in Figure 2.1 corresponds to the graph (2.2), and the unlabelled graph on the right hand side represents to the structure of point-block incidences present in the labelled graph. The vertex labels are clearly necessary to describe this particular structure of point-block incidences using (2.2), but are otherwise redundant. So, to study the structure of point-block incidences, we shall regard all labelled versions of the same unlabelled structure as equivalent.

The formal definition of isomorphism is as follows. We restrict the consideration to set systems over a fixed finite point set $\mathbb{Z}_v = \{0, 1, \ldots, v-1\}$, and denote by S_v the symmetric group on $\mathbb{Z}_v$.

Definition 2.1.7 Two set systems $\mathcal{B}_1, \mathcal{B}_2 \in \mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$ are *isomorphic* if one can be transformed into the other by a permutation $\sigma \in S_v$ of the points in the blocks.

Example 2.1.8 The set systems $\mathcal{B}_1, \mathcal{B}_2 \in \mathscr{M}[\mathscr{P}[\mathbb{Z}_7]]$,

$$\begin{aligned}\mathcal{B}_1 &= \{\{0,1,2\},\{0,3,4\},\{0,5,6\},\{1,3,5\},\{1,4,6\},\{2,3,6\},\{2,4,5\}\},\\ \mathcal{B}_2 &= \{\{0,1,2\},\{0,3,5\},\{0,4,6\},\{1,3,6\},\{1,4,5\},\{2,3,4\},\{2,5,6\}\}\end{aligned}$$

are isomorphic because $\sigma\mathcal{B}_1 = \mathcal{B}_2$ for $\sigma = (0\ 2\ 1) \in S_7$. (We use the standard cycle notation for permutations, consult Appendix B.) ◊

The somewhat informal saying "by a permutation of the points in the blocks" in Definition 2.1.7 can be made more precise in terms of a *group action*. We define a group action of S_v on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$ by extending the induced action of S_v on $\mathbb{Z}_v$ using the actions given in the following two lemmata. (Our groups act on the left, cf. Definition B.2.26.)

Lemma 2.1.9 *Let G act on X. For all $g \in G$ and $E \subseteq X$, define $gE = \{gx : x \in E\}$. Then, the mapping $(g, E) \mapsto gE$ is a group action of G on $\mathscr{P}[X]$.*

Lemma 2.1.10 *Let G act on X. For all $g \in G$ and $\mathcal{E} \in \mathscr{M}[X]$, define $g\mathcal{E} \in \mathscr{M}[X]$ by the rule $g\mathcal{E} : x \mapsto \mathcal{E}(g^{-1}x)$ for all $x \in X$. Then, the mapping $(g, \mathcal{E}) \mapsto g\mathcal{E}$ is a group action of G on $\mathscr{M}[X]$.*

Henceforth we will speak simply of the (induced) action of S_v on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$, although naturally what is meant is the extension of the induced action.

Isomorphism is clearly an equivalence relation on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$. The isomorphism equivalence classes of set systems in $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$ have a precise characterization in the group action framework, where they are the *orbits* of the induced action of S_v on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$.

Definition 2.1.11 Two set systems over $\mathbb{Z}_v$ are *isomorphic* if they are on the same orbit of the induced action of S_v on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$. We write $S_v \setminus\!\!\setminus \mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$ for the set of all orbits of S_v on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$.

Example 2.1.12 Figure 2.2 shows all 11 unlabelled graphs on four vertices. These correspond to the 11 orbits of the induced action of S_4 on the set of graphs over $\{0, 1, 2, 3\}$. ◊

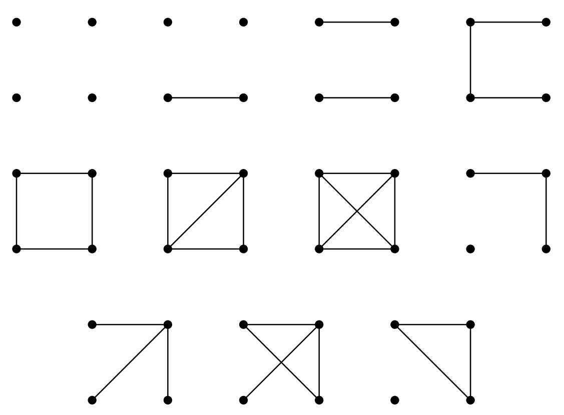


Figure 2.2: The 11 unlabelled graphs on four vertices.

The group action framework gives us another useful concept: the stabilizer subgroup (cf. Definition B.2.32).

Definition 2.1.13 The stabilizer subgroup of a $\mathcal{B} \in \mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$ in S_v is in the context of set systems called the *(full) automorphism group* of $\mathcal{B}$ and denoted by $\operatorname{Aut}(\mathcal{B})$. A permutation $\sigma \in \operatorname{Aut}(\mathcal{B})$ is called an *automorphism* of $\mathcal{B}$.

Knowledge of a single representative $\mathcal{B}$ from an orbit and of the corresponding stabilizer subgroup $\operatorname{Aut}(\mathcal{B})$ is sufficient for straightforward construction of the entire orbit as follows: compute a left transversal of $\operatorname{Aut}(\mathcal{B})$ in S_v and apply the transversal to $\mathcal{B}$ to construct the distinct elements in the orbit (cf. Theorem B.2.33). In particular, the cardinality of the orbit of $\mathcal{B}$ can be determined as $[S_v : \operatorname{Aut}(\mathcal{B})] = |S_v|/|\operatorname{Aut}(\mathcal{B})|$. For example, $|\operatorname{Aut}(\mathcal{B}_1)| = 168$ in Example 2.1.8. Consequently, there are $7!/168 = 5040/168 = 30$ set systems in the orbit of $\mathcal{B}_1$ in $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$. These set systems can be constructed by computing a left transversal of

$$\operatorname{Aut}(\mathcal{B}_1) = \langle (0\ 4\ 5\ 6\ 1\ 3\ 2), (0\ 5\ 1\ 4)(3\ 6)\rangle$$

in S_7. (See [58, Sec. 5] for an efficient algorithm for computing transversals of subgroups in S_v.)

2.2 BLOCK DESIGNS

Definition 2.2.1 A set system $\mathcal{B} \in \mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$ is called a *block design* with parameters $v, k, \lambda \in \mathbb{N} \setminus \{0\}$ (briefly, a $B(v, k, \lambda)$ design) if all blocks of $\mathcal{B}$ have cardinality k and each 2-subset of $\mathbb{Z}_v$ occurs with total multiplicity λ in the blocks.

Example 2.2.2 The unique $B(7, 3, 1)$ design (up to isomorphism) is known as the *Fano plane*:

$$\{\{0,1,2\},\{0,3,4\},\{0,5,6\},\{1,3,5\},\{1,4,6\},\{2,3,6\},\{2,4,5\}\}. \quad (2.3)$$

A graphic representation of the Fano plane is given in Figure 2.3, from which it is easy to verify that each 2-subset of $\{0, 1, 2, 3, 4, 5, 6\}$ occurs with total multiplicity one in the blocks. ♢

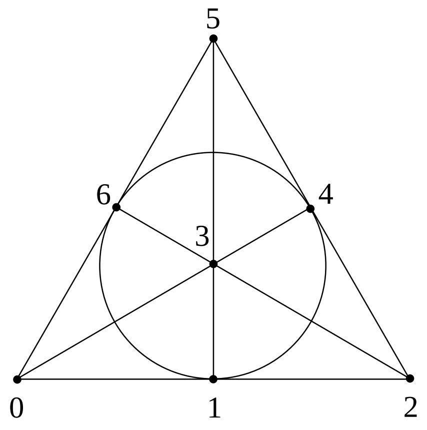


Figure 2.3: The Fano plane.

We remark that block designs as defined above are also often called *balanced incomplete block designs* (briefly *BIBDs*). Our convention of calling them simply block designs is from [4].

Definition 2.2.3 We write $\mathscr{B}(v, k, \lambda)$ for the set of all $B(v, k, \lambda)$ designs in $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$.

The notion of isomorphism for block designs is inherited from set systems. It is clear that the conditions in Definition 2.2.1 that characterize a block design do not depend on the identities of the points in the blocks, that is, if we permute the points in the blocks of a block design, the resulting set system is still a block design. Consequently, the set $\mathscr{B}(v, k, \lambda)$ is a union of orbits of S_v on $\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]$, and we may restrict the induced action of S_v to $\mathscr{B}(v, k, \lambda)$ when necessary.

We derive next some well-known necessary existence conditions for $B(v, k, \lambda)$ designs.

Theorem 2.2.4 *In any $B(v, k, \lambda)$ design each point occurs in $\lambda(v-1)/(k-1)$ blocks, and the total number of blocks is $\lambda v(v-1)/(k(k-1))$.*

Proof. Consider an arbitrary $B(v, k, \lambda)$ design, and fix any $x \in \mathbb{Z}_v$. Clearly, x is an element in exactly $v-1$ 2-subsets of $\mathbb{Z}_v$. Because any such pair occurs with multiplicity λ in the blocks, the total number of such containments of pairs in blocks is $\lambda(v-1)$. But any block that contains x must contain $k-1$ other elements as well, and hence exactly $k-1$ pair-block containments occur per block. Thus, the number of blocks that contain x is $\lambda(v-1)/(k-1)$. Denote by b the total number of blocks, and count the total number of occurences of points in blocks in two ways. A block-by-block count gives bk, and a point-by-point count gives $\lambda v(v-1)/(k-1)$. Consequently, $b = \lambda v(v-1)/(k(k-1))$. ■

If we denote by b the total number of blocks of a $B(v, k, \lambda)$ design and by r the number of blocks in which a point occurs, we obtain as a corollary:

$$vr = bk, \qquad \lambda(v-1) = r(k-1). \tag{2.4}$$

Additionally, since r and b must both be integers, we obtain a necessary existence condition:

Corollary 2.2.5 *There exists a $B(v,k,\lambda)$ design only if* $\lambda(v-1) \equiv 0 \pmod{k-1}$ *and* $\lambda v(v-1) \equiv 0 \pmod{k(k-1)}$.

Fisher's inequality is a well-known necessary existence condition for block designs when v is small in relation to k, λ:
Theorem 2.4.11 *A $B(v,k,\lambda)$ design with $k < v$ exists only if $b \geq v$.*
(A proof can be found in Section 2.4.)
Fisher's inequality can be strengthened if we assume that the design contains two disjoint blocks. (The inequality (2.5) is known as *Bose's condition*.)

Theorem 2.2.6 *A $B(v,k,\lambda)$ design with two disjoint blocks exists only if*

$$b \geq v + r - 1. \tag{2.5}$$

Proof. See [4, Corollary 8.6]. ■

A fundamental problem in design theory is to determine *sufficient* existence conditions for block designs. This typically involves giving an explicit construction for the designs, although asymptotic existence results do exist (see for example [4, Ch. XI]). As an example of an infinite family of block designs, we consider the family of Steiner triple systems.

Definition 2.2.7 A $B(v,3,1)$ design is called a *Steiner triple system* of order v, or briefly an $STS(v)$.

The constructions for $STS(6j+3)$ and $STS(6j+1)$ given below show that the necessary existence condition $v \equiv 1,3 \pmod 6$ from Corollary 2.2.5 is sufficient for Steiner triple systems. (The constructions are from [117, Ch. 19].)

Example 2.2.8 *Construction for an $STS(6j+3)$.* Let $n = 2j+1$. For convenience we use $\mathbb{Z}_n \times \mathbb{Z}_3$ as the point set. The blocks consist of all triples of the form $\{(x,0),(x,1),(x,2)\}$ with $x \in \mathbb{Z}_n$ and all triples $\{(x,i),(y,i),((x+y)(j+1),i+1)\}$ with $x,y \in \mathbb{Z}_n$, $x \neq y$, and $i \in \mathbb{Z}_3$. (Addition and multiplication is coordinatewise modulo n and 3.) ♢

Example 2.2.9 *Construction for an $STS(6j+1)$.* We take as point set $\mathbb{Z}_{2j} \times \mathbb{Z}_3 \cup \{\infty\}$. The $6j+1$ *base blocks* are:

$$\{(0,0),(0,1),(0,2)\}; \tag{2.6}$$
$$\{\infty,(0,0),(j,1)\},\ \{\infty,(0,1),(j,2)\},\ \{\infty,(0,2),(j,0)\}; \tag{2.7}$$
$$\{(0,0),(i,1),(-i,1)\},\ \{(0,1),(i,2),(-i,2)\},\ \{(0,2),(i,0),(-i,0)\}; \tag{2.8}$$
$$\{(j,0),(i,1),(1-i,1)\},\ \{(j,1),(i,2),(1-i,2)\},\ \{(j,2),(i,0),(1-i,0)\}, \tag{2.9}$$

where the blocks (2.8) are included once for every $i = 1,\dots,j-1$ and (2.9) for every $i = 1,\dots,j$. The block set is now constructed by adding the element $(l,0)$ to each of the $6j+1$ base blocks for all $l = 0,\dots,j-1$. (Addition of elements is coordinatewise modulo $2j$ and 3 with $\infty + (x,i) = (x,i) + \infty = \infty$ for all $(x,i) \in \mathbb{Z}_{2j} \times \mathbb{Z}_3$.) ♢

These two constructions solve the Steiner triple system existence problem. The next step is of course to classify all the nonisomorphic $STS(v)$. This, however, is not an easy problem because the number of nonisomorphic designs increases very rapidly with increasing v by a result of Wilson [119]:

Theorem 2.2.10 *The number of nonisomorphic $STS(v)$ is at least $(e^{-5}v)^{v^2/6}$ for all $v \equiv 1, 3 \pmod 6$.*

It is known that there are 2 nonisomorphic $STS(13)$ and 80 nonisomorphic $STS(15)$ [78, 79], however, already the number of nonisomorphic $STS(19)$ is unknown, but estimated to be between 1.1×10^{10} and 1.2×10^{10} [84]. (See [22] for further reference on triple systems.)

To illustrate nonisomorphic designs, we give a complete classification of the nonisomorphic $B(8, 4, 3)$ designs, that is, a transversal of orbits $S_8 \setminus\!\!\setminus \mathscr{B}(8, 4, 3)$. Altogether there are four orbits of S_8 on $\mathscr{B}(8, 4, 3)$. Example 2.2.11 lists a collection of representatives for the four orbits and gives generators for their full automorphism groups.

Example 2.2.11 The lexicographic minimum representatives of the orbits of S_8 on $\mathscr{B}(8, 4, 3)$ are:

$$\begin{aligned}
\mathcal{B}_1 = \{&\{0,1,2,3\},\{0,1,2,4\},\{0,1,5,6\},\{0,2,5,7\},\{0,3,4,5\},\\
&\{0,3,6,7\},\{0,4,6,7\},\{1,2,6,7\},\{1,3,4,6\},\{1,3,5,7\},\\
&\{1,4,5,7\},\{2,3,4,7\},\{2,3,5,6\},\{2,4,5,6\}\},\\
\mathcal{B}_2 = \{&\{0,1,2,3\},\{0,1,2,4\},\{0,1,5,6\},\{0,2,5,7\},\{0,3,4,5\},\\
&\{0,3,6,7\},\{0,4,6,7\},\{1,2,6,7\},\{1,3,4,7\},\{1,3,5,6\},\\
&\{1,4,5,7\},\{2,3,4,6\},\{2,3,5,7\},\{2,4,5,6\}\}\\
\mathcal{B}_3 = \{&\{0,1,2,3\},\{0,1,2,4\},\{0,1,5,6\},\{0,2,5,7\},\{0,3,4,7\},\\
&\{0,3,5,6\},\{0,4,6,7\},\{1,2,6,7\},\{1,3,4,5\},\{1,3,6,7\},\\
&\{1,4,5,7\},\{2,3,4,6\},\{2,3,5,7\},\{2,4,5,6\}\}\\
\mathcal{B}_4 = \{&\{0,1,2,3\},\{0,1,4,5\},\{0,1,6,7\},\{0,2,4,6\},\{0,2,5,7\},\\
&\{0,3,4,7\},\{0,3,5,6\},\{1,2,4,7\},\{1,2,5,6\},\{1,3,4,6\},\\
&\{1,3,5,7\},\{2,3,4,5\},\{2,3,6,7\},\{4,5,6,7\}\}.
\end{aligned}$$

The corresponding full automorphism groups and their orders are:

$$\begin{aligned}
\operatorname{Aut}(\mathcal{B}_1) &= \langle (1\,7)(2\,6), (0\,2\,1)(3\,4)(5\,7\,6)\rangle, & |\operatorname{Aut}(\mathcal{B}_1)| &= 48,\\
\operatorname{Aut}(\mathcal{B}_2) &= \langle (0\,4\,5)(2\,7\,6), (1\,7)(2\,6)\rangle, & |\operatorname{Aut}(\mathcal{B}_2)| &= 12,\\
\operatorname{Aut}(\mathcal{B}_3) &= \langle (1\,4\,6)(2\,7\,5), (0\,5\,2\,6\,7\,4\,1)\rangle, & |\operatorname{Aut}(\mathcal{B}_3)| &= 21,\\
\operatorname{Aut}(\mathcal{B}_4) &= \langle (0\,2\,3\,7\,4\,1\,6), (0\,3\,2\,6\,1\,7\,5)\rangle, & |\operatorname{Aut}(\mathcal{B}_4)| &= 1344.
\end{aligned}$$

$\diamondsuit$

Given this information, we can calculate the total number of $B(8, 4, 3)$ designs in $\mathscr{B}(8, 4, 3)$ using Corollary B.2.34:

$$\begin{aligned}
|\mathscr{B}(8,4,3)| &= \sum_{i=1}^{4} |S_8| / |\operatorname{Aut}(\mathcal{B}_i)| =\\
&= 8! \cdot (1/48 + 1/12 + 1/21 + 1/1344) = 6150.
\end{aligned}$$

Thus, most of the $\mathscr{B}(8,4,3)$ are isomorphic copies of each other, and only four of the $\mathscr{B}(8,4,3)$ suffice to characterize them all.

2.3 RESOLVABILITY AND RESOLUTIONS OF BLOCK DESIGNS

Resolvability is a property of a block design which simplifies the data analysis of a statistical experiment conducted according to the block design (see [114]).

Definition 2.3.1 A $B(v,k,\lambda)$ design $\mathcal{B}$ is said to be *resolvable* (briefly, an $RB(v,k,\lambda)$ design) if the blocks in $\mathcal{B}$ can be partitioned into *parallel classes*, each of which partitions the point set $\mathbb{Z}_v$. Such a partition into parallel classes is called a *resolution*.

Example 2.3.2 The $STS(9)$ below is resolvable.

$$\begin{aligned}\mathcal{B} = \{\, &\{0,1,2\},\{0,3,6\},\{0,4,8\},\{0,5,7\}\\ &\{1,4,7\},\{1,3,8\},\{1,5,6\},\{2,3,7\}\\ &\{2,4,6\},\{2,5,8\},\{3,4,5\},\{6,7,8\}\,\}.\end{aligned}$$

The parallel classes of the unique resolution $\mathcal{R} = \{P_1,P_2,P_3,P_4\}$ of $\mathcal{B}$ are:

$$\begin{aligned}P_1 &= \{\{0,1,2\},\{3,4,5\},\{6,7,8\}\}, & P_2 &= \{\{0,3,6\},\{1,4,7\},\{2,5,8\}\},\\ P_3 &= \{\{0,4,8\},\{1,5,6\},\{2,3,7\}\}, & P_4 &= \{\{0,5,7\},\{1,3,8\},\{2,4,6\}\}.\end{aligned}$$

$\diamondsuit$

Example 2.3.3 Recalling the introduction, a solution to Kirkman's problem is a resolution of a resolvable $STS(15)$. In general, a resolvable $STS(v)$ is called a *Kirkman triple system* of order v, or briefly a $KTS(v)$. $\diamondsuit$

Note that we have not yet defined precisely what constitutes a partition of a multi-set, and this is clearly required so that Definition 2.3.1 is unambiguous for block designs that contain a block whose multiplicity exceeds one. For this purpose some additional terminology is required. The *sum* $\mathcal{E}_1 + \mathcal{E}_2$ of multisets $\mathcal{E}_1, \mathcal{E}_2 \in \mathscr{M}[X]$ is the multiset over X defined by the rule $x \mapsto \mathcal{E}_1(x) + \mathcal{E}_2(x)$ for all $x \in X$. For an integer $k \in \mathbb{N}$ and a multiset $\mathcal{E} \in \mathscr{M}[X]$, the *product* $k \cdot \mathcal{E}$ is the multiset over X defined by $x \mapsto k \cdot \mathcal{E}(x)$ for all $x \in X$.

Definition 2.3.4 Let $\mathcal{E} \in \mathscr{M}[X]$ be a multiset. A multiset $\mathcal{P} \in \mathscr{M}[\mathscr{M}[X]]$ is a *partition* of $\mathcal{E}$ if $\sum_{\mathcal{C} \in \mathscr{M}[X]} \mathcal{P}(\mathcal{C}) \cdot \mathcal{C} = \mathcal{E}$ and $\mathcal{P}(\emptyset) = 0$.

So, a $\mathcal{B} \in \mathscr{B}(v,k,\lambda)$ is resolvable if and only if there exists a $\mathcal{R} \in \mathscr{M}[\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]]$ that partitions $\mathcal{B}$ so that each element of $\mathcal{R}$ partitions $\mathbb{Z}_v$.

Definition 2.3.5 We write $\mathscr{RB}(v,k,\lambda)$ for the set of all $RB(v,k,\lambda)$ designs in $\mathscr{B}(v,k,\lambda)$. The set of all resolutions of $RB(v,k,\lambda)$ designs in $\mathscr{M}[\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]]$ is denoted by $\mathscr{R}(v,k,\lambda)$.

Not all $B(v, k, \lambda)$ designs admit a resolution. For example, if a v-set is to be partitioned so that each cell of the partition is a k-set, then it is clear that k must divide v for this to be possible. This observation together with (2.4) leads to the following necessary condition for resolvability.

Theorem 2.3.6 *If a $B(v, k, \lambda)$ design is resolvable, then k divides v. Furthermore, every resolution of an $RB(v, k, \lambda)$ design has exactly r parallel classes, and each parallel class consists of v/k blocks.*

Together with Corollary 2.2.5 this gives:

Corollary 2.3.7 *An $RB(v, k, \lambda)$ design exists only if $\lambda(v-1) \equiv 0 \pmod{k-1}$ and $v \equiv 0 \pmod{k}$.*

Example 2.3.8 The previous corollary gives $v \equiv 3 \pmod{6}$ as a necessary existence condition for a $KTS(v)$. This condition is also sufficient (see [4, Ch. IX] or [102]). $\diamondsuit$

Example 2.3.9 The necessary existence conditions of Corollary 2.3.7 are not always sufficient (although in an asymptotic sense they are, see [4, Ch. XI]). For example, an $RB(15, 5, 4)$ design does not exist, as we shall show later. $\diamondsuit$

Example 2.3.10 Even if an $RB(v, k, \lambda)$ design exists, not all $B(v, k, \lambda)$ designs need to be resolvable. As an example, recall the classification of $B(8, 4, 3)$ designs given in Example 2.2.11. In the example, design $\mathcal{B}_4$ is resolvable and its unique resolution is

$$\begin{aligned}\mathcal{R} = \{&\{\{0,1,2,3\},\{4,5,6,7\}\},\ \{\{0,1,4,5\},\{2,3,6,7\}\},\\ &\{\{0,1,6,7\},\{2,3,4,5\}\},\ \{\{0,2,4,6\},\{1,3,5,7\}\},\\ &\{\{0,2,5,7\},\{1,3,4,6\}\},\ \{\{0,3,4,7\},\{1,2,5,6\}\},\\ &\{\{0,3,5,6\},\{1,2,4,7\}\}\}.\end{aligned}$$

However, designs $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ are not resolvable. (To see this, note that block $\{0, 1, 2, 3\}$ cannot form a parallel class since the aforementioned designs do not contain block $\{4, 5, 6, 7\}$.) $\diamondsuit$

Bose's condition (2.5) gives us an additional necessary existence condition for a resolvable block design.

Theorem 2.3.11 *An $RB(v, k, \lambda)$ design with $k < v$ exists only if $b \geq v + r - 1$.*

Proof. Clearly, if $k < v$, then a parallel class of a resolution must contain at least two disjoint blocks. Thus, Bose's condition applies. $\blacksquare$

A resolution is clearly a labelled structure related to the underlying labelled block design. The following definitions enable the study of the labelling-independent structure present in a resolution. We extend the induced action of S_v on $\mathbb{Z}_v$ using Lemmata 2.1.9 and 2.1.10 to $\mathscr{M}[\mathscr{M}[\mathscr{P}[\mathbb{Z}_v]]]$, and define:

Definition 2.3.12 Two resolutions of $RB(v, k, \lambda)$ designs are *isomorphic* if they are on the same orbit of the induced action of S_v on $\mathscr{R}(v, k, \lambda)$.

Definition 2.3.13 Let $\mathcal{R} \in \mathscr{R}(v,k,\lambda)$. The stabilizer of $\mathcal{R}$ in S_v is called the *(full) automorphism group* of $\mathcal{R}$ and denoted by $\mathrm{Aut}(\mathcal{R})$. A permutation $\sigma \in \mathrm{Aut}(\mathcal{R})$ is called an *automorphism* of $\mathcal{R}$.

We observe that resolvability is an orbit-invariant property on $\mathscr{B}(v,k,\lambda)$:

Lemma 2.3.14 *Let $\mathcal{B} \in \mathscr{RB}(v,k,\lambda)$. Then $\sigma\mathcal{B} \in \mathscr{RB}(v,k,\lambda)$ for all $\sigma \in S_v$.*

Proof. If $\mathcal{R}$ is a resolution of $\mathcal{B}$, then $\sigma\mathcal{R}$ is a resolution of $\sigma\mathcal{B}$. ■

It is not hard to see that isomorphism of resolutions is a stronger condition than isomorphism of block designs in the sense that if two resolutions are isomorphic, then the underlying designs are isomorphic. The converse is not true; a resolvable design may have nonisomorphic resolutions (see Example 2.3.22 below).

However, there are two special cases in which a resolvable block design always has a unique resolution.

Theorem 2.3.15 *An $RB(2k,k,\lambda)$ design has a unique resolution. Furthermore, the automorphism groups of the resolution and the underlying block design agree.*

Proof. Every parallel class of a resolution of an $RB(2k,k,\lambda)$ design consists of a block and its complement (cf. Example 2.3.10). Consequently, the resolution is unique, and every automorphism of the design maps parallel classes to parallel classes. ■

Definition 2.3.16 An $RB(v,k,\lambda)$ design is *affine* if the design admits a resolution in which any two blocks belonging to different parallel classes intersect in a constant $\mu > 0$ number of points. The constant μ is called the *intersection parameter* of the design.

Example 2.3.17 The $KTS(9)$ in Example 2.3.2 is affine with $\mu = 1$. ◇

Theorem 2.3.18 *The resolution of an affine $B(v,k,\lambda)$ design is unique. Furthermore, the automorphism groups of the resolution and the underlying block design agree.*

Proof. Consider an arbitrary resolution of an affine design, and select a parallel class of the resolution. Clearly, the blocks in the parallel class are pairwise disjoint. Now consider a resolution that makes the design affine. By definition any two blocks belonging to different parallel classes of the affine resolution have nonempty intersection. Since the arbitrary resolution and the affine resolution consist of the same blocks, it must be that the selected arbitrary parallel class is a parallel class of the affine resolution. Consequently, the resolution that makes the design affine is unique. Furthermore, since any automorphism of the design must map disjoint blocks to disjoint blocks, every automorphism of the design is also an automorphism of the resolution. ■

The following theorem characterizing affine designs is due to Bose [8].

Theorem 2.3.19 *An $RB(v,k,\lambda)$ design with $k < v$ is affine if and only if $b = v + r - 1$. Furthermore, the parameters of an affine design with intersection parameter μ can always be written in the form*

$$v = q^2\mu, \quad k = q\mu, \quad \lambda = \frac{q\mu - 1}{q-1}, \quad r = \frac{q^2\mu - 1}{q-1}, \quad b = qr,$$

where q is a positive integer.

Proof. See [4, Theorem 8.7]. ■

A resolvable design may also have distinct isomorphic resolutions. The following theorem gives a necessary and sufficient condition as to when this is the case.

Theorem 2.3.20 *Let $\mathcal{B} \in \mathscr{RB}(v,k,\lambda)$ and suppose $\mathcal{R} \in \mathscr{R}(v,k,\lambda)$ is a resolution of $\mathcal{B}$. Then, $\mathrm{Aut}(\mathcal{R}) \leq \mathrm{Aut}(\mathcal{B})$, and the number of resolutions of $\mathcal{B}$ isomorphic to $\mathcal{R}$ is $[\mathrm{Aut}(\mathcal{B}) : \mathrm{Aut}(\mathcal{R})]$.*

Proof. Clearly, if $\sigma \in \mathrm{Aut}(\mathcal{R})$, then $\sigma \in \mathrm{Aut}(\mathcal{B})$. Denote by $\mathscr{R}(\mathcal{B})$ the set of all resolutions of $\mathcal{B}$. We note that $\mathrm{Aut}(\mathcal{B})$ acts on $\mathscr{R}(\mathcal{B})$; the orbit of $\mathcal{R}$ on $\mathscr{R}(\mathcal{B})$ under $\mathrm{Aut}(\mathcal{B})$ is precisely the set of resolutions in $\mathscr{R}(\mathcal{B})$ isomorphic to $\mathcal{R}$. Additionally, we note that $\mathrm{Aut}(\mathcal{R})$ is the stabilizer of $\mathcal{R}$ in $\mathrm{Aut}(\mathcal{B})$. Thus, Corollary B.2.34 applies and the orbit of $\mathcal{R}$ under $\mathrm{Aut}(\mathcal{B})$ has cardinality $[\mathrm{Aut}(\mathcal{B}) : \mathrm{Aut}(\mathcal{R})]$. ■

Corollary 2.3.21 *Let $\mathcal{B} \in \mathscr{RB}(v,k,\lambda)$. Then, the number of nonisomorphic resolutions of $\mathcal{B}$ is $|\mathrm{Aut}(\mathcal{B}) \mathbin{\backslash\backslash} \mathscr{R}(\mathcal{B})|$. In particular, if $|\mathrm{Aut}(\mathcal{B})| = 1$, then distinct resolutions of $\mathcal{B}$ are nonisomorphic.*

As an example of a design with multiple isomorphic and nonisomorphic resolutions we consider the $KTS(15)$

$$\begin{aligned}\mathcal{B} = \{&\{0,1,2\},\{3,7,11\},\{4,10,12\},\{5,8,13\},\{6,9,14\},\\ &\{0,3,4\},\{1,7,9\},\{2,11,13\},\{5,10,14\},\{6,8,12\},\\ &\{0,5,6\},\{1,13,14\},\{2,7,10\},\{3,9,12\},\{4,8,11\},\\ &\{0,7,8\},\{1,3,5\},\{2,12,14\},\{4,9,13\},\{6,10,11\},\\ &\{0,9,10\},\{1,11,12\},\{2,4,5\},\{3,8,14\},\{6,7,13\},\\ &\{0,11,14\},\{1,4,6\},\{2,8,9\},\{3,10,13\},\{5,7,12\},\\ &\{0,12,13\},\{1,8,10\},\{2,3,6\},\{4,7,14\},\{5,9,11\}\},\end{aligned}$$

which has two nonisomorphic resolutions. (There are 80 nonisomorphic $STS(15)$, only four of these are resolvable. Three of the four have two nonisomorphic resolutions each, and one has a unique resolution up to isomorphism [79].) The full automorphism group of $\mathcal{B}$ has order 20160 and

$$\begin{aligned}\mathrm{Aut}(\mathcal{B}) = \langle&(0\ 3\ 11\ 9\ 6\ 8\ 1)(2\ 4\ 7\ 5\ 14\ 12\ 10),\\ &(0\ 9\ 12\ 2\ 7\ 11)(3\ 14\ 10)(4\ 6)(5\ 13\ 8)\rangle.\end{aligned}$$

Example 2.3.22 Representatives for the two orbits of $\mathrm{Aut}(\mathcal{B})$ on $\mathscr{R}(\mathcal{B})$ are

$$\begin{aligned}
\mathcal{R}_1 = \{&\{\{0,1,2\},\{3,7,11\},\{4,10,12\},\{5,8,13\},\{6,9,14\}\},\\
&\{\{0,3,4\},\{1,7,9\},\{2,11,13\},\{5,10,14\},\{6,8,12\}\},\\
&\{\{0,5,6\},\{1,13,14\},\{2,7,10\},\{3,9,12\},\{4,8,11\}\},\\
&\{\{0,7,8\},\{1,3,5\},\{2,12,14\},\{4,9,13\},\{6,10,11\}\},\\
&\{\{0,9,10\},\{1,11,12\},\{2,4,5\},\{3,8,14\},\{6,7,13\}\},\\
&\{\{0,11,14\},\{1,4,6\},\{2,8,9\},\{3,10,13\},\{5,7,12\}\},\\
&\{\{0,12,13\},\{1,8,10\},\{2,3,6\},\{4,7,14\},\{5,9,11\}\}\},\\
\mathcal{R}_2 = \{&\{\{0,1,2\},\{3,7,11\},\{4,10,12\},\{5,8,13\},\{6,9,14\}\},\\
&\{\{0,3,4\},\{1,7,9\},\{2,11,13\},\{5,10,14\},\{6,8,12\}\},\\
&\{\{0,5,6\},\{1,11,12\},\{2,8,9\},\{3,10,13\},\{4,7,14\}\},\\
&\{\{0,7,8\},\{1,3,5\},\{2,12,14\},\{4,9,13\},\{6,10,11\}\},\\
&\{\{0,9,10\},\{1,13,14\},\{2,3,6\},\{4,8,11\},\{5,7,12\}\},\\
&\{\{0,11,14\},\{1,8,10\},\{2,4,5\},\{3,9,12\},\{6,7,13\}\},\\
&\{\{0,12,13\},\{1,4,6\},\{2,7,10\},\{3,8,14\},\{5,9,11\}\}\}.
\end{aligned}$$

The corresponding full automorphism groups are (both groups have order 168):

$$\begin{aligned}
\mathrm{Aut}(\mathcal{R}_1) &= \langle(0\ 6\ 12)(1\ 14\ 10)(2\ 9\ 4)(5\ 8\ 13),\\
&\qquad (0\ 7)(1\ 10)(3\ 6)(4\ 13)(5\ 11)(12\ 14)\rangle,\\
\mathrm{Aut}(\mathcal{R}_2) &= \langle(0\ 7\ 12\ 6)(1\ 11\ 10\ 9)(2\ 3\ 4\ 14)(5\ 8),\\
&\qquad (0\ 10\ 4)(1\ 5\ 11)(2\ 14\ 8)(3\ 9\ 12)\rangle.
\end{aligned}$$

◊

From this information we can calculate that there are $[S_{15} : \mathrm{Aut}(\mathcal{B})] = 64864800$ $KTS(15)$ in $\mathscr{B}(15,3,1)$ isomorphic to $\mathcal{B}$. Each of these has $[\mathrm{Aut}(\mathcal{B}) : \mathrm{Aut}(\mathcal{R}_1)] = 120$ resolutions isomorphic to $\mathcal{R}_1$. Thus, the total number of resolutions isomorphic to $\mathcal{R}_1$ in $\mathscr{R}(15,3,1)$ is $64864800 \cdot 120 = 7783776000$.

2.4 INCIDENCE SYSTEMS

It is sometimes more convenient to further explicate the point-block incidence structure of a set system by labelling the blocks. The resulting *incidence systems* are more suitable for algorithmic construction and they allow the use of tools from linear algebra to study block designs via the associated incidence matrix representation.

The definition for an incidence system presented below is motivated by the following idea. Suppose we label the blocks of a set system $\mathcal{B} \in \mathscr{M}_b[\mathscr{P}[\mathbb{Z}_v]]$ of cardinality b as $\mathcal{B} = \{B_0, B_1, \ldots, B_{b-1}\}$. Then, the point-block incidences of $\mathcal{B}$ subject to this labelling are completely described by the mapping $A : \mathbb{Z}_v \times \mathbb{Z}_b \to \mathbb{Z}_2$ defined by

$$A(i,j) = \begin{cases} 1 & \text{if } i \in B_j\text{; and} \\ 0 & \text{otherwise,} \end{cases} \tag{2.10}$$

for all $(i, j) \in \mathbb{Z}_v \times \mathbb{Z}_b$.

Definition 2.4.1 A $v \times b$ *incidence system* is a mapping $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$.

In the situation above, we say that the incidence system A *corresponds* to the set system $\mathcal{B}$.

(We warn the reader that our definition for an incidence system is somewhat nonstandard in the sense that typically an incidence system is defined as the relation $R = \{(i, j) \in P \times B \ : \ i \text{ is incident to } j\}$, where P and B label the points and blocks, respectively.)

It is clear that any set system in $\mathscr{M}_b[\mathscr{P}[\mathbb{Z}_v]]$ admits description as a $v \times b$ incidence system and vice versa, however, the correspondence is not bijective. (A set system with at least two distinct blocks has multiple corresponding incidence systems.)

We identify $v \times b$ incidence systems with $v \times b$ matrices over $\mathbb{Z}_2$. The following conventions are used with matrices.

Definition 2.4.2 For positive integers v, b and a nonempty set X, a $v \times b$ *matrix* A over X is an array with v rows and b columns consisting of elements of X. The rows are numbered $0, \ldots, v-1$ from top to bottom, the columns $0, \ldots, b-1$ from left to right. The entry at row i, column j is denoted by $A(i, j)$.

Example 2.4.3 Suppose we label the edges of the graph (2.2) in the order they are listed. The incidence system subject to this labelling of the edges is

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}. \tag{2.11}$$

$\diamondsuit$

Example 2.4.4 If we label the blocks of the Fano plane in Example 2.2.2 in the order they are listed, we obtain the incidence system

$$\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}. \tag{2.12}$$

$\diamondsuit$

The isomorphism equivalence classes of set systems in $\mathscr{M}_b[\mathscr{P}[\mathbb{Z}_v]]$ can be described as orbits of the direct product group $S_v \times S_b$ on $\mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$. First, we observe that permuting the rows of an $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ is equivalent to permuting the points in the blocks of the corresponding set system $\mathcal{B}$. Second, permuting the columns of A corresponds to relabelling the blocks. Consequently, two set systems $\mathcal{B}_1, \mathcal{B}_2 \in \mathscr{M}_b[\mathscr{P}[\mathbb{Z}_v]]$ are isomorphic if and only if any two incidence systems $A_1, A_2 \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ corresponding to $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively, are related by a permutation of the rows and columns.

Permuting the rows and columns of an incidence system can be formulated as a group action. We let the direct product group $S_v \times S_b$ act on $\mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ by $((\sigma, \tau), A) \mapsto (\sigma, \tau)A$, where $(\sigma, \tau) \in S_v \times S_b$, and

$$(\sigma, \tau)A : (i, j) \mapsto A(\sigma^{-1}(i), \tau^{-1}(j)) \qquad (2.13)$$

for all $(i, j) \in \mathbb{Z}_v \times \mathbb{Z}_b$.

Definition 2.4.5 Two $v \times b$ incidence systems are *isomorphic* if they are on the same orbit of $S_v \times S_b$ on $\mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ under the *row and column permuting action* (2.13).

Example 2.4.6 Suppose $\sigma = (0\ 1\ 2\ 3) \in S_7$ and $\tau = (0\ 1) \in S_7$. Then,

$$(\sigma, \tau)\begin{pmatrix} 1&1&1&0&0&0&0\\ 1&0&0&1&1&0&0\\ 1&0&0&0&0&1&1\\ 0&1&0&1&0&1&0\\ 0&0&1&0&1&1&0\\ 0&1&0&0&1&0&1\\ 0&0&1&1&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&1&0&1&0\\ 1&1&1&0&0&0&0\\ 0&1&0&1&1&0&0\\ 0&1&0&0&0&1&1\\ 0&0&1&0&1&1&0\\ 1&0&0&0&1&0&1\\ 0&0&1&1&0&0&1 \end{pmatrix}.$$

This corresponds to

$$\begin{aligned} &\sigma\{\{0,1,2\},\{0,3,5\},\{0,4,6\},\{1,3,6\},\{1,4,5\},\{2,3,4\},\{2,5,6\}\} = \\ &=\{\{0,1,5\},\{1,2,3\},\{1,4,6\},\{0,2,6\},\{2,4,5\},\{0,3,4\},\{3,5,6\}\}. \end{aligned}$$

$\diamondsuit$

We now turn our attention to block designs and their representation as incidence systems. The following lemma characterizes the incidence systems that correspond to $B(v, k, \lambda)$ designs in $\mathscr{M}_b[\mathscr{P}[\mathbb{Z}_v]]$.

Lemma 2.4.7 *Let v, b, r, k, λ be positive integers that satisfy (2.4) and suppose $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$. Then, A corresponds to a $B(v, k, \lambda)$ design if and only if*

(i) $\sum_{i=0}^{v-1} A(i, j) = k$ *for all* $j \in \mathbb{Z}_b$*; and*

(ii) $\sum_{j=0}^{b-1} A(i, j) = r$ *for all* $i \in \mathbb{Z}_v$*; and*

(iii) $\sum_{j=0}^{b-1} A(i_1, j)A(i_2, j) = \lambda$ *for all* $i_1, i_2 \in \mathbb{Z}_v$, $i_1 \neq i_2$.

Proof. Condition (i) states that each block of the set system which corresponds to A has cardinality k. Condition (ii) says that each point occurs exactly r times in the blocks of the corresponding set system. Condition (iii) is equivalent to the last condition of Definition 2.2.1. To see this, note that $A(i_1, j)A(i_2, j) = 1$ if and only if block labelled j contains the 2-subset $\{i_1, i_2\}$. ■

Definition 2.4.8 We write $\mathscr{IB}(v, k, \lambda)$ for the set of all incidence systems in $\mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ that correspond to $B(v, k, \lambda)$ designs.

We observe that conditions (i) and (iii) of Lemma 2.4.7 imply (ii) by Theorem 2.2.4. It is also the case that conditions (ii) and (iii) imply (i):

Corollary 2.4.9 *In Lemma 2.4.7 conditions (ii) and (iii) imply (i).*

Proof. Put $k_j = \sum_{i=0}^{v-1} A(i, j)$ for all $j \in \mathbb{Z}_b$. We will show that $k_j = vr/b$ for all $j \in \mathbb{Z}_b$. From (ii), we obtain

$$\sum_{j=0}^{b-1} k_j = vr \tag{2.14}$$

by counting the points in the blocks in two different ways. Furthermore, condition (iii) gives $\sum_{j=0}^{b-1} \binom{k_j}{2} = \lambda\binom{v}{2}$ by counting the pairs in the blocks in two different ways. Expanding and substituting first (2.14) and then $\lambda(v - 1) = r(vr/b - 1)$, we obtain $\sum_{j=0}^{b-1} k_j^2 = \frac{v^2r^2}{b}$. Now,

$$\sum_{j=0}^{b-1}\big(k_j - \frac{vr}{b}\big)^2 = \frac{v^2r^2}{b} - 2\frac{vr}{b}\sum_{j=0}^{b-1} k_j + \sum_{j=0}^{b-1} k_j^2 = -\frac{v^2r^2}{b} + \sum_{j=0}^{b-1} k_j^2 = 0.$$

Thus, $k_j = vr/b$ for all $j \in \mathbb{Z}_b$. ■

Incidence systems that correspond to block designs can now be characterized by a real-coefficient matrix equation that combines (ii) and (iii):

Corollary 2.4.10 *Let v, b, r, k, λ be positive integers which satisfy (2.4) and suppose $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$. Then A corresponds to a $B(v, k, \lambda)$ design if and only if*

$$AA^T = (r - \lambda)I + \lambda J. \tag{2.15}$$

(A^T denotes the transpose of A, I is the $v \times v$ identity matrix, and J denotes the $v \times v$ matrix with all entries equal to 1. For terminology and reference on linear algebra, we refer the reader to [49].)

Equation (2.15) transforms the block design existence problem into a problem in linear algebra. A simple proof of *Fisher's inequality* can be given in this setting:

Theorem 2.4.11 *A $B(v, k, \lambda)$ design with $k < v$ exists only if $b \geq v$.*

Proof. It suffices to show that the v rows of an $A \in \mathscr{IB}(v, k, \lambda)$ are linearly independent when considered as vectors of a b-dimensional real vector space because then $b \geq v$. Observe that the entries of AA^T are the pairwise inner products of the row vectors in A. The row vectors of A are then linearly independent if and only if $\det(AA^T) \neq 0$ [49, p. 192]. By (2.15) and [16, Lemma 1.12] we have

$$\begin{aligned}\det(AA^T) &= \det((r-\lambda)I + \lambda J) = (r + \lambda(v-1))(r-\lambda)^{v-1} \\ &= rk(r-\lambda)^{v-1},\end{aligned}$$

which is nonzero because $r - \lambda > 0$ by the assumption $v > k$ and (2.4). ■

2.5 RESOLVED INCIDENCE SYSTEMS

In this section we derive a representation for resolutions of block designs and their isomorphism equivalence classes using incidence systems. The main motivation for this representation is that it allows us to prove in the next chapter that resolutions of block designs are equivalent to a family of error-correcting codes.

In what follows we assume that the parameters v, k, λ satisfy the necessary existence conditions of Corollary 2.3.7 for an $RB(v, k, \lambda)$ design and that b, r are determined by (2.4). Furthermore, for notational convenience, we put $q = v/k$. (Recall that k must divide v for a design to be resolvable.) Consequently, since $vr = qkr = bk$, we have $b = qr$.

If we are to represent resolutions of block designs using incidence systems, it is evident that we must introduce additional structure to an incidence system to identify the resolution at hand (cf. Example 2.3.22). A straightforward way to accomplish this is to require the block labelling to present the blocks of every parallel class of a resolution in adjacent columns.

Definition 2.5.1 Suppose that $A \in \mathscr{IB}(v, k, \lambda)$ corresponds to the $B(v, k, \lambda)$ design $\mathcal{B} = \{B_0, B_1, \ldots, B_{qr-1}\}$. We say that A is *resolved* if the blocks
$B_{jq}, B_{jq+1}, \ldots, B_{jq+q-1}$ form a parallel class for all $j \in \mathbb{Z}_r$.

In the situation above, we say that the resolved incidence system A *corresponds* to the resolution $\mathcal{R} = \{P_0, \ldots, P_{r-1}\}$ of $\mathcal{B}$, where $P_j = \{B_{jq}, B_{jq+1}, \ldots, B_{jq+q-1}\}$ for all $j \in \mathbb{Z}_r$.

Example 2.5.2 A resolved incidence system that corresponds to the resolution $\mathcal{R} \in \mathscr{R}(9, 3, 1)$ in Example 2.3.2 is

$$\left(\begin{array}{ccc|ccc|ccc|ccc}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0
\end{array}\right).$$

(The parallel classes are separated by vertical lines for clarity.) ◇

Definition 2.5.3 We write $\mathscr{IR}(v,k,\lambda)$ for the set of all resolved incidence systems in $\mathscr{IB}(v,k,\lambda)$.

Lemma 2.5.4 *An incidence system $A \in \mathscr{IB}(v,k,\lambda)$ is resolved if and only if*

$$\sum_{l=0}^{q-1} A(i, qj+l) = 1 \tag{2.16}$$

for all $(i,j) \in \mathbb{Z}_v \times \mathbb{Z}_r$.

Proof. (We remark that (2.16) is a refinement of condition (ii) in Lemma 2.4.7.) Suppose A corresponds to the $B(v,k,\lambda)$ design $\mathcal{B} = \{B_0, B_1, \ldots, B_{qr-1}\}$. Select a $j \in \mathbb{Z}_r$. By (2.10) blocks $B_{jq}, B_{jq+1}, \ldots, B_{jq+q-1}$ partition $\mathbb{Z}_v$ if and only if every row of A contains a one in exactly one of the columns $jq, jq+1, \ldots, jq+q-1$. Consequently, (2.16) holds for all $(i,j) \in \mathbb{Z}_v \times \mathbb{Z}_r$ if and only if A is resolved. ■

Next we define a group action on $\mathscr{IR}(v,k,\lambda)$ whose orbits correspond to the isomorphism equivalence classes of resolutions, that is, the orbits of S_v on $\mathscr{R}(v,k,\lambda)$. This is accomplished by restricting the row and column permuting action (2.13) so that the resolution described by a resolved set system remains invariant under a permutation of the columns.

Accordingly, we put $\Delta_j = \{jq, jq+1, \ldots, jq+q-1\}$ for all $j \in \mathbb{Z}_r$ and denote by S_Δ the set of all permutations $\tau \in S_{qr}$ that stabilize setwise the partition $\Delta = \{\Delta_0, \Delta_1, \ldots, \Delta_{r-1}\}$ of $\mathbb{Z}_{qr}$. Clearly, $S_\Delta \leq S_{qr}$.

The following lemma shows that S_Δ is isomorphic to the permutation wreath product $S_q \wr S_r$. Recall that elements of the permutation wreath product $S_q \wr S_r$ are ordered pairs (μ, π), where $\pi \in S_r$ and $\mu = (\mu_1, \ldots, \mu_r)$ is an ordered r-tuple of permutations $\mu_j \in S_q$. (See Appendix B for further information on wreath products.)

Lemma 2.5.5 *The group S_Δ is isomorphic to the permutation wreath product $S_q \wr S_r$ under the isomorphism $\Phi : S_\Delta \to S_q \wr S_r$, $\Phi(\tau) = (\mu, \pi)$, where (μ, π) is defined by*

$$\tau(qj+l) = q\pi(j) + \mu_{\pi(j)}(l) \tag{2.17}$$

for all $(j,l) \in \mathbb{Z}_r \times \mathbb{Z}_q$.

Proof. Let $\tau \in S_\Delta$. Observe that $\pi \in S_r$ is determined from $\tau\Delta_j = \Delta_{\pi(j)}$ for all $j \in \mathbb{Z}_r$. (No other choice is possible because then $\tau(qj+l) - q\pi(j) \notin \mathbb{Z}_q$ for some $(j,l) \in \mathbb{Z}_r \times \mathbb{Z}_q$.) Furthermore, μ is determined from $\mu_{\pi(j)}(l) = \tau(qj+l) - q\pi(j)$ for all $(j,l) \in \mathbb{Z}_r \times \mathbb{Z}_q$, so Φ is well-defined and bijective.

It remains to show that Φ is a group homomorphism. Let $\tau, \hat\tau \in S_\Delta$, and suppose $\Phi(\tau) = (\mu, \pi)$, $\Phi(\hat\tau) = (\hat\mu, \hat\pi)$. Then, (2.17) applied twice gives

$$\tau\hat\tau(qj+l) = \tau(q\hat\pi(j) + \hat\mu_{\hat\pi(j)}(l)) = q\pi\hat\pi(j) + \mu_{\pi\hat\pi(j)}\hat\mu_{\hat\pi(j)}(l).$$

Consequently, if we put $(\hat{\hat\mu}, \hat{\hat\pi}) = \Phi(\tau\hat\tau)$, then $\hat{\hat\pi} = \pi\hat\pi$ and $\hat{\hat\mu}_j = \mu_j\hat\mu_{\pi^{-1}(j)}$ for all $j \in \mathbb{Z}_r$. But this is by definition the product of (μ, π) and $(\hat\mu, \hat\pi)$ in $S_q \wr S_r$, so $\Phi(\tau\hat\tau) = \Phi(\tau)\Phi(\hat\tau)$. ■

Corollary 2.5.6 *The inverse of a $\tau \in S_\Delta$ with $\Phi(\tau) = (\mu, \pi)$ satisfies*

$$\tau^{-1}(qj + l) = q\pi^{-1}(j) + \mu_j^{-1}(l) \tag{2.18}$$

for all $(j, l) \in \mathbb{Z}_r \times \mathbb{Z}_q$.

Proof. Substitute (2.18) to (2.17) to obtain the identity mapping. ■

We show next that the orbits of $S_v \times S_\Delta$ on $\mathscr{I}\mathscr{R}(v, k, \lambda)$ correspond to the orbits of S_v on $\mathscr{R}(v, k, \lambda)$. First we have to establish that $\mathscr{I}\mathscr{R}(v, k, \lambda)$ is closed under the row and column permuting action of $S_v \times S_\Delta$.

Lemma 2.5.7 *Let $A \in \mathscr{I}\mathscr{R}(v, k, \lambda)$, and suppose $(\sigma, \tau) \in S_v \times S_\Delta$. Then, $(\sigma, \tau)A \in \mathscr{I}\mathscr{R}(v, k, \lambda)$.*

Proof. Conditions (i)-(iii) of Lemma 2.4.7 are obviously preserved by row and column permutation, so $(\sigma, \tau)A \in \mathscr{I}\mathscr{B}(v, k, \lambda)$. To establish $(\sigma, \tau)A \in \mathscr{I}\mathscr{R}(v, k, \lambda)$, we must verify condition (2.16) of Lemma 2.5.4. By Corollary 2.5.6 and (2.13) we have, for all $(i, j, l) \in \mathbb{Z}_v \times \mathbb{Z}_r \times \mathbb{Z}_q$,

$$(\sigma, \tau)A(i, qj + l) = A(\sigma^{-1}(i), \tau^{-1}(qj + l)) = A(\sigma^{-1}(i), q\pi^{-1}(j) + \mu_j^{-1}(l)),$$

where $\Phi(\tau) = (\mu, \pi)$. Because μ_j is a bijection, we thus have

$$\sum_{l=0}^{q-1} (\sigma, \tau)A(i, qj + l) = \sum_{l=0}^{q-1} A(\sigma^{-1}(i), q\pi^{-1}(j) + l),$$

for all $(i, j) \in \mathbb{Z}_v \times \mathbb{Z}_r$. Consequently, (2.16) holds for $(\sigma, \tau)A$ as well. ■

Definition 2.5.8 We say that two resolved incidence systems $A, A' \in \mathscr{I}\mathscr{R}(v, k, \lambda)$ are *isomorphic* if they are on the same orbit under the row and column permuting action of $S_v \times S_\Delta$.

The following theorem establishes that the isomorphism equivalence classes on $\mathscr{R}(v, k, \lambda)$ and $\mathscr{I}\mathscr{R}(v, k, \lambda)$ are in a bijective correspondence.

Theorem 2.5.9 *Let $\mathcal{R}, \mathcal{R}' \in \mathscr{R}(v, k, \lambda)$ and suppose $A, A' \in \mathscr{I}\mathscr{R}(v, k, \lambda)$ correspond to $\mathcal{R}$ and $\mathcal{R}'$, respectively. Then, $\mathcal{R}$ and $\mathcal{R}'$ are isomorphic if and only if A and A' are isomorphic.*

Proof. Let A correspond to the $B(v, k, \lambda)$ design $\mathcal{B} = \{B_0, B_1, \ldots, B_{qr-1}\}$. By Definition 2.5.1, $\mathcal{R} = \{P_0, P_1, \ldots, P_{r-1}\}$ where $P_j = \{B_{qj+0}, B_{qj+1}, \ldots, B_{qj+q-1}\}$ for all $j \in \mathbb{Z}_r$. Furthermore, suppose that $\mathcal{B}'$, B'_{qj+l}, $\mathcal{R}'$, and P'_j, respectively, denote the analogous objects for A'.

(if) Suppose $A' = (\sigma, \tau)A$ with $(\sigma, \tau) \in S_v \times S_\Delta$. Put $\Phi(\tau) = (\mu, \pi)$. From (2.10) and (2.13) we obtain the chain of equivalences

$$\begin{aligned} i \in B_{qj+l} &\Leftrightarrow A(i, qj + l) = 1 \Leftrightarrow (\sigma, \tau)A(\sigma(i), \tau(qj + l)) = 1 \\ &\Leftrightarrow A'(\sigma(i), \tau(qj + l)) = 1 \Leftrightarrow \sigma(i) \in B'_{\tau(qj+l)} \\ &\Leftrightarrow \sigma(i) \in B'_{q\pi(j)+\mu_{\pi(j)}(l)} \end{aligned} \tag{2.19}$$

for all $(i, j, l) \in \mathbb{Z}_v \times \mathbb{Z}_r \times \mathbb{Z}_q$. In other words, $\sigma B_{qj+l} = B'_{q\pi(j)+\mu_{\pi(j)}(l)}$ for all $(j, l) \in \mathbb{Z}_r \times \mathbb{Z}_q$, which implies $\sigma P_j = P'_{\pi(j)}$ for all $j \in \mathbb{Z}_r$. Consequently, $\sigma\mathcal{R} = \{\sigma P_j\}_{j\in\mathbb{Z}_r} = \{P'_{\pi(j)}\}_{j\in\mathbb{Z}_r} = \{P'_j\}_{j\in\mathbb{Z}_r} = \mathcal{R}'$.

(only if) Let $\mathcal{R}' = \sigma\mathcal{R}$ with $\sigma \in S_v$. We observe that this implies

$$\mathcal{R}(P_j) = \sigma\mathcal{R}(\sigma P_j) = \mathcal{R}'(\sigma P_j) \tag{2.20}$$

for all $j \in \mathbb{Z}_r$. We construct a $\tau \in S_\Delta$ such that $A' = (\sigma, \tau)A$ step by step as follows: Select a $P_j \in \mathcal{R}$. Then, $\sigma P_j \in \mathcal{R}'$ and, consequently, there exists a $j' \in \mathbb{Z}_r$ such that $\sigma P_j = P'_{j'}$. Put $\pi(j) = j'$, and define $\mu_{\pi(j)} \in S_q$ so that

$$\sigma B_{qj+l} = B'_{q\pi(j)+\mu_{\pi(j)}(l)} \tag{2.21}$$

holds for all $l \in \mathbb{Z}_q$. Remove P_j and $P'_{j'}$ from consideration, and continue with the next parallel class in $\mathcal{R}$ until all r parallel classes have been considered. This way we obtain a $\tau \in S_\Delta$ with $\tau(qj + l) = q\pi(j) + \mu_{\pi(j)}(l)$ for all $(j, l) \in \mathbb{Z}_r \times \mathbb{Z}_q$. The construction is well-defined because (2.20) guarantees the existence of a new j' on each step. Moreover, $\sigma P_j = P'_{j'}$ clearly implies

$$\sigma P_j = \{\sigma B_{qj+0}, \dots, \sigma B_{qj+q-1}\} = \{B'_{qj'+0}, \dots, B'_{qj'+q-1}\} = P'_{j'}$$

making step (2.21) well-defined. The constructed $\tau \in S_\Delta$ now satisfies

$$\begin{aligned} A(i, qj + l) = 1 \;&\Leftrightarrow\; i \in B_{qj+l} \;\Leftrightarrow\; \sigma(i) \in B'_{\tau(qj+l)} \\ &\Leftrightarrow\; A'(\sigma(i), \tau(qj + l)) = 1 \end{aligned}$$

for all $(i, j, l) \in \mathbb{Z}_v \times \mathbb{Z}_r \times \mathbb{Z}_q$. Thus, $(\sigma, \tau)A = A'$. ■

3 ERROR-CORRECTING CODES

Often in combinatorics a class of objects has several descriptions. This is particularly true with designs, codes, and graphs; see [16, 117]. In this chapter we describe one such connection discovered by Semakov and Zinov'ev [109] between resolutions of block designs and a certain family of q-ary error-correcting codes. Before discussing this correspondence between codes and resolutions, however, we give a brief introduction to coding-theoretic terminology and discuss code equivalence.

For a proper and thorough introduction to coding theory, consult [53, 75]. Coding theory and its connections with design theory are discussed in [16].

3.1 HAMMING SPACES AND BLOCK CODES

Definition 3.1.1 A nonempty set X together with a mapping $d : X \times X \to \mathbb{N}$ is a *(discrete) metric space* if the mapping d, called the *metric* or *distance function*, has the following properties

(i) $d(x, y) = 0$ if and only if $x = y$; and

(ii) $d(x, y) = d(y, x)$ for all $x, y \in X$; and

(iii) $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$.

Let F_q be a finite set of cardinality $q \geq 2$. Suppose $n \geq 1$, and denote by F_q^n the set of all ordered n-tuples over F_q.

Lemma 3.1.2 *The mapping $d_H : F_q^n \times F_q^n \to \mathbb{N}$ defined by*

$$d_H((x_0, x_1 \dots, x_{n-1}), (y_0, y_1, \dots, y_{n-1})) = |\{j \in \mathbb{Z}_n : x_j \neq y_j\}|$$

for all $x = (x_0, x_1, \dots, x_{n-1}) \in F_q^n$ and $y = (y_0, y_1, \dots, y_{n-1}) \in F_q^n$ is a metric for F_q^n.

Definition 3.1.3 The metric space (F_q^n, d_H) is called the *Hamming space* of dimension n over F_q. The corresponding metric d_H is called the *Hamming metric*.

The elements $x \in F_q^n$ are in the context of coding theory called *codewords*. The components $(x_0, x_1, \dots, x_{n-1})$ of a codeword are called *coordinates*, and the values $x_j \in F_q$ *coordinate values*.

Definition 3.1.4 A q-ary *(block) code* of length n is a nonempty subset of F_q^n.

The Hamming space (F_q^n, d_H) can be made a finite vector space if F_q has the structure of a finite field. In this case we can use tools from linear algebra to manipulate *linear* codes.

Definition 3.1.5 Let F_q be a finite field. A code $C \in \mathscr{P}[F_q^n]$ is *linear* if it forms a subspace of the vector space F_q^n. Otherwise C is *nonlinear*.

The codes we study in this report are typically nonlinear. Therefore, we can make the simplifying assumption $F_q = \mathbb{Z}_q$ without loss of generality.

Definition 3.1.6 The *minimum distance* of a code $C \in \mathscr{P}[\mathbb{Z}_q^n]$ with at least two codewords is the quantity $d(C) = \min\{d_H(x,y) \,:\, x, y \in C,\ x \neq y\}$. A code is *equidistant* if $d(x,y) = d(C)$ for all distinct $x, y \in C$.

The minimum distance of a code is probably the most important parameter of a code in the study of error-correcting codes because it measures the ability of the code to sustain random transmission errors. Error-correcting codes for information transmission are often classified according to length and minimum distance as follows.

Definition 3.1.7 An $(n,d)_q$ *code* is a q-ary code of length n and minimum distance d. An $(n,d)_q$ code with cardinality M is called an $(n,M,d)_q$ *code*.

(To exclude degenerate cases in the previous definition we always implicitly assume $M \geq 2$ and $q \geq 2$.)

A fundamental problem in coding theory is to determine, for fixed parameter values q, n, d, the maximum cardinality of an $(n,d)_q$ code. This maximum is usually denoted by $A_q(n,d)$, and the corresponding codes of maximum cardinality are called *optimal*.

3.2 ED$_m$-CODES

The family of ED$_m$-codes (ED$_m$ – equidistant with maximal distance) was introduced and studied by Semakov and Zinov'ev [109].

Definition 3.2.1 An $(n,M,d)_q$ code is an *ED$_m$-code* if (i) q divides M; and (ii) $\binom{M}{2}d = n\binom{q}{2}(M/q)^2$.

Example 3.2.2 Below is an ED$_m$-code with parameters $n = 4$, $M = 9$, $d = 3$, and $q = 3$:

$$\begin{aligned} C = \{&(0,0,0,0),(0,1,1,1),(0,2,2,2),(1,0,2,1),(1,1,0,2),\\ &(1,2,1,0),(2,0,1,2),(2,1,2,0),(2,2,0,1)\}. \end{aligned}$$

$\diamondsuit$

The family of ED$_m$-codes is interesting from a coding-theoretic point of view because its codes are both equidistant and optimal. These properties can be derived from the *q-ary Plotkin bound* [7, Theorem 3]:

Theorem 3.2.3 *If there exists an $(n,M,d)_q$ code, then*

$$\binom{M}{2}d \leq n \sum_{i=0}^{q-2} \sum_{j=i+1}^{q-1} M_i M_j, \tag{3.1}$$

where $M_i = \lfloor (M+i)/q \rfloor$. If equality holds, the code is equidistant, and the distribution of values in a coordinate is (up to permutation) uniquely given by the values of M_i.

For an ED$_m$-code it is clear by (i) of Definition 3.2.1 that $M_i = M/q$ for all $i \in \mathbb{Z}_q$. Thus, the right hand side of (3.1) becomes $n\binom{q}{2}(M/q)^2$, which together with (ii) implies that equality holds in the Plotkin bound.

Corollary 3.2.4 *An ED$_m$-code is equidistant, and every coordinate value occurs exactly $k = M/q$ times in every coordinate of the code.*

Establishing optimality of an ED$_m$-code is somewhat more involved, because mere equality in (3.1) is not sufficient to establish optimality. (As an example, if $q = 3$, $n = 14$, and $d = 10$, then equality holds for both $M = 14$ and $M = 15$.)

Corollary 3.2.5 *An ED$_m$-code is optimal.*

Proof. Let n, M, d, q constitute the parameters of an ED$_m$-code. Because codewords can always be deleted from a code—and the minimum distance can only increase in the process—it suffices to show an $(n, d)_q$ code of cardinality $M + 1$ is impossible. Suppose, for the sake of contradiction, that such a code C exists. Then, we can delete any codeword $x \in C$ and obtain an ED$_m$-code C', which is equidistant and each coordinate value occurs exactly M/q times in a coordinate by Corollary 3.2.4. Clearly, the total Hamming distance $\sum_{y_1 \in C} \sum_{y_2 \in C} d_H(y_1, y_2)$ of C must be at least $\binom{M+1}{2}d$. Because C' is equidistant, its total Hamming distance is exactly $\binom{M}{2}d$, which forces $\binom{M+1}{2}d - \binom{M}{2}d = Md \le \sum_{y \in C'} d_H(x, y)$. Because each coordinate value occurs exactly M/q times in a coordinate in the ED$_m$-code, we calculate $\sum_{y \in C'} d_H(x, y) = n(q - 1)M/q$. By condition (ii) of Definition 3.2.1, we have $(M - 1)d = n(q - 1)M/q$, which implies $Md > n(q - 1)M/q$. Thus, C cannot have minimum distance d, which is a contradiction. ■

3.3 CODE EQUIVALENCE

In the study of error-correcting codes the notion of code equivalence has several variations. For example, if we are only interested in the error-detecting capability of a code, then it makes sense to regard as equivalent all codes of equal cardinality that have the same collection of pairwise codeword distances. In other words, two codes $C, C' \in \mathscr{P}[\mathbb{Z}_q^n]$ are regarded as equivalent if there exists a bijection $\phi : C \to C'$ satisfying $d_H(x, y) = d_H(\phi(x), \phi(y))$ for all $x, y \in C$. In particular, we observe that all equidistant codes of fixed length and cardinality are equivalent.

Another standard approach to code equivalence is to regard two codes as equivalent if they are related by an isometry of the Hamming space.

Definition 3.3.1 Let (X, d) be a metric space. A mapping $\psi : X \to X$ is an *isometry* of (X, d) if $d(x, y) = d(\psi(x), \psi(y))$ for all $x, y \in X$. We denote by $\mathrm{Iso}(X, d)$ the set of all isometries of (X, d).

In this report we use this latter notion of code equivalence because it regards as inequivalent precisely those ED$_m$-codes that correspond to nonisomorphic resolutions of $RB(v, k, \lambda)$ designs. This connection between isometry equivalence classes of codes and isomorphism equivalence classes of resolutions is best described if we have a group action analogous to those presented in the previous chapter whose orbits correspond to the isometry equivalence classes of codes. In what follows we describe such a group action on $\mathbb{Z}_q^n$.

We begin by noting that the set of all isometries of the Hamming space, $\operatorname{Iso}(\mathbb{Z}_q^n, d_H)$, is a group under composition of isometries.

Theorem 3.3.2 *For a finite metric space* (X, d), $\operatorname{Iso}(X, d) \leq \operatorname{Sym}(X)$.

Proof. Let $\psi \in \operatorname{Iso}(X, d)$. Suppose $\psi(x) = \psi(y)$ for some $x, y \in X$. Then, $0 = d(\psi(x), \psi(y)) = d(x, y)$. Thus, $x = y$, and ψ is injective since x, y were arbitrary. Now, since X is finite and $\psi : X \to X$, ψ must be surjective as well. Thus, $\psi \in \operatorname{Sym}(X)$. Two isometries clearly compose to form an isometry, which by Theorem B.2.11 is sufficient to establish the claim. ∎

Definition 3.3.3 Two codes $C_1, C_2 \in \mathscr{P}[\mathbb{Z}_q^n]$ are *equivalent* if they are on the same orbit of the induced action of $\operatorname{Iso}(\mathbb{Z}_q^n, d_H)$ on $\mathscr{P}[\mathbb{Z}_q^n]$.

Example 3.3.4 The mapping $\psi : \mathbb{Z}_2^3 \to \mathbb{Z}_2^3$ defined by $\psi(x_0, x_1, x_2) = (x_2, x_1, x_0)$ for all $x \in \mathbb{Z}_2^3$ is obviously an isometry of $(\mathbb{Z}_2^3, d_H)$ (cf. Lemma 3.3.6). So, for example $C = \{(0, 0, 1), (1, 1, 0)\}$ and $C' = \{(1, 0, 0), (0, 1, 1)\}$ are equivalent because $C' = \psi C$. ◊

The group $\operatorname{Iso}(\mathbb{Z}_q^n, d_H)$ is still too abstract for our purpose of demonstrating that ED_m-codes and resolutions are equivalent. By a result of Ho [52], the group $\operatorname{Iso}(\mathbb{Z}_q^n, d_H)$ is isomorphic to the permutation wreath product $S_q \wr S_n$; unfortunately, we were unable to verify the proof given in [52]. (An independent graph-theoretic proof appears in [120].)

The two lemmata below give two families of isometries of $(\mathbb{Z}_q^n, d_H)$ which together generate $\operatorname{Iso}(\mathbb{Z}_q^n, d_H)$. The first family of isometries permutes the coordinate values in every coordinate of a codeword separately. For convenience in what follows, we identify a codeword $x = (x_0, x_1, \ldots, x_{n-1}) \in \mathbb{Z}_q^n$ in the obvious way with the mapping $x : \mathbb{Z}_n \to \mathbb{Z}_q$ defined by $x(i) = x_i$ for all $i \in \mathbb{Z}_n$.

Lemma 3.3.5 *Let* $\mu \in S_q^n$. *The mapping* $\bar{\mu} : \mathbb{Z}_q^n \to \mathbb{Z}_q^n$, *defined by* $\bar{\mu}(x) : i \mapsto \mu_i(x(i))$ *for all* $i \in \mathbb{Z}_n$ *and* $x \in \mathbb{Z}_q^n$, *is an isometry of* $(\mathbb{Z}_q^n, d_H)$.

Proof. Let $\mu \in S_q^{\mathbb{Z}_n}$ and suppose $x, y \in \mathbb{Z}_q^n$. By definition of d_H we have $d_H(x, y) = |\{i \in \mathbb{Z}_n : x(i) \neq y(i)\}|$. Because μ_i is a bijection for all $i \in \mathbb{Z}_n$, we have $\mu_i(x(i)) \neq \mu_i(y(i))$ if and only if $x(i) \neq y(i)$. Thus, $d_H(x, y) = |\{i \in \mathbb{Z}_n : \mu_i(x(i)) \neq \mu_i(y(i))\} = d_H(\bar{\mu}(x), \bar{\mu}(y))$. ∎

The second family permutes the coordinates of a codeword.

Lemma 3.3.6 *Let* $\pi \in S_n$. *The mapping* $\bar{\pi} : \mathbb{Z}_q^n \to \mathbb{Z}_q^n$, *defined by* $\bar{\pi}(x) : i \mapsto x(\pi^{-1}(i))$ *for all* $i \in \mathbb{Z}_n$ *and* $x \in \mathbb{Z}_q^n$, *is an isometry of* $(\mathbb{Z}_q^n, d_H)$.

Proof. Let $\pi \in S_n$ and suppose $x, y \in \mathbb{Z}_q^n$. Since π is a bijection, we have $d_H(x, y) = |\{i \in \mathbb{Z}_n : x(\pi^{-1}(i)) \neq y(\pi^{-1}(i))\}| = d_H(\bar{\pi}(x), \bar{\pi}(y))$. ∎

We define a group action of $S_q \wr S_n$ on $\mathbb{Z}_q^n$ by letting $(\mu, \pi) \in S_q^n \times S_n$ act on $x \in \mathbb{Z}_q^n$ first by coordinate permutation (Lemma 3.3.6) and then by permutation in the coordinate values (Lemma 3.3.5). The following theorem gives the details of the action.

Theorem 3.3.7 *The mapping $((\mu,\pi),x) \mapsto (\mu,\pi)x$ defined by $(\mu,\pi)x = \bar{\mu}(\bar{\pi}(x))$ for all $(\mu,\pi) \in S_q \wr S_n$ and $x \in \mathbb{Z}_q^n$ is a group action of $S_q \wr S_n$ on $\mathbb{Z}_q^n$.*

Proof. Condition (i) of Definition B.2.26 is clearly satisfied. For condition (ii), select $(\mu,\pi),(\tilde{\mu},\tilde{\pi}) \in S_q \wr S_n$ and $x \in \mathbb{Z}_q^n$. For convenience, put $\tilde{x} = (\tilde{\mu},\tilde{\pi})x$. Compose the mappings in Lemmata 3.3.5 and 3.3.6 to obtain $\tilde{x}(i) = \tilde{\mu}_i(x(\tilde{\pi}^{-1}(i)))$ for all $i \in \mathbb{Z}_n$. Furthermore,

$$(\mu,\pi)\tilde{x}(i) = \mu_i(\tilde{x}(\pi^{-1}(i))) = \mu_i\tilde{\mu}_{\pi^{-1}(i)}(x(\tilde{\pi}^{-1}\pi^{-1}(i))). \tag{3.2}$$

The product $(\tilde{\tilde{\mu}},\tilde{\tilde{\pi}}) = (\mu,\pi)(\tilde{\mu},\tilde{\pi})$ is defined by $\tilde{\tilde{\pi}} = \pi\tilde{\pi}$ and $\tilde{\tilde{\mu}}_i = \mu_i\tilde{\mu}_{\pi^{-1}(i)}$ for all $i \in \mathbb{Z}_n$. Hence,

$$(\tilde{\tilde{\mu}},\tilde{\tilde{\pi}})x(i) = \mu_i\tilde{\mu}_{\pi^{-1}(i)}(x(\tilde{\pi}^{-1}\pi^{-1}(i))). \tag{3.3}$$

The right hand sides of (3.2) and (3.3) are identical, which implies (*) in

$$\big((\mu,\pi)(\tilde{\mu},\tilde{\pi})\big)x = (\tilde{\tilde{\mu}},\tilde{\tilde{\pi}})x \overset{(*)}{=} (\mu,\pi)\tilde{x} = (\mu,\pi)\big((\tilde{\mu},\tilde{\pi})x\big).$$

Thus, condition (ii) holds and the proof is complete. ■

We next argue that $\mathrm{Iso}(\mathbb{Z}_q^n, d_H)$ is in fact the image of the faithful permutation representation that corresponds (cf. Theorem B.2.27) to the action of $S_q \wr S_n$ given in the previous theorem. (To see that the representation is faithful, suppose that $(\mu,\pi)x = x$ for all $x \in \mathbb{Z}_q^n$. In particular, this holds for all codewords that contain the same coordinate value in every coordinate. Consequently, we must have $\mu = 1$, and then clearly also $\pi = 1$ because all codewords that contain exactly one occurence of some coordinate value must be fixed. Thus, the kernel of the representation is trivial and the representation injective.)

It is clear from Lemmata 3.3.5 and 3.3.6 that the mapping $x \mapsto (\mu,\pi)x = \bar{\mu}(\bar{\pi}(x))$ is an isometry for all $(\mu,\pi) \in S_q \wr S_n$. Naturally, it is not so evident that all isometries in $\mathrm{Iso}(\mathbb{Z}_q^n, d_H)$ are of this form. This is the content of the following theorem.

Theorem 3.3.8 *Let $\psi \in \mathrm{Iso}(\mathbb{Z}_q^n, d_H)$. Then, there exists a $(\mu,\pi) \in S_q \wr S_n$ such that $\psi(x) = (\mu,\pi)x$ for all $x \in \mathbb{Z}_q^n$.*

Proof. See [26, Theorem 1]; an earlier graph-theoretic proof appears in [120, Theorem 1]. ■

The permutation representation of the action in Theorem 3.3.7 restricted to
$\mathrm{Iso}(\mathbb{Z}_q^n, d_H)$ is thus an isomorphism between $S_q \wr S_n$ and $\mathrm{Iso}(\mathbb{Z}_q^n, d_H)$. This leads immediately to the following theorem.

Theorem 3.3.9 *Let $S_q \wr S_n$ act on $\mathbb{Z}_q^n$ as in Theorem 3.3.7, and suppose that this action is extended to $\mathscr{P}[\mathbb{Z}_q^n]$ using Lemma 2.1.9. Then, two codes are equivalent if and only if they are on the same orbit of $S_q \wr S_n$ on $\mathscr{P}[\mathbb{Z}_q^n]$.*

As a special case, the previous action can be restricted to the set of all $(n,M,d)_q$ codes because it clearly preserves code cardinality and minimum distance.

Definition 3.3.10 We denote by $\mathscr{C}_q(n,M,d)$ the set of all $(n,M,d)_q$ codes in $\mathscr{P}[\mathbb{Z}_q^n]$.

3.4 MATRIX REPRESENTATION FOR CODES

A code $C \in \mathscr{P}_M[\mathbb{Z}_q^n]$ can be represented using a $M \times n$ matrix over $\mathbb{Z}_q$ whose rows consist of the codewords of C. Conversely, the rows of any matrix $C \in \mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_q}$ whose rows are pairwise distinct clearly represent the codewords of a q-ary code of length n and cardinality M.

This representation of a code in terms of a matrix is useful in the next section where explicit labelling of the codewords is required to describe the correspondence between ED$_m$-codes and resolutions. Furthermore, it will prove useful later in describing the algorithms that generate resolutions of block designs in Chapter 5.

We let the direct product group $S_M \times (S_q \wr S_n)$ act on the set of matrices $\mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n}$ so that $\sigma \in S_M$ permutes the rows of a matrix and $(\mu, \pi) \in S_q \wr S_n$ acts on the rows of a matrix as on codewords in Theorem 3.3.7. Formally, for a $C \in \mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n}$ and a $(\sigma, (\mu, \pi)) \in S_M \times (S_q \wr S_n)$, we define

$$(\sigma, (\mu, \pi))C : (i, j) \mapsto \mu_j(C(\sigma^{-1}(i), \pi^{-1}(j))) \tag{3.4}$$

for all $(i, j) \in \mathbb{Z}_M \times \mathbb{Z}_n$.

The orbits of matrices with pairwise distinct rows in $\mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n}$ under (3.4) are clearly in a bijective correspondence to the isometry equivalence classes on $\mathscr{P}_M[\mathbb{Z}_q^n]$.

Definition 3.4.1 We denote by $\mathscr{L}\mathscr{C}_q(n, M, d)$ the set of all $M \times n$ matrices over $\mathbb{Z}_q$ whose rows form an $(n, M, d)_q$ code.

In particular, there is a bijective correspondence between the orbits $S_q \wr S_n \backslash\backslash \mathscr{C}_q(n, M, d)$ and $S_M \times (S_q \wr S_n) \backslash\backslash \mathscr{L}\mathscr{C}_q(n, M, d)$.

3.5 ED$_m$-CODES AND RESOLUTIONS OF BLOCK DESIGNS

Semakov and Zinov'ev proved in [109] that resolutions of $RB(v, k, \lambda)$ designs are equivalent to ED$_m$-codes. In this section we describe this result and show that it also extends to isomorphism equivalence classes of resolutions and isometry equivalence classes of codes.

The following two lemmata connect the parameter families of ED$_m$-codes and $RB(v, k, \lambda)$ designs. (The conclusions of both lemmata appear in [109].) We denote by $[a, b]$ the greatest common divisor of $a, b \in \mathbb{N}$.

Lemma 3.5.1 *The parameters $(n, M, d)_q$ of an ED$_m$-code can always be written in the form*

$$M = qk, \qquad n = c \cdot \frac{qk-1}{[q-1, k-1]}, \qquad d = n - c \cdot \frac{k-1}{[q-1, k-1]}, \tag{3.5}$$

where k and c are positive integers.

Proof. Since q divides M, k is uniquely determined by $M = qk$. Condition (ii) of Definition 3.2.1 is thus equivalent to

$$(qk - 1)d = k(q - 1)n. \tag{3.6}$$

From elementary number theory we know that all integer solutions to (3.6) can be written in the form

$$d = c \cdot \frac{k(q-1)}{[k(q-1), qk-1]}, \qquad n = c \cdot \frac{qk-1}{[k(q-1), qk-1]},$$

where c is an arbitrary integer. To obtain the claim, observe that

$$[k(q-1), qk-1] = [k(q-1), k-1] = [q-1, k-1]$$

and that

$$d = c \cdot \frac{qk-1+1-k}{[q-1, k-1]} = n - c \cdot \frac{k-1}{[q-1, k-1]}.$$

■

Lemma 3.5.2 *The parameters v, r and λ of an $RB(v, k, \lambda)$ design can always be written in the form*

$$v = qk, \qquad r = c \cdot \frac{qk-1}{[q-1, k-1]}, \qquad \lambda = c \cdot \frac{k-1}{[q-1, k-1]}, \qquad (3.7)$$

where c and q are positive integers.

Proof. Recall that k must divide v for a design to be resolvable. Substitute $v = qk$ to (2.4) to obtain $(qk-1)\lambda = (k-1)r$. The rest of the proof is similar to that of the previous lemma. ■

From Lemmata 3.5.1 and 3.5.2 we see that the parameter families satisfying necessary existence conditions for ED$_m$-codes and $RB(v, k, \lambda)$ designs are connected by

$$n = r = \frac{\lambda(v-1)}{k-1}, \qquad M = v = qk, \qquad d = r - \lambda, \qquad b = qr; \quad (3.8)$$

the exception being the trivial families with $k = v$ and $d = n$, which have no corresponding parameters of the other type.

We next show that the elements of $\mathscr{IR}(v, k, \lambda)$ and $\mathscr{LC}_q(n, M, d)$ with parameters connected by (3.8) are in a bijective correspondence. The following theorem and its corollary are essentially a reformulation of [109, Theorem 1].

Theorem 3.5.3 *Let $\mathscr{IR}(v, k, \lambda)$ be nonempty and suppose that $k < v$. Then, the mapping $\Psi : \mathscr{IR}(v, k, \lambda) \to \mathscr{LC}_q(r, v, r-\lambda)$ defined by*

$$\Psi A(i, j) = l \quad \Leftrightarrow \quad A(i, qj + l) = 1 \qquad (3.9)$$

for all $(i, j, l) \in \mathbb{Z}_v \times \mathbb{Z}_r \times \mathbb{Z}_q$ and $A \in \mathscr{IR}(v, k, \lambda)$ is a bijection.

Proof. We illustrate the mapping Ψ with a short example before proceeding with the proof:

$$\Psi\left(\begin{array}{ccc|ccc|ccc|ccc}
1&0&0&1&0&0&1&0&0&1&0&0\\
1&0&0&0&1&0&0&1&0&0&1&0\\
1&0&0&0&0&1&0&0&1&0&0&1\\
0&1&0&1&0&0&0&0&1&0&1&0\\
0&1&0&0&1&0&1&0&0&0&0&1\\
0&1&0&0&0&1&0&1&0&1&0&0\\
0&0&1&1&0&0&0&1&0&0&0&1\\
0&0&1&0&1&0&0&0&1&1&0&0\\
0&0&1&0&0&1&1&0&0&0&1&0
\end{array}\right)=\left(\begin{array}{cccc}
0&0&0&0\\
0&1&1&1\\
0&2&2&2\\
1&0&2&1\\
1&1&0&2\\
1&2&1&0\\
2&0&1&2\\
2&1&2&0\\
2&2&0&1
\end{array}\right). \qquad (3.10)$$

(In the example above, $v = 9$, $k = 3$, $\lambda = 1$; $r = 4$, $q = 3$.) First we have to check that Ψ is well-defined. Select any $A \in \mathscr{IR}(v, k, \lambda)$. Since A satisfies (2.16), every entry $\Psi A(i, j)$ is uniquely determined by (3.9), so it suffices to check that the code that corresponds to ΨA has minimum distance $r - \lambda$. (Note that $k < v$ implies $\lambda < r$ by (2.4).) Select any two rows of ΨA labelled $i_1, i_2 \in \mathbb{Z}_v$, $i_1 \neq i_2$. We observe that $\Psi A(i_1, j) \neq \Psi A(i_2, j)$ if and only if $A(i_1, qj + l_1) = 1$ and $A(i_2, qj + l_2) = 1$ for some $l_1 \neq l_2$, that is, points i_1 and i_2 occur in different blocks of parallel class j of the resolution that corresponds to A. By (2.16) we therefore have

$$\sum_{l=0}^{q-1} A(i_1, qj + l)A(i_2, qj + l) = \begin{cases} 1 & \text{if } \Psi A(i_1, j) = \Psi A(i_2, j); \\ 0 & \text{if } \Psi A(i_1, j) \neq \Psi A(i_2, j). \end{cases}$$

Now since every $A \in \mathscr{IR}(v, k, \lambda)$ must by Lemma 2.4.7 satisfy

$$\sum_{s=0}^{b-1} A(i_1, s)A(i_2, s) = \sum_{j=0}^{r-1}\sum_{l=0}^{q-1} A(i_1, qj + l)A(i_2, qj + l) = \lambda,$$

we have $\Psi A(i_1, j) \neq \Psi A(i_2, j)$ for exactly $r - \lambda$ coordinates $j \in \mathbb{Z}_r$. Furthermore, because i_1, i_2 were arbitrary, ΨA is equidistant with minimum distance $r - \lambda$.

We still have to show that Ψ is bijective. Given ΨA it is possible to reconstruct A uniquely using (3.9), so Ψ is injective. Let $C \in \mathscr{LC}_q(r, v, r - \lambda)$. It is clear that we can transform C to an $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_{qr}}$ using (3.9) so that $\Psi A = C$ and (2.16) holds. To establish surjectivity we must verify that $A \in \mathscr{IR}(v, k, \lambda)$, which amounts to checking properties (i) and (iii) given in Lemma 2.4.7. Recall Corollary 3.2.4, which states that an ED$_m$-code is equidistant and every coordinate value occurs exactly $k = v/q$ times in every coordinate. Consequently, every column of A constructed from C using (3.9) must contain exactly k ones, which establishes (i). Furthermore, the equidistance of C implies that any two distinct codewords agree on exactly λ coordinates and differ in the remaining $r - \lambda$. Thus, the rows of A that correspond to any two codewords contain a 1 in precisely λ common columns, which establishes (iii). ■

Based on the previous proof it is evident that it is possible to present a similar bijection in the reverse direction if $\mathscr{LC}_q(r, v, r - \lambda)$ is nonempty. Consequently,

Corollary 3.5.4 *Let $v, k, \lambda, n, M, d, q \in \mathbb{N} \setminus \{0\}$ satisfy (3.8) and suppose that $k < v$ and $d < n$. Then, an $RB(v, k, \lambda)$ design exists if and only if there exists an ED_m-code with parameters $(n, M, d)_q$.*

To establish that the isomorphism equivalence classes on $\mathscr{R}(v, k, \lambda)$ and the isometry equivalence classes on $\mathscr{C}_q(r, v, r - \lambda)$ are in a bijective correspondence, it suffices to show that the orbits $S_v \times S_\Delta \setminus\!\!\setminus \mathscr{I}\mathscr{R}(v, k, \lambda)$ and $S_v \times (S_q \wr S_r) \setminus\!\!\setminus \mathscr{L}\mathscr{C}_q(r, v, r - \lambda)$ are in a bijective correspondence, because these in turn correspond bijectively to the aforementioned equivalence classes.

Since $\Psi : \mathscr{I}\mathscr{R}(v, k, \lambda) \to \mathscr{L}\mathscr{C}_q(r, v, r - \lambda)$ is a bijection between $\mathscr{I}\mathscr{R}(v, k, \lambda)$ and $\mathscr{L}\mathscr{C}_q(r, v, r-\lambda)$, it suffices to show that the diagram (3.11) is commutative. (In the diagram, Φ is the isomorphism from Lemma 2.5.5.)

$$\begin{tikzcd} S_v \times S_\Delta \times \mathscr{I}\mathscr{R}(v, k, \lambda) \arrow[r, "\mathrm{id}\times\Phi\times\Psi"] \arrow[d, "(2.13)"'] & S_v \times (S_q \wr S_r) \times \mathscr{L}\mathscr{C}_q(r, v, r - \lambda) \arrow[d, "(3.4)"'] \\ \mathscr{I}\mathscr{R}(v, k, \lambda) \arrow[r, "\Psi"] & \mathscr{L}\mathscr{C}_q(r, v, r - \lambda) \end{tikzcd} \tag{3.11}$$

If this is the case, then the group actions (2.13) and (3.4) are clearly equivalent up to a relabelling of the group elements and the elements of the set on which the group acts. Consequently, the orbits of the group actions must be in a bijective correspondence. The following theorem shows that (3.11) is commutative.

Theorem 3.5.5 *Let $\mathscr{I}\mathscr{R}(v, k, \lambda)$ be nonempty with $k < v$ and suppose that*

$$\Phi : S_\Delta \to S_q \wr S_r, \qquad \Psi : \mathscr{I}\mathscr{R}(v, k, \lambda) \to \mathscr{L}\mathscr{C}_q(r, v, r - \lambda)$$

are the mappings from Lemma 2.5.5 and Theorem 3.5.3, respectively. Then,

$$(\sigma, \Phi(\tau))\Psi A = \Psi(\sigma, \tau)A$$

for all $(\sigma, \tau) \in S_v \times S_\Delta$ and $A \in \mathscr{I}\mathscr{R}(v, k, \lambda)$.

Proof. Select any $(\sigma, \tau) \in S_v \times S_\Delta$ and $A \in \mathscr{I}\mathscr{R}(v, k, \lambda)$. Put $\Phi(\tau) = (\mu, \pi)$. By Corollary 2.5.6, we have $\tau^{-1}(qj + l) = q\pi^{-1}(j) + \mu_j^{-1}(l)$ for all $(j, l) \in \mathbb{Z}_r \times \mathbb{Z}_q$. Select $(i, j, l) \in \mathbb{Z}_v \times \mathbb{Z}_r \times \mathbb{Z}_q$. By (2.13) we have

$$(\sigma, \tau)A(i, qj + l) = 1 \quad \Leftrightarrow \quad A(\sigma^{-1}(i), q\pi^{-1}(j) + \mu_j^{-1}(l)) = 1,$$

which combined with (3.9) gives

$$\Psi(\sigma, \tau)A(i, j) = l \quad \Leftrightarrow \quad A(\sigma^{-1}(i), q\pi^{-1}(j) + \mu_j^{-1}(l)) = 1. \tag{3.12}$$

From (3.9) we obtain

$$\mu_j(\Psi A(\sigma^{-1}(i), \pi^{-1}(j))) = l \quad \Leftrightarrow \quad A(\sigma^{-1}(i), q\pi^{-1}(j) + \mu_j^{-1}(l)) = 1.$$

In other words, by (3.4)

$$(\sigma, (\mu, \pi))\Psi A(i, j) = l \quad \Leftrightarrow \quad A(\sigma^{-1}(i), q\pi^{-1}(j) + \mu_j^{-1}(l)) = 1. \tag{3.13}$$

Thus, if we combine (3.12) and (3.13), we obtain

$$\Psi(\sigma, \tau)A(i, j) = l \quad \Leftrightarrow \quad (\sigma, (\mu, \pi))\Psi A(i, j) = l,$$

which establishes the claim as i, j, l were arbitrary. ■

Corollary 3.5.6 *Let $v, k, \lambda, n, M, d, q \in \mathbb{N}\backslash\{0\}$ satisfy (3.8) and suppose that $k < v$ and $d < n$. Then, an $RB(v, k, \lambda)$ design exists if and only if there exists an ED$_m$-code with parameters $(n, M, d)_q$. Furthermore, the isomorphism equivalence classes on $\mathscr{R}(v, k, \lambda)$ and the isometry equivalence classes on $\mathscr{C}_q(n, M, d)$ are in a bijective correspondence.*

4 ISOMORPH-FREE EXHAUSTIVE GENERATION

Most problems of generation of combinatorial objects can be formulated in terms of a nonempty set of objects X and a group G acting on X. Two objects are then *isomorphic* if they are on the same orbit of G on X, and the isomorph-free exhaustive generation problem is to produce exactly one representative from each G-orbit on X. Such a set of representatives is called an *orbit transversal*. Obvious examples of families of combinatorial objects that fall within this generation framework are block designs and error-correcting codes discussed in the preceding chapters.

This chapter examines algorithm families developed for isomorph-free exhaustive generation in an abstract setting. In the next chapter we use these algorithm families to develop isomorph-free exhaustive generation algorithms for block designs and their resolutions.

An algorithm for isomorph-free exhaustive generation typically has two principal components. The first component is a method for generating objects from X so that at least one representative from each G-orbit is obtained, and the second component is devoted to eliminating objects isomorphic to those already generated from consideration.

This division into components suggests a straightforward algorithm that keeps a record of orbit representatives encountered so far and employs an exhaustive method for generation of candidate orbit representatives. Whenever a new candidate representative is obtained, it is tested for isomorphism against the stored orbit representatives. A representative that is not isomorphic to any of the stored representatives is added to the record and output, whereas isomorphic copies of stored representatives are simply discarded.

The straightforward algorithm is effective in many cases, however, it has two fundamental sources of inefficiency.

The first drawback of the straightforward algorithm is that it has to keep track of all the orbit representatives encountered thus far. If the number of orbits is large, then the available storage may run out, or the isomorphism testing of each new candidate against a large number of representatives may require too much time even if sufficient storage is available for the orbit representatives.

This drawback can be circumvented by using a *canonical placement map*, which transforms isomorphism testing to testing computed canonical representatives for equality.

Definition 4.0.1 A mapping $c : X \to X$ is a *canonical placement map* with respect to G if it satisfies the properties

(i) $c(gx) = c(x)$ for all $g \in G$ and $x \in X$; and

(ii) for all $x \in X$ there exists a $g \in G$ such that $c(x) = gx$.

The element $c(x)$ is called the *canonical representative* of the orbit Gx.

(For examples of canonical placement maps, see [3, 81, 82].) Isomorphism testing can be performed using a canonical placement map by computing the corresponding canonical representative for each candidate representative and testing it for equality against a collection of stored canonical representatives. Furthermore, if every canonical representative appears as a candidate representative exactly once, then it suffices to accept $x \in X$ as a new orbit representative whenever $c(x) = x$. Note that no representatives need to be stored with this latter approach.

The second drawback of the straightforward algorithm involves the set X, which may be many orders of magnitude larger than the number of G-orbits on X. A good example is given by the $B(31, 10, 3)$ designs, for which

$$|\mathscr{B}(31, 10, 3)| = 9819113523076271961547525324800000000,$$

yet $|S_{31} \backslash\!\backslash \mathscr{B}(31, 10, 3)| = 151$. In this case it is clear that an exhaustive search through all elements of $\mathscr{B}(31, 10, 3)$ with the straightforward algorithm is infeasible due to the huge number of elements to be searched. To make a classification of the nonisomorphic designs in $\mathscr{B}(31, 10, 3)$ possible, isomorphic copies of partial configurations must be detected and eliminated from consideration already during the generation phase, which requires dedicated algorithms.

McKay [84] (see also [12]) classifies modern algorithms for isomorph-free exhaustive generation into roughly three types, although noting that the boundary between different algorithm families is unclear. The first algorithm family, introduced independently by Read [103] and Faradžev [35], is well-established and consists of *orderly algorithms*. (The term *orderly*, introduced by Read, comes from the fact that objects are generated subject to some total order on X and on the set of subobjects.) An orderly algorithm is essentially a recursive version of the straightforward algorithm involving alternating steps of generation and rejection of isomorphic copies. The rejection of isomorphic copies during the step-by-step construction is based on the properties of the selected canonical representatives of the orbits to be generated. Examples of orderly algorithms in the literature are numerous, [11, 30, 32, 87] provides a small sample. The second algorithm family, described by McKay in [84], constructs orbit representatives step by step along a canonical construction path. The existence of such a path is guaranteed by an axiomatic model which defines a tree structure on the set of unlabelled partial configurations. Practical examples of McKay-type algorithms appear in [10, 106]; see the references in McKay's article [84] for further examples. The last algorithm family is based on the homomorphism principle for group actions [62, 63], and uses homomorphisms of group actions for step-by-step refinement of an orbit transversal. Examples of this approach are provided by [50, 63, 73].

In this chapter we examine in detail Read–Faradžev-type and McKay-type algorithms and briefly illustrate the homomorphism principle for group actions using construction of double coset representatives as an example.

4.1 READ–FARADŽEV-TYPE ALGORITHMS

We describe Read–Faradžev-type algorithms using the model and ideas from Faradžev's paper [35]. Read's paper [103] presents perhaps a more generic model, but lacks the group-theoretic treatment present in Faradžev's paper.

To describe combinatorial objects, Faradžev uses the successful Pólya–de Bruijn model from enumerative combinatorics [27, 28, 100], which is as follows. Let X and Y be finite nonempty sets, and denote by Y^X the set of functions of X into Y. Suppose A and B are groups acting on X and Y, respectively, and let the direct product group $G = A \times B$ act on Y^X by the rule $((a, b), \varphi) \mapsto (a, b)\varphi$, where $(a, b) \in G$, $\varphi \in Y^X$, and

$$(a, b)\varphi : x \mapsto b\,\varphi(a^{-1}x) \tag{4.1}$$

for all $x \in X$. The elements of Y^X are the *labelled objects* of the model, and the orbits $G \setminus\!\!\setminus Y^X$ are the *unlabelled objects*.

The collection of combinatorial objects to be generated is described within this model as a union P of orbits of G on Y^X. In other words, P describes an orbit-invariant property shared by the objects to be constructed. (For example, $P = \mathscr{IB}(v, k, \lambda)$ with $X = \mathbb{Z}_v$, $Y = \mathbb{Z}_2^{\mathbb{Z}_b}$, and $G = S_v \times S_b$; cf. Section 5.1.) The aim of a Read–Faradžev-type algorithm is to generate exactly one representative from each orbit in P. For this purpose, we select an (for the moment) arbitrary transversal C for the orbits $G \setminus\!\!\setminus Y^X$. Elements of C are called the *canonical representatives* of their orbits.

A Read–Faradžev-type algorithm performs an exhaustive search for all $\varphi \in P \cap C$ by extending the domain of a mapping $\varphi' : X' \to Y$, where $X' \subseteq X$, one element of $X \setminus X'$ at a time so that all extensions of φ' to an element of $P \cap C$ are considered. The following definitions formalize the extension process. (We remark that our treatment is somewhat modified from [35].) Fix a total order on X, label the $|X| = N$ elements of X subject to the total order as $x_1 < x_2 < \cdots < x_N$, and put $X_i = \{x_1, x_2, \ldots, x_i\}$ for all $i = 1, \ldots, N$. Additionally, put $X_0 = \emptyset$. Denote by Y^{X_*} the disjoint union $Y^{X_0} \cup \cdots \cup Y^{X_N}$. (The set Y^{X_0} is only a formal entity; it consists of the "empty mapping," to which all other elements of Y^{X_*} are extensions in the following sense:) A mapping $\varphi' \in Y^{X_j}$ is an *extension* of a mapping $\varphi \in Y^{X_i}$ if $i \le j$ and $\varphi(x) = \varphi'(x)$ for all $x \in X_i$; we write $\varphi \preceq \varphi'$ to indicate this. An extension φ' of φ is *proper* if $i < j$; this is indicated by $\varphi \prec \varphi'$. Finally, a proper extension is *minimal* if $j = i + 1$.

Lemma 4.1.1 *The binary relation $\preceq$ is a partial order on Y^{X_*} with a unique minimal element. The maximal elements in Y^{X_*} constitute the labelled objects Y^X.*

Proof. The claimed properties follow immediately from the definitions and from the properties of the standard total order $\le$ on $\mathbb{Z}$. ∎

This partial order gives Y^{X_*} the structure of a search tree. The children of a tree node are its minimal proper extensions, the maximal elements are the leaves of the tree, and the unique minimal element is the root of the tree.

So far we have gained nothing over the straightforward algorithm presented earlier. However, we now have formalized the construction process for the objects in $P \cap C$ to a sufficient degree to enable pruning of isomorphic copies *during* the construction process. (Recall that the straightforward algorithm performs isomorph rejection only *after* an object has been completely constructed.) The condition as to when a search tree node may be pruned admits a straightforward formalization: Let $E \subseteq Y^X$. A subset $E^* \subseteq Y^{X_*}$ is called an *extract* of E if, for all $\varphi' \in E$, we have $\varphi \in E^*$ whenever $\varphi \preceq \varphi'$. In other words, if $\varphi \notin E^*$, then no extension of φ is in E. Consequently, an extract of $P \cap C$ describes a necessary condition for the extendibility of a mapping $\varphi \in Y^{X_*}$ to an element of $P \cap C$ and hence can be used to prune the search tree.

Pseudocode for a Read–Faradžev-type orbit transversal algorithm is given as Algorithm 1. (In the algorithm, we denote by $(P \cap C)^*$ an arbitrary extract of $P \cap C$.) The algorithm performs a depth-first traversal of the search tree defined by $\preceq$ as can be easily seen.

Algorithm 1 A Read–Faradžev-type orbit transversal algorithm.

```
procedure ORDERLY-SCAN($\varphi$ : a mapping in $Y^{X_*}$, $i$ : integer)
   if $\varphi \notin (P \cap C)^*$ then
      return
   end if
   if $i = N$ then
      if $\varphi \in P \cap C$ then
         output $\varphi$
      end if
      return
   end if
   construct all minimal proper extensions $\mathscr{E}[\varphi]$ of $\varphi$
   for all $\varphi' \in \mathscr{E}[\varphi]$ do
      ORDERLY-SCAN($\varphi'$, $i + 1$)
   end for
endprocedure
```

The performance of a Read–Faradžev-type algorithm depends heavily on the choice of canonical representatives C and the extracts used. A widely-used and successful (see [30, 32] and the references therein) family of canonical representatives, described in [35], is defined as follows. Recall that X is ordered. Introduce a total order on Y to obtain a lexicographic order on sets Y^{X_i} for all $i = 1, \dots, N$. Because $Y^X = Y^{X_N}$ is finite, we may define the unique lexicographic maximum element of every orbit of G on Y^X to be the canonical representative:

$$C = \{\varphi \in Y^X \,:\, \varphi \geq g\varphi \text{ for all } g \in G\}. \tag{4.2}$$

The success of this set of canonical representatives is based on its extracts, which can be used to prune the search after every extension step in Algorithm 1.

In what follows we describe two generic extracts for (4.2). For notational convenience we write G_{X_i} for the subgroup $A_{X_i} \times B \leq G$, where A_{X_i} is the setwise stabilizer of X_i in A. Additionally, we denote the restriction $\varphi|_{X_i}$ of a $\varphi \in Y^{X_j}$, where $i < j$, simply by φ_i.

Lemma 4.1.2 *Let $\varphi \in C$ and suppose $1 \leq i \leq N$. Then, $\varphi_i \geq g\varphi_i$ for all $g \in G_{X_i}$.*

Proof. Select $1 \le i \le N$. First we note that the action (4.1) of G_{X_i} is well-defined on Y^{X_i} since $A_{X_i} \le A$ is the setwise stabilizer of X_i on A. To reach a contradiction, suppose that there exists a $g \in G_{X_i}$ such that $g\varphi_i > \varphi_i$. By definition of lexicographic order and our choice of labelling $x_1 < \cdots < x_N$, there exists a $1 \le k \le i$ such that $g\varphi_i(x_k) > \varphi_i(x_k)$ and $g\varphi_i(x_j) = \varphi_i(x_j)$ for all $1 \le j < k$. Now, since $x_l > x_i$ for all $l > i$, we must have $g\varphi > \varphi$, which is a contradiction to the assumption $\varphi \in C$. ■

Corollary 4.1.3 *The set*

$$C^* = \bigcup_{i=1}^{N} \{\varphi \in Y^{X_i} \,:\, \varphi \ge g\varphi \text{ for all } g \in G_{X_i}\} \tag{4.3}$$

is an extract of C.

In applications extract (4.3) can be expensive to compute. The following lemma yields an extract which is often easier to compute, but which is inferior to (4.3) in pruning efficiency. We write G_{φ_i} for the stabilizer of φ_i in G_{X_i}; the stabilizer of x_{i+1} in G is denoted by $G_{x_{i+1}}$ as usual.

Lemma 4.1.4 *Let* $\varphi \in C$ *and suppose* $1 \le i < N$. *Then,* $\varphi_{i+1} \ge g\varphi_{i+1}$ *for all* $g \in G_{\varphi_i} \cap G_{x_{i+1}}$.

Proof. We note that the action (4.1) of $G_{\varphi_i} \cap G_{x_{i+1}}$ is well-defined on the set of minimal proper extensions of φ_i. Additionally, $G_{\varphi_i} \cap G_{x_{i+1}} \le G_{X_{i+1}}$. Thus, a $g \in G_{\varphi_i} \cap G_{x_{i+1}}$ with $g\varphi_{i+1} > \varphi_{i+1}$ is a contradiction to Lemma 4.1.2. ■

Corollary 4.1.5 *The set*

$$C^* = \bigcup_{i=1}^{N-1} \{\varphi \in Y^{X_{i+1}} \,:\, \varphi \ge g\varphi \text{ for all } g \in G_{\varphi_i} \cap G_{x_{i+1}}\} \tag{4.4}$$

is an extract of C.

4.2 MCKAY-TYPE ALGORITHMS

In this section we essentially follow the treatment in McKay's article [84]; the proofs of the theorems and lemmata have been slightly expanded, axiom (C3*) is new, and Lemma 4.2.4 is new.

The abstract model for McKay-type algorithms is as follows. Let G be a group that acts on a nonempty set X. The elements of X are the *labelled objects* of the model, and the orbits $G \backslash\!\backslash X$ are the *unlabelled objects*. For notational convenience we write $\mathscr{U}$ for the set of all unlabelled objects.

Associated with each labelled object $x \in X$ is a finite set $L(x)$ of *lower objects* and a finite set $U(x)$ of *upper objects*. The sets $\{x_1\}$,$\{x_2\}$,$L(x_1)$,$U(x_1)$, $L(x_2)$,$U(x_2)$ are constrained to be pairwise disjoint for all distinct $x_1, x_2 \in X$. For notational convenience we write $\check{X} = \cup_{x\in X} L(x)$ and $\hat{X} = \cup_{x\in X} U(x)$ for the sets of all lower and upper objects, respectively. The lower and upper objects are connected by means of a binary relation $R \subseteq \check{X} \times \hat{X}$ which is, for convenience, accessed using functions $u : \check{X} \to \mathscr{P}[\hat{X}]$ and $l : \hat{X} \to \mathscr{P}[\check{X}]$ defined by

$$u : \check{y} \mapsto \{\hat{x} \in \hat{X} \ : \ (\check{y}, \hat{x}) \in R\}, \qquad l : \hat{x} \mapsto \{\check{y} \in \check{X} \ : \ (\check{y}, \hat{x}) \in R\}.$$

The group G is assumed to act on $X \cup \check{X} \cup \hat{X}$ so that conditions (C1)-(C5) are satisfied. (Condition (C3*) is new and necessary for validity of Theorem 4.2.5; see Example 4.2.6 for a counterexample without (C3*).)

(C1) G fixes each of $X, \check{X}$ and $\hat{X}$ setwise.

(C2) For each $x \in X$ and $g \in G$ we have $L(gx) = gL(x)$ and $U(gx) = gU(x)$

(C3) For each $\check{y} \in \check{X}$, $u(\check{y}) \neq \emptyset$.

(C3*) For each $\hat{x} \in \hat{X}$ and $g \in G$, if $l(\hat{x}) \neq \emptyset$ then $l(g\hat{x}) \neq \emptyset$.

(C4) For any $\check{y} \in \check{X}$, $g \in G$, $\hat{x}_1 \in u(\check{y})$, and $\hat{x}_2 \in u(g\check{y})$, there exists a $h \in G$ such that $h\hat{x}_1 = \hat{x}_2$.

(C5) For any $\hat{x} \in \hat{X}$, $g \in G$, $\check{y}_1 \in l(\hat{x})$, and $\check{y}_2 \in l(g\hat{x})$, there exists a $h \in G$ such that $h\check{y}_1 = \check{y}_2$.

Furthermore, every $x \in X$ is assumed to have an *order* $o(x) \in \mathbb{N}$ that is shared by the elements of $L(x)$ and $U(x)$ so that conditions (O1) and (O2) given below are satisfied.

(O1) For each $x \in X$ and $g \in G$, we have $o(gx) = o(x)$.

(O2) For each $\check{x} \in \check{X}$ and $\hat{y} \in u(\check{x})$, we have $o(\hat{y}) < o(\check{x})$.

Finally, the existence of a function $m : X \to \mathscr{P}[\check{X}]$ satisfying the following conditions is assumed. (We write G_x for the stabilizer of $x \in X$ in G as usual.)

(M1) If $L(x) = \emptyset$, then $m(x) = \emptyset$.

(M2) If $L(x) \neq \emptyset$, then $m(x) = G_x\check{x}$ for some $\check{x} \in L(x)$.

(M3) For each $x \in X$ and $g \in G$, we have $m(gx) = gm(x)$.

This concludes the description of the McKay model.

The McKay model guarantees the existence of a tree structure on the set of unlabelled objects. We describe this structure next.

Definition 4.2.1 An unlabelled object $S \in \mathscr{U}$ is *irreducible* if $L(x) = \emptyset$ for each $x \in S$; otherwise an unlabelled object is *reducible*.

We write $\mathscr{U}_0$ for the set of all irreducible unlabelled objects, and $\mathscr{U}_1$ for the set of all reducible unlabelled objects. The following lemma gives a mapping that associates a parent unlabelled object to each reducible unlabelled object.

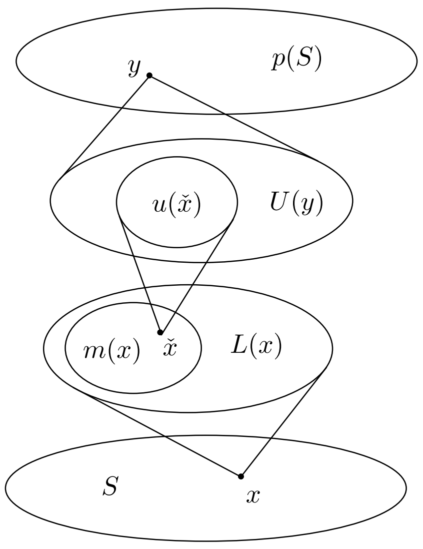


Figure 4.1: Illustration of Lemma 4.2.2.

Lemma 4.2.2 *There is a unique mapping* $p : \mathscr{U}_1 \to \mathscr{U}$ *with the following property: For all* $S \in \mathscr{U}_1$, $x \in S$, *and* $\check{x} \in m(x)$, *there exists a* $y \in p(S)$ *such that* $u(\check{x}) \subseteq U(y)$.

Proof. It suffices to show that, starting from a reducible unlabelled object $S \in \mathscr{U}_1$, an y will always exist and—no matter how the choices are made—all such y are on the same G-orbit. Let $S \in \mathscr{U}_1$ and choose any $x_1, x_2 \in S$. Since S is reducible, $L(x_1), L(x_2) \neq \emptyset$. Thus, by (M2) both $m(x_1)$ and $m(x_2)$ are nonempty. For $i = 1, 2$, choose any $\check{x}_i \in m(x_i)$ and $\hat{y}_i \in u(\check{x}_i)$. (The set $u(\check{x}_i)$ is nonempty by (C3).) Since $\hat{y}_i \in \hat{X}$, there exists a unique $y_i \in X$ such that $\hat{y}_i \in U(y_i)$; existence is clear by definition of $\hat{X}$, uniqueness is guaranteed by disjointness of the sets $U(y)$ for distinct y. Now, since $S = Gx_1 = Gx_2$, there exists a $g \in G$ such that $gx_1 = x_2$. Furthermore, (M2) and (M3) imply that there exists a $k \in G_{x_1}$ such that $gk\check{x}_1 = \check{x}_2$. Thus, by (C4) there exists an $h \in G$ such that $h\hat{y}_1 = \hat{y}_2$. By (C2), $h\hat{y}_1 \in hU(y_1) = U(hy_1)$. On the other hand, $\hat{y}_2 = h\hat{y}_1 \in U(y_2)$. Therefore, since the sets $U(y)$ are disjoint for distinct y, we must have $hy_1 = y_2$. ■

Definition 4.2.3 The unlabelled object $p(S)$ is the *parent* of $S \in \mathscr{U}_1$, and unlabelled objects in the set $\{S, p(S), p(p(S)), \dots\}$ are the *ancestors* of S. Conversely, an $S \in \mathscr{U}$ is a *child* of a $T \in \mathscr{U}$ if T is the parent of S, and S is a *descendant* of T if T is an ancestor of S.

By (O1) we can associate order $o(S)$ to an unlabelled object $S \in \mathscr{U}$ by setting $o(S) = o(x)$ for any $x \in S$.

Lemma 4.2.4 *Each unlabelled object* $S \in \mathscr{U}$ *has a finite number of ancestors and a finite number of children.*

Proof. Lemma 4.2.2 together with (O2) shows that $o(S) > o(p(S))$ for all $S \in \mathscr{U}_1$, which implies that p can be applied to an $S \in \mathscr{U}_1$ only a finite number of times before reaching an irreducible object. Let $T \in \mathscr{U}$ and fix a $y_0 \in T$. Suppose $S_1, S_2 \in \mathscr{U}_1$ are distinct children of T, that is, $p(S_1) =$

$p(S_2) = T$ and $S_1 \neq S_2$. For $i = 1, 2$, select $x_i \in S_i$, $\check{x}_i \in m(x_i)$, and let $y_i \in T$ such that $u(\check{x}_i) \subseteq U(y_i)$. Since $y_0, y_1, y_2 \in T$, we have $y_0 = h_1 y_1 = h_2 y_2$ for some $h_1, h_2 \in G$. Thus, $h_i u(\check{x}_i) \subseteq U(y_0)$ for $i = 1, 2$ by (C2). Suppose, for the sake of contradiction, that $\check{z} \in h_1 u(\check{x}_1) \cap h_2 u(\check{x}_2)$. Then, for $i = 1, 2$, we have $h_i^{-1}\check{z} \in u(\check{x}_i)$, that is, $\check{x}_i \in l(h_i^{-1}\check{z})$, which by (C5) implies $k\check{x}_1 = \check{x}_2$ for some $k \in G$. Now, by (M3), $m(kx_1) = km(x_1) \ni k\check{x}_1 = \check{x}_2 \in m(x_2)$. This together with (M2), (C2), and the disjointness of the sets $L(x)$ for distinct $x \in X$ implies $kx_1 = x_2$, which is a contradiction to the assumption $S_1 \neq S_2$. Thus, $h_1 u(\check{x}_1)$ and $h_2 u(\check{x}_2)$ are disjoint and, because $U(y_0)$ is finite, T can have only a finite number of children. ■

Consequently, p gives $\mathscr{U}$ the structure of a finitely branching forest of rooted trees where the roots of the trees are the irreducible unlabelled objects. A transversal algorithm for the orbits $\mathscr{U}$ is given as Algorithm 2.

Algorithm 2 A McKay-type orbit transversal algorithm.

```
procedure MCKAY-SCAN($x$ : labelled object, $n$ : integer)
  if $o(x) \leq n$ then
    output $x$
    for each orbit $G_x\hat{x}$ of $G_x$ on $U(x)$ do
      if $l(\hat{x}) \neq \emptyset$ then
        select any $\check{y} \in l(\hat{x})$, and suppose $\check{y} \in L(y)$
        if $\check{y} \in m(y)$ then
          MCKAY-SCAN($y, n$)
        end if
      end if
    end for
  end if
endprocedure
```

Theorem 4.2.5 *Let $S_0 \in \mathscr{U}$, and select any $x_0 \in S_0$. Then, the call* MCKAY-SCAN(x_0, n) *will output exactly one labelled object $x \in S$ for each descendant S of S_0 with order at most n.*

Proof. We say that an unlabelled object $S \in \mathscr{U}$ belongs to *generation i* if S has precisely $i + 1$ ancestors. Let S_0 belong to generation i_0.

Suppose, as an induction hypothesis, that the algorithm outputs exactly one labelled object $x \in S$ for each descendant S of S_0 belonging to generation i and with order at most n. The induction base case $i = i_0$ is clear – the algorithm outputs $x_0 \in S_0$ if $o(x_0) \leq n$. For the inductive step, consider a descendant S of S_0 belonging to generation $i + 1$ with order at most n.

We first establish that at least one representative of S is output. Since $p(S)$ belongs to generation i and (O2) implies $o(p(S)) < o(S) \leq n$, there exists an $x_1 \in p(S)$ such that x_1 is output by the induction hypothesis. By Lemma 4.2.2 there exists a $y_2 \in S$, a $\check{y}_2 \in m(y_2)$ and an $\hat{x}_2 \in u(\check{y}_2)$ such that $\hat{x}_2 \in U(x_2)$ for some $x_2 \in p(S)$. Since $x_1, x_2 \in p(S)$, we have $x_2 = gx_1$ for some $g \in G$. Thus, by (C2) $\hat{x}_2 \in gU(x_1)$. Let $\hat{x}_1 \in U(x_1)$ be the orbit representative chosen by the algorithm for which $\hat{x}_2 \in gG_{x_1}\hat{x}_1$, and let $h \in G_{x_1}$ such that $\hat{x}_2 = gh\hat{x}_1$. Since $\check{y}_2 \in l(\hat{x}_2)$, axiom (C3*) implies $l(\hat{x}_1) \neq \emptyset$.

Suppose $\check{y}_1 \in l(\hat{x}_1)$ is selected by the algorithm and that $\check{y}_1 \in L(y_1)$. Then, by (C5), there exists a $k \in G$ such that $k\check{y}_1 = \check{y}_2$, which implies $ky_1 = y_2$ by (C2). This together with $\check{y}_2 \in m(y_2)$ and (M3) implies $\check{y}_1 \in m(y_1)$. Therefore, the call MCKAY-SCAN(y_1, n) is made for $y_1 \in S$, and, because $o(y_1) = o(S) \leq n$, y_1 is output.

Now, suppose, for the sake of contradiction, that $y_1, y_2 \in S$ are both output. This implies that there were at least two calls to MCKAY-SCAN: one with arguments (y_1, n) and another with (y_2, n). For these calls to occur, there must exist a $\check{y}_1 \in m(y_1)$ and a $\check{y}_2 \in m(y_2)$ for which there exist $\hat{x}_1 \in u(\check{y}_1)$ and $\hat{x}_2 \in u(\check{y}_2)$. Furthermore, for $i = 1, 2$, $\hat{x}_i$ must occur as the representative of an orbit of G_{x_i} on $U(x_i)$ for some labelled object $x_i \in T_i$. By Lemma 4.2.2, we must have $T_1 = T_2 = p(S)$. Thus, by the induction hypothesis, $x_1 = x_2$. Since y_1 and y_2 are isomorphic, we have $g\check{y}_1 = \check{y}_2$ for some $g \in G$ by (M2) and (M3). Hence, by (C4), there exists a $h \in G$ such that $h\hat{x}_1 = \hat{x}_2$. By (C2) we have $U(hx_1) \ni h\hat{x}_1 = \hat{x}_2 \in U(x_2)$, and hence $hx_1 = x_2 = x_1$. Consequently, $h \in G_{x_1}$, and $\hat{x}_1$ and $\hat{x}_2$ are on the same orbit of G_{x_1} on $U(x_1)$. Since exactly one representative per orbit is selected by the algorithm, we must have $\hat{x}_1 = \hat{x}_2$, which is a contradiction as the algorithm makes at most one call to MCKAY-SCAN per orbit representative. ■

Example 4.2.6 We demonstrate that axiom (C3*) added to the original McKay model is necessary for the validity of Theorem 4.2.5. Figure 4.2 illustrates an instance of the McKay model that satisfies the axioms except for (C3*). The instance is as follows. Put $X = \{a_1, a_2, d_1, d_2\}$ and suppose that

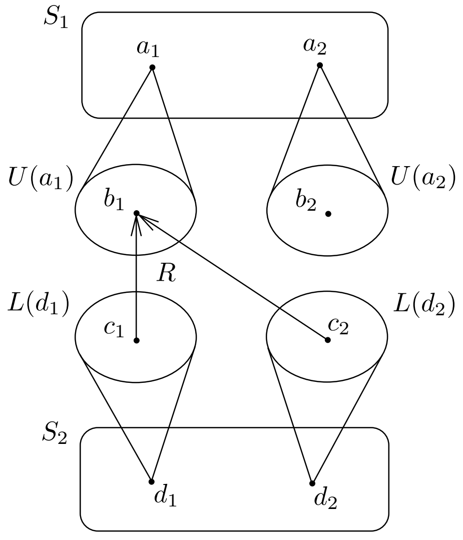


Figure 4.2: An instance of the McKay model without (C3*).

the sets of lower and upper objects are

$$\begin{aligned} &L(a_1) = L(a_2) = \emptyset, \quad && L(d_1) = \{c_1\}, \quad && L(d_2) = \{c_2\}, \\ &U(d_1) = U(d_2) = \emptyset, && U(a_1) = \{b_1\}, && U(a_2) = \{b_2\}, \end{aligned}$$

where b_1, b_2, c_1, c_2 are distinct from each other and from a_1, a_2, d_1, d_2.

Let $G = \{1, g\}$ act on $X \cup \check{X} \cup \hat{X} = \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$ by

$$\begin{array}{llll} ga_1 = a_2, & ga_2 = a_1, & gb_1 = b_2, & gb_2 = b_1, \\ gc_1 = c_2, & gc_2 = c_1, & gd_1 = d_2, & gd_2 = d_1. \end{array}$$

Finally, define $R = \{(c_1, b_1), (c_2, b_1)\}$,

$$o(a_1) = o(a_2) = o(b_1) = o(b_2) = 0, \qquad o(c_1) = o(c_2) = o(d_1) = o(d_2) = 1,$$

and

$$m(a_1) = m(a_2) = \emptyset, \qquad m(d_1) = \{c_1\}, \qquad m(d_2) = \{c_2\}.$$

It is straightforward to verify that conditions (C1)-(C5), (O1)-(O2), and (M1)-(M3) are satisfied, however, (C3*) does not hold because $l(b_1) \neq \emptyset$ and $l(gb_1) = \emptyset$.

The instance has two unlabelled objects, namely $S_1 = \{a_1, a_2\}$ and $S_2 = \{d_1, d_2\}$, where S_2 is a descendant of S_1 by Definition 4.2.3 and Lemma 4.2.2. Consequently, by Theorem 4.2.5 Algorithm 2 should output a representative for S_2 when started with either MCKAY-SCAN$(a_1, 1)$ or MCKAY-SCAN$(a_2, 1)$. On input $x = a_1$, $n = 1$ the algorithm behaves correctly and outputs a representative for S_2 regardless whether c_1 or c_2 is chosen as $\check{y}$ during the course of the algorithm. However, on input $x = a_2$, $n = 1$ we have $l(\check{x}) = l(b_2) = \emptyset$ and no representative is output. Thus, Theorem 4.2.5 requires axiom (C3*) for its validity. $\diamondsuit$

If determining orbit representatives of G_x on $U(x)$ is computationally demanding, an alternative transversal algorithm can be used. This algorithm employs explicit isomorphism testing instead of orbit computations, and it is given as Algorithm 3.

Algorithm 3 A McKay-type algorithm with explicit isomorphism testing.

```
procedure MCKAY-SCAN2($x$ : labelled object, $n$ : integer)
  if $o(x) \leq n$ then
    output $x$
    $T \leftarrow \emptyset$
    for each $\hat{x} \in U(x)$ do
      if $l(\hat{x}) \neq \emptyset$ then
        select any $\check{y} \in l(\hat{x})$, and suppose $\check{y} \in L(y)$
        if $\check{y} \in m(y)$ and $y$ is not isomorphic to any $y' \in T$ then
          MCKAY-SCAN2$(y, n)$
          $T \leftarrow T \cup \{y\}$
        end if
      end if
    end for
  end if
endprocedure
```

Theorem 4.2.7 *Let $S_0 \in \mathscr{U}$, and select any $x_0 \in S_0$. Then the call* MCKAY-SCAN2(x_0, n) *will output exactly one labelled object $x \in S$ for each descendant S of S_0 with order at most n.*

Proof. Since at least one representative per orbit of G_x on $U(x)$ is considered, it is clear by Theorem 4.2.5 that the algorithm outputs a representative for each descendant S of S_0 with order at most n. On the other hand, the explicit isomorphism testing performed by the algorithm suffices to delete all isomorphic labelled objects from output. The proof of this is similar to that of Theorem 4.2.5: First conclude inductively that isomorphic $y_1, y_2 \in S$ must originate from a unique call MCKAY-SCAN2(x, n) with $x \in p(S)$. But then either y_1 or y_2 must be removed by isomorphism testing. ■

4.3 THE HOMOMORPHISM PRINCIPLE

The homomorphism principle for group actions [62, 63] gives a different approach to the problem of constructing G-orbit representatives on X. The idea is to use a chain of homomorphisms of group actions to proceed step by step from a transversal of orbits of a "simpler" group action to the desired orbit transversal. We give an overview of the method and illustrate it using double cosets as an example.

The crucial ingredient of the method is the concept of a homomorphism of group actions:

Definition 4.3.1 Let G and H be groups and suppose G and H act on nonempty sets X and Y, respectively. A pair of mappings (η, θ), where $\theta : X \to Y$ and $\eta : G \to H$ is a group homomorphism, is a *homomorphism of group actions* if $\theta(gx) = \eta(g)\theta(x)$ for all $g \in G$ and $x \in X$.

The following theorem connects the orbits $H \setminus\!\!\setminus Y$ to the orbits $G \setminus\!\!\setminus X$ via a homomorphism of group actions, and is consequently called the *homomorphism principle*.

Theorem 4.3.2 *Let groups G and H act on nonempty sets X and Y, respectively, and suppose (η, θ) is a homomorphism connecting the two actions so that both η and θ are surjective. Then,*

(i) Suppose T is an arbitrary transversal of $H \setminus\!\!\setminus Y$. Then, for each orbit Gx in $G \setminus\!\!\setminus X$, there exists a unique $y \in T$ such that $Gx \cap \theta^{-1}(y) \neq \emptyset$.

(ii) For any $y \in Y$ the orbits in $G \setminus\!\!\setminus X$ that intersect $\theta^{-1}(y)$ are in a bijective correspondence to the orbits $\eta^{-1}(H_y) \setminus\!\!\setminus \theta^{-1}(y)$.

Proof. Select an orbit $Gx \in G \setminus\!\!\setminus X$ and put $\theta(x) = y_0$. Because T is a transversal of $H \setminus\!\!\setminus Y$, there exists a $y \in T$ and a $h \in H$ such that $y = hy_0$. Since η is surjective, there exists a $g \in G$ such that $\eta(g) = h$. Now, $\theta(gx) = \eta(g)\theta(x) = hy_0 = y$, which implies $Gx \cap \theta^{-1}(y) \neq \emptyset$. Suppose $Gx \cap \theta^{-1}(y') \neq \emptyset$ for some $y' \in T$. Then there exists a $g'x \in Gx$ such that $y' = \theta(g'x) = \eta(g')\theta(x) = \eta(g')y_0$. So, $y' = \eta(g')h^{-1}y$. Since both $y, y' \in T$, we must have $y = y'$ as T contains exactly one representative from each orbit. For (ii), select $y \in Y$. It is straightforward to verify that $\eta^{-1}(H_y) \leq G$. Select a $g \in \eta^{-1}(H_y)$ and suppose $x \in \theta^{-1}(y)$. Then $\eta(g) \in H_y$ and we have $\theta(gx) = \eta(g)\theta(x) = \eta(g)y = y$. As a result, $gx \in \theta^{-1}(y)$ and the action of $\eta^{-1}(H_y)$ on $\theta^{-1}(y)$ is well-defined. Now select any $x, x' \in \theta^{-1}(y)$

and $g \in G$. If $x' = gx$, then necessarily $y = \theta(x') = \theta(gx) = \eta(g)y$, so $g \in \eta^{-1}(H_y)$. Thus x, x' reside in the same G-orbit if and only if they reside in the same $\eta^{-1}(H_y)$-orbit. ■

We give the following two observations to illustrate the usefulness of the situation of Theorem 4.3.2 from a constructive point of view. (We note, however, that practical algorithms utilizing the situation of Theorem 4.3.2 are more involved, see [63, 72, 73, 107, 108] for examples.) First, given a transversal T of $H \setminus\setminus Y$ and the associated stabilizers H_y for each $y \in T$, we can construct a transversal T' of $G \setminus\setminus X$ by taking a union of transversals of $\eta^{-1}(H_y) \setminus\setminus \theta^{-1}(y)$ over all $y \in T$. Second, it is also possible to proceed from a transversal T' of $G \setminus\setminus X$ to a transversal of $H \setminus\setminus Y$ because $\theta(Gx) = \eta(G)\theta(x) = H\theta(x)$ for all $x \in X$. So, it suffices to consider only images $\theta(x)$ of $x \in T'$ and to eliminate all but one of those images that reside on each H-orbit.

A concrete example of the homomorphism principle is given by the construction of *double coset* representatives.

Definition 4.3.3 For $U, V \leq G$ and $g \in G$ the set $UgV = \{ugv : u \in U, v \in V\}$ is called the (U, V)-*double coset* of g.

Double cosets can alternatively defined as orbits of a group action. Let G be a group and suppose $U, V \leq G$. The mapping $(u, gV) \mapsto ugV$ of $U \times G/V$ into G/V is clearly a group action of U on the set of left cosets of V in G. Because two left cosets are either identical or disjoint (Theorem B.2.17), we may identify the orbits $U \setminus\setminus G/V$ with the set $U\backslash G/V = \{UgV : g \in G\}$.

A transversal for the (U, V)-cosets in G can now be constructed using a homomorphism of group actions and Theorem 4.3.2 as follows: Suppose that $V \leq \tilde{V} \leq G$ and let U act on $G/\tilde{V}$ similarly by left multiplication. Put $\eta = \mathrm{id}$ and define $\theta : G/V \to G/\tilde{V}$ by $gV \mapsto g\tilde{V}$ for all $g \in G$. We observe that θ is well-defined because

$$g_1V = g_2V \quad \Leftrightarrow \quad g_2^{-1}g_1 \in V \leq \tilde{V} \quad \Rightarrow \quad g_1\tilde{V} = g_2\tilde{V}$$

holds for all $g_1, g_2 \in G$. The pair (η, θ) is a homomorphism of group actions because $\theta(ugV) = ug\tilde{V} = u\theta(gV) = \eta(u)\theta(gV)$ for all $u \in U$ and $g \in G$. Moreover, both mappings are surjective as can be easily seen. Thus, Theorem 4.3.2 applies and from a transversal of $(U, \tilde{V})$-cosets in G we can construct a transversal of (U, V)-cosets in G, or the other way around. This enables the construction of a transversal of (U, V)-cosets in G step by step along a *ladder* of subgroups between V and G. (Schmalz introduced this construction technique for a transversal of double cosets in [107] and called it "Leiterspiel", or the *ladder game*.)

Definition 4.3.4 A *subgroup ladder* between groups V and G, $V \leq G$, is a sequence $V_0, V_1, \ldots, V_n$ such that $V = V_0$, $V_n = G$, and, for all $i \in \mathbb{Z}_n$, either $V_i \leq V_{i+1}$ or $V_i \geq V_{i+1}$.

For an application of the ladder game and double cosets to the construction of t-designs (a generalization of block designs) with a prescribed automorphism group, see [5, 108].

Double cosets have a range of applications in determining a set of orbit representatives (see [62]). This is a consequence of the following theorem, which transforms the problem of constructing an orbit transversal for the orbits $U \bbslash X$ to the problem of determining transversal(s) of double cosets.

Theorem 4.3.5 *Let G act on X and suppose $U \leq G$. Then, for all $x \in X$, the mapping $Ugx \mapsto UgG_x$ of $U \bbslash Gx$ into $U \backslash G / G_x$ is a bijection.*

Proof. Let $x \in X$. Recall that the mapping $gx \mapsto gG_x$ is a bijection of Gx onto G/G_x (Theorem B.2.33). Put $\eta = \mathrm{id}$ and $\theta : gx \mapsto gG_x$, and let U act on Gx by $(u, gx) \mapsto ugx$ and on G/G_x by $(u, gG_x) \mapsto ugG_x$. Consequently, the pair (η, θ) is a homomorphism of group actions. Because both η and θ are bijective, Theorem 4.3.2 shows that the orbits $U \bbslash Gx$ and $U \bbslash G/G_x = U \backslash G / G_x$ are in a bijective correspondence under $\bar{\theta} : Ugx \mapsto UgG_x$. ■

5 ALGORITHMS FOR BLOCK DESIGNS

In this chapter we consider the problem of isomorph-free exhaustive generation of block designs and their resolutions. We start by giving a survey of existing research, and then proceed to develop three algorithms for isomorph-free exhaustive generation of block designs and resolutions based on the Read–Faradžev and McKay frameworks from the previous chapter.

The most common algorithmic discipline employed in exhaustive generation of block designs is *backtrack search* (see [48] and [69, Ch. 4]) combined with some method for removing isomorphic partial designs from consideration such as canonical placement, explicit isomorphism testing, or the Read–Faradžev and McKay frameworks. There are at least three main approaches in the literature to formulate an exhaustive backtrack search for block designs. The first is to construct designs block by block and remove isomorphic partial solutions after addition of a block. This approach was used by Mathon and Lomas [77]. The second approach, examples of which include the work of McKay and Radziszowski [85] and Denny and Gibbons [30], is to proceed from smaller designs to larger designs by using some extension method, such as Alltop's extension theorem [1], combined with isomorph rejection. The third, and the most common, approach is to construct a design point by point and to perform isomorph rejection after addition of a point [31, 41, 57, 85, 93, 99, 112].

We emphasize that these backtrack search methods are aimed at the generation of a complete set of isomorphism equivalence class representatives for a given parameter family (v, k, λ). If additional properties are assumed of the designs, then other successful exhaustive methods exist. For example, generation of designs with a prescribed group of automorphisms has been studied by Kramer and Mesner [67], Kreher and Radziszowski [68], Schmalz [108], Betten *et al.* [5], and Laue [73].

There are at least three main approaches for generating resolutions of block designs in the literature. The first is to construct all nonisomorphic resolutions, if any, for each isomorphism class representative during an exhaustive search that considers representatives for all $B(v, k, \lambda)$ designs. This approach has been successfully applied by Östergård [93, 96]. The second approach is to use a suitable combinatorial subconfiguration as a starting point for generation, see Lam and Tonchev [70] and Ito *et al.* [56] for examples. The third approach is to generate resolutions from scratch using backtrack search with isomorph rejection. There are again two variants to such a backtrack search. The first variant is to construct resolutions parallel class by parallel class. It was used by Dinitz *et al.* [32] to classify the nonisomorphic one-factorizations of K_{12}, the complete graph on 12 vertices. (This corresponds to a classification of the nonisomorphic resolutions in $\mathscr{R}(12, 2, 1)$.) Additional examples of this variant include the classification of $RB(12, 4, 3)$ designs conducted by Morales and Velarde [90]. The second variant is to proceed point by point, that is, recalling the correspondence from Section 3.5, codeword by codeword. Backtrack search with isomorph rejection has been used for construction of codes in at least [13, 95, 92].

5.1 A READ–FARADŽEV-TYPE ALGORITHM FOR BLOCK DESIGNS

The algorithm described in this section is a variant of a well-known exhaustive generation algorithm for block designs. Several researchers, including at least Gibbons [41], Ivanov [57], Pietsch [99], Spence [112], and Denny [30], have contributed to its development.

The algorithm constructs lexicographically maximum representatives of isomorphism equivalence classes of $B(v, k, \lambda)$ designs point by point using backtrack search combined with a maximum clique algorithm. This approach was motivated by the successful classification of the nonisomorphic $B(31, 10, 3)$ and $B(24, 4, 1)$ designs conducted by Spence [110, 112] using a similar algorithm. (So, in this sense our algorithm is not new.) Additional motivation for this approach of combining a comparably slow orderly approach early in the search with a fast-running clique algorithm later in the search can be found in a set of principles for designing fast backtrack algorithms presented in [80].

Our contribution to the algorithm is that we improve the efficiency of the lexicographic maximum test using a pruning technique applied by Meringer [87] (see also [86]) in orderly construction of regular graphs and cages. A second novelty is that we apply a recent maximum clique algorithm of Östergård [94] in the second stage of the algorithm.

A partial solution in the backtrack search is a $w \times b$ incidence system $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$, where $1 \leq w \leq v$. (Throughout this section we assume that v, b, r, k, λ are positive integers that satisfy the necessary conditions (2.4) for $B(v, k, \lambda)$ design existence.) For every $1 \leq w \leq v$, the set $\mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ is assumed to have lexicographic order induced by the standard lexicographic order on $\mathbb{Z}_w \times \mathbb{Z}_b$ and $\mathbb{Z}_2$ (see Appendix B for a precise definition of lexicographic order). A complete solution in the search is an incidence system $A \in \mathscr{I}\mathscr{B}(v, k, \lambda)$ that is the lexicographic maximum of its orbit under the row and column permuting action (2.13) of $S_w \times S_b$.

On input $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ the behaviour of the algorithm is divided into two stages depending on the number of points (that is, rows) in the incidence system. Let $1 \leq p \leq v$ be the number of rows that separates the two stages.

Stage 1. If $w < p$, then the algorithm performs a two-level backtrack search for extensions to the input incidence system. The first level constructs every possible row which extends the input incidence system. (The row generation procedure is described in Section 5.1.2.) Whenever a valid extending row is found on the first level, a second level of backtrack search is performed to verify that the extended incidence system is the lexicographic maximum of its orbit under row and column permutation. (This test is described in Section 5.1.3.) If the extended incidence system is the maximum of its orbit, then it is recursively considered for extensions and maximality.

Stage 2. If $w \geq p$, then the algorithm first generates a list of *all* rows that extend the input incidence system. These rows form the vertices of a graph whose $(v - p)$-cliques contain all extensions of the input incidence system to a lexicographically maximal block design, if any. (The graph generation and the clique search are described in Section 5.1.4.)

To argue the correctness of the algorithm we embed it into the Read–Faradžev framework. This embedding is then used as a basis for deriving pruning conditions to the backtrack search.

5.1.1 Embedding into the Read–Faradžev framework

The embedding to the Read–Faradžev framework rests on the following elementary identification: Any incidence system $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ can be thought as a mapping $\varphi : \mathbb{Z}_v \to \mathbb{Z}_2^{\mathbb{Z}_b}$ that takes $i \in \mathbb{Z}_v$ to a mapping $\varphi_i : \mathbb{Z}_b \to \mathbb{Z}_2$ that describes the entries of row i of A, that is, $\varphi_i : j \mapsto A(i,j)$ for all $(i,j) \in \mathbb{Z}_v \times \mathbb{Z}_b$.

This identification of $\mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ and $(\mathbb{Z}_2^{\mathbb{Z}_b})^{\mathbb{Z}_v}$ allows us to consider the point-by-point construction of incidence systems $A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ in the Read–Faradžev framework by setting $X = \mathbb{Z}_v$, $Y = \mathbb{Z}_2^{\mathbb{Z}_b}$, $G = S_v \times S_b$, and letting $(\sigma, \tau) \in G$ act on $\varphi \in Y^X$ by

$$(\sigma,\tau)\varphi : i \mapsto \tau\varphi_{\sigma^{-1}(i)}, \qquad \tau\varphi_{\sigma^{-1}(i)} : j \mapsto \varphi_{\sigma^{-1}(i)}(\tau^{-1}(j)) \tag{5.1}$$

for all $(i,j) \in \mathbb{Z}_v \times \mathbb{Z}_b$. It is straightforward to verify that the orbits $G \setminus\!\!\setminus Y^X$ of (5.1) correspond to the orbits $S_v \times S_b \setminus\!\!\setminus \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ of the row and column-permuting action (2.13). Moreover, lexicographic order on Y^X is equivalent to lexicographic order on $\mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ if $Y = \mathbb{Z}_2^{\mathbb{Z}_b}$ has the standard lexicographic order.

To generate block designs in the Read–Faradžev framework, we put $P = \mathscr{IB}(v,k,\lambda)$ and select the lexicographic maximum incidence system of each orbit in $S_v \times S_b \setminus\!\!\setminus \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b}$ as the canonical representative. Formally, we let

$$C = \{A \in \mathbb{Z}_2^{\mathbb{Z}_v \times \mathbb{Z}_b} : (\sigma,\tau)A \le A \text{ for all } (\sigma,\tau) \in G\}. \tag{5.2}$$

We observe that with these choices extending a partial solution in the Read–Faradžev framework corresponds to adding zero or more rows to an incidence system. Moreover, a proper extension corresponds to the addition of at least one row, and a minimal proper extension adds exactly one row.

The lexicographic maximum test performed after addition of each row in Stage 1 of the algorithm now corresponds to applying extract (4.3) to prune the search in the Read–Faradžev framework. (To see this, note that for $1 \le w \le v$ the action of the stabilizer G_{X_w} in (4.3) corresponds to the row and column permuting action of $S_w \times S_b$; cf. Lemma 5.1.2.) Thus, Stage 1 of the algorithm is essentially an implementation of Algorithm 1 for the generation of block designs.

We will now derive necessary conditions for extendibility of a partial solution based on the properties of the incidence systems in $P \cap C$. Each of the conditions corresponds to an extract in the Read–Faradžev framework. So, as long as only such necessary conditions are used to prune the search, the first stage of the algorithm correctly generates all partial solutions that have an extension in $P \cap C$.

Recall that the incidence systems in $P = \mathscr{IB}(v,k,\lambda)$ are characterized by properties (i)-(iii) of Lemma 2.4.7. These properties give us the following necessary condition for extendibility:

Lemma 5.1.1 *Let* $w \in \{1, 2, \ldots, v\}$. *An incidence system* $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ *has an extension in* P *only if* *(i)* $\sum_{i=0}^{w-1} A(i,j) \leq k$ *for all* $j \in \mathbb{Z}_b$*; and*

(ii) $\sum_{j=0}^{b-1} A(i,j) = r$ *for all* $i \in \mathbb{Z}_w$*; and*

(iii) $\sum_{j=0}^{b-1} A(i_1,j)A(i_2,j) = \lambda$ *for all* $i_1, i_2 \in \mathbb{Z}_w$, $i_1 \neq i_2$.

The next two necessary conditions for extendibility appear in [57]. The first condition is extract (4.3) adapted to the generation of lexicographically maximal incidence systems under the row and column permuting action.

Lemma 5.1.2 *Let* $w \in \{1, 2, \ldots, v\}$ *and suppose* $S_w \times S_b$ *acts on* $\mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ *by permutation of the rows and columns. Then, an incidence system* $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ *has an extension in* C *only if* $(\sigma, \tau)A \leq A$ *for all* $(\sigma, \tau) \in S_w \times S_b$.

This condition implies that the rows and columns of any incidence system that has an extension in C must appear in decreasing lexicographic order. We write $A(i, \cdot)$ for row i of $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ and $A(\cdot, j)$ for column j of A. Formally, we consider $A(i, \cdot) : \mathbb{Z}_b \to \mathbb{Z}_2$ and $A(\cdot, j) : \mathbb{Z}_w \to \mathbb{Z}_2$ as mappings defined by

$$A(i, \cdot) : j \mapsto A(i,j), \qquad A(\cdot, j) : i \mapsto A(i,j)$$

for all $(i,j) \in \mathbb{Z}_w \times \mathbb{Z}_b$ so that the rows and columns of A inherit the standard lexicographic order from $\mathbb{Z}_2^{\mathbb{Z}_b}$ and $\mathbb{Z}_2^{\mathbb{Z}_w}$, respectively.

Lemma 5.1.3 *Let* $w \in \{1, 2, \ldots, v\}$. *An incidence system* $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ *has an extension in* C *only if* $A(i, \cdot) \geq A(i+1, \cdot)$ *for all* $i \in \mathbb{Z}_{w-1}$ *and* $A(\cdot, j) \geq A(\cdot, j+1)$ *for all* $j \in \mathbb{Z}_{b-1}$.

Proof. If either the rows or the columns of A do not appear in decreasing lexicographic order, then we can sort them to decreasing order. Such a sorted incidence system is lexicographically greater than A and related to A by a permutation of the rows and columns. Consequently, the necessary condition of Lemma 5.1.2 is not satisfied and A cannot be extended to an element of C. ■

The next condition is a reformulation of an observation made by Denny and Mathon [31].

Lemma 5.1.4 *Let* $w \in \{1, 2, \ldots, v\}$, *let* $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$, *and suppose that there exists a* $j_0 \in \mathbb{Z}_b$ *satisfying* $\sum_{i=0}^{w-1} A(i, j_0) < k$. *Then,* A *has an extension in* $P \cap C$ *only if* $A(s, t_A(s)) = 1$ *for all* $s \in \{1, 2, \ldots, w\}$, *where*

$$t_A(s) = \min\{j \in \mathbb{Z}_b : \sum_{i=0}^{s-1} A(i,j) < k\}.$$

Proof. Select $s \in \{1, 2, \ldots, w\}$ and suppose that A has an extension in $P \cap C$. Then $A(s,j) = 0$ for all $j \in \mathbb{Z}_{t_A(s)}$, because otherwise condition (i) of Lemma 5.1.1 would be violated. To reach a contradiction, suppose that $A(s, t_A(s)) = 0$. Then, for every extension $A' \in P$ of A, there must exist a $r > s$ such that $A'(r, t_A(s)) = 1$ and $A'(r, j) = 0$ for every $j \in \mathbb{Z}_{t_A(s)}$. But then $A'(s, \cdot) < A'(r, \cdot)$ and A' has no extension in C (that is, $A' \notin C$) by Lemma 5.1.3. This is a contradiction since A' was arbitrary. ■

Although the following is not implemented in our algorithm, we wish to mention that Denny and Gibbons [30] have developed a pruning method (*strong partial isomorph rejection*) based on stored automorphisms of the input incidence system, which also enables backjumping in a backtrack search for row extensions; we note that this pruning technique is essentially an application of extract (4.4) to incidence systems.

5.1.2 Row generation

The row generation algorithm accepts as input an incidence system $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$, and produces all rows $x \in \mathbb{Z}_2^{\mathbb{Z}_b}$ so that A extended with x satisfies a suitable collection of necessary conditions for extendibility to an element of $P \cap C$. Our row generation algorithm uses the necessary conditions for extendability given in Lemmata 5.1.1, 5.1.3, and 5.1.4. (Naturally, the input A must also satisfy these conditions, because otherwise it clearly has no extension to an element of $P \cap C$.)

The problem of producing all rows $x \in \mathbb{Z}_2^{\mathbb{Z}_b}$ so that A extended with x satisfies the necessary condition of Lemma 5.1.1 is equivalent to determining all integer solutions $x \in \mathbb{Z}_2^{\mathbb{Z}_b}$ to the system of linear equations

$$\begin{pmatrix} 1 & \cdots & 1 \\ A(0,0) & \cdots & A(0,b-1) \\ \vdots & & \vdots \\ A(w-1,0) & \cdots & A(w-1,b-1) \end{pmatrix} \begin{pmatrix} x(0) \\ \vdots \\ x(b-1) \end{pmatrix} = \begin{pmatrix} r \\ \lambda \\ \vdots \\ \lambda \end{pmatrix}, \tag{5.3}$$

where $x(j) = 0$ whenever $\sum_{i=0}^{w-1} A(i,j) = k$. (So, we can remove the columns j satisfying $\sum_{i=0}^{w-1} A(i,j) = k$ from (5.3) and fix $x(j) = 0$. In what follows we implicitly assume this deletion has been performed.)

The remaining necessary conditions allow us to further constrain the set of admissible row extensions. The lexicographic order constraint (Lemma 5.1.3) is straightforward to incorporate into (5.3): The decreasing lexicographic row order is enforced by requiring that a solution $x \in \mathbb{Z}_2^{\mathbb{Z}_b}$ to (5.3) must be lexicographically lesser than the lexicographically least row in A. The decreasing lexicographic column order is enforced by requiring for every $j \in \{1, 2, \ldots, b-1\}$ that $x(j) \le x(j-1)$ whenever $A(\cdot, j) = A(\cdot, j-1)$. Finally, Lemma 5.1.4 fixes $x(t_A(w)) = 1$.

All solutions to (5.3) satisfying the additional constraints of Lemmata 5.1.3 and 5.1.4 can now be determined using, for example, backtrack search. Our implementation relies on backtrack search which first fixes $x(0)$, then $x(1)$, $x(2)$ and so on until either a solution is reached or it is no longer possible satisfy (5.3), at which point backtracking is performed. Inextendible partial solutions are detected by keeping track of the sum of all the $x(j)$s in the partial solution and the number of ones in each row in the remaining columns of A. A partial solution $x(0), x(1), \ldots, x(s)$ is pruned if either $b - s - 1 < r - \sum_{j=0}^{s} x(j)$ or $\sum_{j=s+1}^{b-1} A(i,j) < \lambda - \sum_{j=0}^{s} A(i,j)x(j)$ for some $i \in \mathbb{Z}_w$. (We point out that more sophisticated methods for detecting infeasible solutions early in this type of backtrack search are available, see [29, p. 119] for a packing constraint.)

Alternative methods to backtrack search for determining all solutions to systems of linear integer equations analogous to (5.3) appear in [108, 118].

5.1.3 Canonicity test

The purpose of the canonicity test is to verify that a given incidence system is the lexicographic maximum of its orbit under the row and column permuting action (2.13). For a given $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$, this is equivalent to deciding whether there exists a $(\sigma, \tau) \in S_w \times S_b$ such that $(\sigma, \tau)A > A$.

A straightforward implementation of the canonicity test simply searches through all possible choices $(\sigma, \tau) \in S_w \times S_b$ and tests whether $(\sigma, \tau)A > A$. However, we can avoid searching through S_b for every $\sigma \in S_w$ by noting that the lexicographic rank of $(\sigma, \tau)A$ is maximal for those $\tau \in S_b$ that sort the columns of $(\sigma, 1)A$ to decreasing lexicographic order. Thus, for canonicity testing it suffices to search through every $\sigma \in S_w$ and test whether the column-sorted version of $(\sigma, 1)A$ is lexicographically greater than A. Furthermore, a search through all $\sigma \in S_w$ can be implemented as a backtrack search with pruning so that in general only a fraction of the $w!$ row permutations need to be considered for a given incidence system.

The backtrack search proceeds step by step fixing one image point of a permutation $\sigma \in S_w$ at a time, that is, the image point $\sigma(0)$ is fixed first, then $\sigma(1)$, $\sigma(2)$, and so forth. Pruning is made possible by the observation that from a partial solution $\sigma(0), \dots, \sigma(i)$ we can determine the first $i+1$ rows of the matrix $(\sigma^{-1}, 1)A$, that is, we can determine the restriction $(\sigma^{-1}, 1)A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$. Consequently, we can sort the columns of $(\sigma^{-1}, 1)A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$ to decreasing lexicographic order and determine whether the matrix so obtained differs from $A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$. If $A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$ is lexicographically greater than the column-sorted matrix, then we can backtrack because no extension of the current partial solution will lead to a matrix greater than A. On the other hand, if $A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$ is lexicographically lesser than the column-sorted matrix, then every extension of the current partial solution will lead to a matrix which is greater than A (and, therefore, A is not the lexicographic maximum of its orbit). Thus, the only case in which we have to search further is when the column-sorted matrix is equal to $A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$.

The backtrack search algorithm just described for checking lexicographic maximality of incidence systems is well-known [30, 31, 93]. A problem with this algorithm is that searches through all elements of the *point automorphism group* $\mathrm{Aut}_P(A) = \{\sigma \in S_w : \exists \tau \in S_b \ (\sigma, \tau)A = A\}$, that is, the full automorphism group of the set system that corresponds to A; cf. Definition 2.1.13.

The following observation, attributed to R. Grund in [87], gives us a way to prune the search whenever a point automorphism is encountered.

Lemma 5.1.5 *Let G act on a nonempty totally ordered set X and let $U \leq G$. Furthermore, suppose that $x \in X$ satisfies $ux \geq x$ for all $u \in U$, and that we have $\hat{u}gx = x$ for some $\hat{u} \in U$ and $g \in G$. Then, $ugx \geq x$ for all $u \in U$.*

Proof. Select any $u \in U$. Then, $ugx = u\hat{u}^{-1}\hat{u}gx = u\hat{u}^{-1}x \geq x$ since $u\hat{u}^{-1} \in U$. ■

Corollary 5.1.6 *Let $S_w \times S_b$ act on $\mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ by permutation of the rows and columns, let $U \leq S_w$, and suppose that $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$ satisfies $(\sigma, \tau)A \leq A$ for all $(\sigma, \tau) \in U \times S_b$. Then, $(\sigma\pi, \tau)A \leq A$ for all $(\sigma, \tau) \in U \times S_b$ whenever $(\hat{\sigma}\pi, \hat{\tau})A = A$ for some $\pi \in S_w$, $\hat{\sigma} \in U$, and $\hat{\tau} \in S_b$.*

In other words, whenever the backtrack search has tested all $\sigma \in U$ for a subgroup $U \leq S_w$, and no counterexample to the maximality of A was found, then the discovery of a point automorphism $\pi \in \mathrm{Aut}_P(A)$ implies that we can ignore the entire right coset $U\pi$ in the search.

To enable as much pruning as possible with a minimal amount of prior testing, we can take advantage of a suitable chain of subgroups

$$S_w \geq U_0 \geq U_1 \geq U_2 \geq \cdots \geq U_{n-1} = \{1\}$$

and order the search so that U_{n-1} is tested first, then $U_{n-2} \setminus U_{n-1}$, $U_{n-3} \setminus U_{n-2}$ and so on until either a counterexample is found or all elements of S_w have been considered.

Meringer [87] uses the chain $U_i = \{\pi \in S_w : \pi(j) = j \text{ for all } j \in \mathbb{Z}_{i+1}\}$, $i \in \mathbb{Z}_w$, of point stabilizer subgroups of S_w in his canonicity testing algorithm for graphs. This has the advantage that a successor group in the chain can be expressed as a disjoint union of left cosets with only transpositions as the coset representatives, that is,

$$U_{i-1} = \bigcup_{j=i}^{w-1} (i\ j)U_i \tag{5.4}$$

for all $i \in \mathbb{Z}_w$ (we assume $U_{-1} = S_w$).

This stabilizer chain leads to a straightforward lexicographic maximality test implementation for incidence systems, which is formulated as Algorithm 4. Note that although Corollary 5.1.6 involves a *right* coset, the search in Algorithm 4 is performed using *left* cosets; this is due to the inversion of σ in (2.13).

Whenever A is the lexicographic maximum representative of its orbit, we obtain as a side effect of the canonicity test a *strong generating set* (see [14, 33]) for the group $\mathrm{Aut}_P(A)$ with respect to the chain $U_0 \geq U_1 \geq \cdots \geq U_{w-1}$ by storing the discovered automorphisms σ during the course of the algorithm.

5.1.4 Clique search

Before discussing the second stage of the algorithm, we review the necessary graph-theoretic definitions. Recall that a *graph* is a pair $G = (V, E)$, where V is a set of vertices and E is a set of 2-subsets of V, called edges (cf. Definition 2.1.5). The *order* of a graph is the number of vertices it contains. A graph is *complete* if E contains all 2-subsets of V. A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. A complete subgraph of a graph is called a *clique*. A clique of order k is called a k-clique.

Let $A \in \mathbb{Z}_2^{\mathbb{Z}_p \times \mathbb{Z}_b}$ be an incidence system which has been input by the first stage of the algorithm to the second stage. (So, A satisfies the necessary conditions for extendibility given in Lemmata 5.1.1 and 5.1.3.)

Algorithm 4 Canonicity test for a $w \times b$ incidence system $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$.

```
function COSET-SEARCH($\sigma$ : permutation, $i, l, w$ : integer) : integer
   if $i = w$ then
      (* Automorphism $\sigma \in \mathrm{Aut}_P(A)$ discovered. *)
      return $0$
   end if
   if $i < l$ then
      $J \leftarrow \{i\}$
   else
      $J \leftarrow \{i, i+1, \dots, w-1\}$
   end if
   if $i = l$ then
      (* Coset of the identity has been tested before, remove it. *)
      $J \leftarrow J \setminus \{i\}$
   end if
   for all $j \in J$ do
      $\sigma' \leftarrow \sigma \circ (i\ j)$
      put $A'(s,t) = A(\sigma'(s), t)$ for all $(s,t) \in \mathbb{Z}_{i+1} \times \mathbb{Z}_b$
      sort columns of $A'$ to decreasing lexicographic order
      if $A' \geq A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$ then
         if $A' > A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_b}$ then
            (* Counterexample found. *)
            return -1
         end if
         $d \leftarrow$ COSET-SEARCH$(\sigma', i+1, l, w)$
         if $d < 0 \vee (d = 0 \wedge i > l)$ then
            (* Counterexample found or Corollary 5.1.6 applies. *)
            return $d$
         end if
      end if
   end for
   return $1$
function CANONICAL($A$ : $w \times b$ incidence system) : boolean
   for $l = w, w-1, \dots, 0$ do
      (* Test left cosets of the stabilizer of $0, 1, \dots, l$ in $S_w$. *)
      if COSET-SEARCH$(\mathrm{id}, 0, l, w) < 0$ then
         return FALSE
      end if
   end for
   return TRUE
```

The problem of extending A to a block design in $P\cap C$ can be transformed to a problem of locating a collection $(v-p)$-cliques in the *compatibility graph* of A as follows.

Definition 5.1.7 The *compatibility graph* $\mathscr{G}_{r,\lambda}(A)$ of an incidence system $A \in \mathbb{Z}_2^{\mathbb{Z}_p\times\mathbb{Z}_b}$ is the graph whose vertices consist of all solutions $x \in \mathbb{Z}_2^{\mathbb{Z}_b}$ to (5.3) that are lexicographically lesser than any row of A; any two vertices x_1, x_2 are connected by an edge if and only if $\sum_{j=0}^{b-1} x_1(j)x_2(j) = \lambda$.

Suppose now that $A' \in \mathscr{I}\mathscr{B}(v,k,\lambda)$ is an extension of A whose rows are in decreasing lexicographic order. Then, by (i) and (iii) of Lemma 2.4.7, the last $v-p$ rows of A' form a set of $v-p$ vertices of $\mathscr{G}_{r,\lambda}(A)$. Furthermore, there is an edge between each pair of these vertices in $\mathscr{G}_{r,\lambda}(A)$. Thus, the vertices and edges form a $(v-p)$-clique in $\mathscr{G}_{r,\lambda}(A)$. Conversely, suppose that $x_1,\ldots,x_{v-p} \in \mathbb{Z}_2^{\mathbb{Z}_b}$ are the vertices of a $(v-p)$-clique in $\mathscr{G}_{r,\lambda}(A)$. Then, we can extend A using $x_1,\ldots,x_{v-p}$ and obtain an incidence system $A' \in \mathbb{Z}_2^{\mathbb{Z}_v\times\mathbb{Z}_b}$ that satisfies properties (i) and (iii) of Lemma 2.4.7. By Corollary 2.4.9, we have $A' \in \mathscr{I}\mathscr{B}(v,k,\lambda)$.

Consequently, every collection of rows that extends A to an element of $P\cap C$ appears as a $(v-p)$-clique in $\mathscr{G}_{r,\lambda}(A)$. However, although each $(v-p)$-clique in $\mathscr{G}_{r,\lambda}(A)$ extends A to an element of P, such extended A is not necessarily the lexicographic maximum of its orbit. Therefore, all extensions obtained from $(v-p)$-cliques of $\mathscr{G}_{r,\lambda}(A)$ must still be tested for maximality before they are accepted as elements of $P\cap C$ by the algorithm. As a side effect of the test, we conveniently obtain a strong generating set for the point automorphism group of each canonical representative.

The following lemma shows that $(v-p)$-cliques are maximum in $\mathscr{G}_{r,\lambda}(A)$.

Lemma 5.1.8 *If $A \in \mathbb{Z}_2^{\mathbb{Z}_p\times\mathbb{Z}_b}$ satisfies the necessary conditions for extendibility given in Lemma 5.1.1, then the order of any clique in $\mathscr{G}_{r,\lambda}(A)$ is at most $v-p$.*

Proof. Put $x_i = A(i,\cdot) \in \mathbb{Z}_2^{\mathbb{Z}_b}$ for all $i \in \mathbb{Z}_p$ and suppose $x_p,\ldots,x_{w-1}$ are the vertices of a $(w-p)$-clique in $\mathscr{G}_{r,\lambda}(A)$. By Lemma 5.1.1 and Definition 5.1.7, each x_i contains precisely r ones, and $\sum_{j=0}^{b-1} x_{i_1}(j)x_{i_2}(j) = \lambda$ for all distinct $i_1, i_2 \in \mathbb{Z}_w$. Consequently, $d_H(x_{i_1}, x_{i_2}) = 2(r-\lambda)$ for all distinct $i_1, i_2 \in \mathbb{Z}_w$. The b-tuples $x_1,\ldots,x_w$ form a constant weight code of length b, weight r, and minimum distance $2(r-\lambda)$. The cardinality of such a code is bounded from above by the so-called Johnson bound (see [75, p. 525])

$$w \le \frac{(r-\lambda)b}{r^2 - rb + (r-\lambda)b},$$

which simplifies to $w \le v$ using (2.4). ■

Standard maximum clique algorithms such as [17, 94] (which locate a single maximum clique of the graph) are easily modified to generate all maximum cliques. Our implementation uses the maximum clique algorithm of Östergård [94].

Although the following is not implemented in our algorithm, we remark that Spence [112] makes some useful observations on the generation of the compatibility graph and on how to prune cliques that do not correspond to a lexicographically maximal extension during the clique search.

5.2 A READ–FARADŽEV-TYPE ALGORITHM FOR RESOLUTIONS

In this section we develop an algorithm for generating all nonisomorphic resolutions of $RB(v, k, \lambda)$ designs, or equivalently, all inequivalent ED$_m$-codes with parameters $(n, M, d)_q$. This algorithm was used to settle the nonexistence of an $RB(15, 5, 4)$ design. (The algorithm that appeared in conjunction with that result in [61] is identical to the algorithm described here with the exception that the present algorithm uses Lemma 5.1.5 for pruning during the canonicity test.)

We use the coding-theoretic framework to discuss the algorithm because this is more convenient. Let n, M, d, q constitute the parameters of an ED$_m$-code. The algorithm constructs lexicographically minimal representatives of $(n, M, d)_q$ codes codeword by codeword using backtrack search combined with a maximum clique algorithm. Similarly to the algorithm presented in the previous section, the present algorithm is divided into two stages depending on the number of codewords in the input code. The first stage proceeds codeword by codeword and applies a lexicographic minimality test after each added codeword. The second stage constructs a graph from all codewords that extend the input code and performs a maximum clique search in the graph to locate extensions of the input code to an $(n, M, d)_d$ code.

5.2.1 Embedding into the Read–Faradžev framework

We represent q-ary codes of length n and cardinality M as $M \times n$ matrices over $\mathbb{Z}_q$ (recall Section 3.4). In particular, the isometry equivalence classes of $(n, M, d)_q$ codes then correspond to the orbits of the action (3.4) of $S_M \times (S_q \wr S_n)$ on $\mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n}$.

The embedding into the Read–Faradžev framework is analogous to that used for block designs. We put $X = \mathbb{Z}_q^{\mathbb{Z}_n}$, $Y = \mathbb{Z}_M$, $G = S_M \times (S_q \wr S_n)$, and let $(\sigma, (\mu, \pi)) \in G$ act on $\varphi \in Y^X$ by

$$(\sigma, (\mu, \pi))\varphi : i \mapsto (\mu, \pi)\varphi(\sigma^{-1}(i)) \qquad (5.5)$$

for all $i \in \mathbb{Z}_n$, where $(\mu, \pi) \in S_q \wr S_n$ acts on a codeword $\varphi(\sigma^{-1}(i)) \in \mathbb{Z}_q^{\mathbb{Z}_n}$ as in Theorem 3.3.7. This action on Y^X is clearly equivalent to the action (3.4) on $\mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n}$. Furthermore, the standard lexicographic order on Y^X is equivalent to the standard lexicographic order on $\mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n}$.

To generate all nonisomorphic $(n, M, d)_q$ codes, we put $P = \mathscr{L}\mathscr{C}_q(n, M, d)$ and select the lexicographic minimum of each orbit as the canonical representative:

$$C = \{A \in \mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_n} \ : \ A \leq (\sigma, (\mu, \pi))A \text{ for all } (\sigma, (\mu, \pi)) \in G\}.$$

We derive next a collection of necessary conditions for extendibility of a partial solution $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$.

Because an ED$_m$-code is equidistant and each coordinate value occurs precisely M/q times in a coordinate (Corollary 3.2.4), we have the following two elementary conditions for extendibility:

Lemma 5.2.1 *Let $M' \in \{1, 2, \ldots, M\}$. An $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in P only if $d_H(A(i_1, \cdot), A(i_2, \cdot)) = d$ for all distinct $i_1, i_2 \in \mathbb{Z}_{M'}$.*

Lemma 5.2.2 *Let $M' \in \{1, 2, \dots, M\}$. An $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in P only if $|\{i \in \mathbb{Z}_{M'} : A(i,j) = l\}| \le M/q$ for all $(j,l) \in \mathbb{Z}_n \times \mathbb{Z}_q$.*

The generic extract (4.3) for the lexicographic maximum representatives in the Read–Faradžev framework can naturally be formulated for the lexicographic minimum representatives of an orbit as well. Thus, any partial solution $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ that has an extension in C must be the lexicographic minimum of its orbit:

Lemma 5.2.3 *Let $M' \in \{1, 2, \dots, M\}$. An $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in C only if $A \le (\sigma, (\mu, \pi))A$ for all $(\sigma, (\mu, \pi)) \in S_{M'} \times (S_q \wr S_n)$.*

This implies that whenever $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in C, then its rows and columns appear in increasing lexicographic order. Furthermore, the columns must be minimal with respect to a permutation of the coordinate values $\mathbb{Z}_q$. These necessary conditions are given in the following two lemmata.

Lemma 5.2.4 *Let $M' \in \{1, 2, \dots, M\}$. An $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in C only if $A(i, \cdot) \le A(i+1, \cdot)$ for all $i \in \mathbb{Z}_{M'-1}$ and $A(\cdot, j) \le A(\cdot, j+1)$ for all $j \in \mathbb{Z}_{n-1}$.*

Proof. The rows and columns of A can be permuted in an arbitrary way by choosing a suitable pair $(\sigma, \pi) \in S_{M'} \times S_n$ and $\mu = 1$. If either the rows or the columns of A do not appear in increasing lexicographic order, then we can sort them to obtain a matrix lexicographically lesser than A, which implies that the necessary condition in Lemma 5.2.3 is not satisfied. ■

Lemma 5.2.5 *Let $M' \in \{1, 2, \dots, M\}$. An $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in C only if $A(\cdot, j) \le \nu A(\cdot, j)$ for all $j \in \mathbb{Z}_n$ and $\nu \in S_q$, where $\nu A(\cdot, j) : i \mapsto \nu(A(i,j))$ for all $i \in \mathbb{Z}_{M'}$.*

Proof. If some column $A(\cdot, j)$ is not lexicographically minimal with respect to a permutation $\nu \in S_q$ of the coordinate values, then we can minimize the column using a suitable $\mu \in S_q^{\mathbb{Z}_n}$ and obtain a matrix lexicographically lesser than A. Consequently, the necessary condition of Lemma 5.2.3 is not satisfied. ■

By the following lemma the first column of an extendible partial solution is always fixed.

Lemma 5.2.6 *Let $M' \in \{1, 2, \dots, M\}$. An $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ has an extension in $P \cap C$ only if $A(i, 0) = \lfloor i/q \rfloor$ for all $i \in \mathbb{Z}_{M'}$.*

Proof. Suppose $A' \in P \cap C$ is an extension of A. The coordinate values in column $A'(\cdot, 0)$ must form an increasing sequence, because otherwise the rows of A' are not in increasing lexicographic order. Since A' forms an ED$_m$-code, each of the q coordinate values in $\mathbb{Z}_q$ must occur exactly M/q times in $A'(\cdot, 0)$ (Corollary 3.2.4). Thus, $A'(i, 0) = \lfloor i/q \rfloor$. ■

5.2.2 Codeword generation

The codeword generation procedure generates on input $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ all codewords that extend A so that the extension satisfies a collection of necessary conditions for extendibility to an element of $P \cap C$.

The necessary conditions used by our backtrack search implementation are given in Lemmata 5.2.1 (equidistance), 5.2.2 (bound on the number of coordinate value occurences), 5.2.4 (lexicographic ordering of rows and columns), 5.2.5 (column minimality), and 5.2.6 (the first column is fixed). Clearly, A must satisfy these conditions because otherwise A has no extension in $P \cap C$.

The backtrack search constructs a codeword $x \in \mathbb{Z}_2^{\mathbb{Z}_n}$ one coordinate at a time, that is, $x(0)$ is fixed first, then $x(1)$ and so on. Backtracking is performed either after a complete solution is reached or if the current partial solution cannot be extended further without violating one or more of the necessary conditions for extendibility.

We describe next how each of the necessary conditions is incorporated into the backtrack search. The easiest condition to incorporate is the one that fixes the first column, namely, $x(0) = \lfloor M'/q \rfloor$ is fixed. Equally straightforward to check is the bound on coordinate value occurences: If coordinate value $l \in \mathbb{Z}_q$ occurs M/q times in $A(\cdot, j)$, then we must have $x(j) \neq l$. The column minimality condition is enforced by the constraint $x(j) \leq 1 + \max_{i \in \mathbb{Z}_{M'}} A(i, j)$. The lexicographic order condition is incorporated into the search in two ways. First, a partial solution must always be lexicographically greater than or equal to the lexicographically greatest row in A. This is easily arranged by restricting the choice of coordinate values. Second, if $A(\cdot, j) = A(\cdot, j-1)$, then we must have $x(j) \geq x(j-1)$, because otherwise the columns of A extended with x are clearly not in increasing lexicographic order.

The equidistance condition is the hardest to incorporate efficiently into the search. Our implementation keeps track of the Hamming-distance of the current partial solution $x(0), x(1), \ldots, x(s)$ to each of the codewords in A. Pruning occurs whenever there exists an $i \in \mathbb{Z}_{M'}$ such that $|\{j \in \mathbb{Z}_{s+1} : x(j) \neq A(i, j)\}| > d$ or $|\{j \in \mathbb{Z}_{s+1} : x(j) \neq A(i, j)\}| + (n - s - 1) < d$.

We note that during the second stage of the algorithm the following necessary conditions are not applicable: the first column condition, the column minimality condition, and the condition $x(j) \geq x(j-1)$ enforcing lexicographic ordering of the columns. This is because an extending codeword is not necessarily placed immediately after the last codeword of A during the second stage of the algorithm.

5.2.3 Canonicity test

Lexicographic minimality of an $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$ under the action of $S_{M'} \times (S_q \wr S_n)$ can be tested using an analogous backtrack search as was presented in Section 5.1.3 to test lexicographic maximality of an incidence system.

The canonicity test algorithm decides using exhaustive search whether there exists a $(\sigma, (\mu, \pi)) \in S_{M'} \times (S_q \wr S_n)$ such that $(\sigma, (\mu, \pi))A < A$. The following observation enables us to avoid a search through $S_q \wr S_n$ for every $\sigma \in S_{M'}$: Suppose that a $(\sigma, (\mu, \pi)) \in S_{M'} \times (S_q \wr S_n)$ satisfying $A' = (\sigma, (\mu, \pi))A < A$ exists. Then, the lexicographic rank of A' can only decrease when we first minimize lexicographic rank of every column of A' with respect to value permutation, and then sort the minimized columns to increasing lexicographic order. Consequently, to establish that A is the lexicographic minimum of its orbit, it suffices to check, for each $\sigma \in S_{M'}$, that the column-minimized column-sorted $(\sigma, 1)A$ is not lexicographically lesser than A.

Our backtrack search implementation of the canonicity test proceeds by fixing one image point of $\sigma \in S_{M'}$ at a time, that is, $\sigma(0)$ is fixed first, then $\sigma(1)$ and so on. We observe that from a partial solution $\sigma(0), \sigma(1), \ldots, \sigma(s)$ we can determine the restriction $(\sigma^{-1}, 1)A|_{\mathbb{Z}_{s+1}\times\mathbb{Z}_n}$. Pruning is enabled by the observation that the restriction of the column-minimized and column-sorted $(\sigma^{-1}, 1)A$ to $\mathbb{Z}_{s+1} \times \mathbb{Z}_n$ remains invariant in all extensions of the partial solution $\sigma(0), \ldots, \sigma(s)$. Consequently, as soon as the column-minimized and sorted $(\sigma^{-1}, 1)A|_{\mathbb{Z}_{s+1}\times\mathbb{Z}_n}$ differs from $A|_{\mathbb{Z}_{s+1}\times\mathbb{Z}_n}$, we can either prune the partial solution ($A|_{\mathbb{Z}_{s+1}\times\mathbb{Z}_n}$ is lexicographically lesser), or conclude that A is not the lexicographic minimum of its orbit ($A|_{\mathbb{Z}_{s+1}\times\mathbb{Z}_n}$ is lexicographically greater).

Lemma 5.1.5 can also be applied in this context for pruning:

Corollary 5.2.7 *Let $S_{M'} \times (S_q \wr S_n)$ act on $\mathbb{Z}_q^{\mathbb{Z}_{M'}\times\mathbb{Z}_n}$ by (3.4), let $U \leq S_{M'}$, and suppose $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'}\times\mathbb{Z}_n}$ satisfies $(\sigma, (\mu, \pi))A \leq A$ for all $(\sigma, (\mu, \pi)) \in U \times (S_q \wr S_n)$. Then, $(\sigma\varrho, (\mu, \pi))A \leq A$ for all $(\sigma, (\mu, \pi)) \in U \times (S_q \wr S_n)$ whenever $(\hat{\sigma}\varrho, (\hat{\mu}, \hat{\pi}))A = A$ for some $\varrho \in S_{M'}$, $\hat{\sigma} \in U$, and $(\hat{\mu}, \hat{\pi}) \in S_q \wr S_n$.*

Pseudocode for the canonicity test algorithm that realizes these observations is given as Algorithm 5. The chain of subgroups used in the search is the same as was used in Algorithm 4.

We note that the set of automorphisms discovered during the course of Algorithm 4 for an $A \in P \cap C$ is a strong genating set for the full automorphism group of the resolution that corresponds to A.

5.2.4 Clique search

Let $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'}\times\mathbb{Z}_n}$ be input by the first stage of the algorithm to the second stage. (So, A satisfies the necessary conditions of Lemmata 5.2.3 and 5.2.1.)

The second stage of the algorithm is analogous to the second stage of the block design generation algorithm presented in Section 5.1.4. First, the compatibility graph of A is generated:

Definition 5.2.8 The *compatibility graph* $\mathscr{G}_d(A)$ of an $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'}\times\mathbb{Z}_n}$ is the graph whose vertices consist of all codewords $x \in \mathbb{Z}_q^{\mathbb{Z}_n}$ that satisfy $d_H(A(i, \cdot), x) = d$ and $x \geq A(i, \cdot)$ for all $i \in \mathbb{Z}_{M'}$; any two vertices x_1, x_2 are connected by an edge if and only if $d_H(x_1, x_2) = d$.

Then, all extensions of A to an ED$_m$-code are located using a maximum clique algorithm to generate all $(M - M')$-cliques in $\mathscr{G}_d(A)$. Each extension is tested for lexicographic minimality and, if the extension is minimal, accepted as an element of $P \cap C$ by the algorithm.

Algorithm 5 Canonicity test for a matrix $A \in \mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$.

```
function COSET-SEARCH($\sigma$ : permutation, $i, l, M'$ : integer) : integer
  if $i = M'$ then
    (* Automorphism $\sigma$ discovered. *)
    return 0
  end if
  if $i < l$ then
    $J \leftarrow \{i\}$
  else
    $J \leftarrow \{i, i+1, \dots, M'-1\}$
  end if
  if $i = l$ then
    (* Coset of the identity has been tested before, remove it. *)
    $J \leftarrow J \setminus \{i\}$
  end if
  for all $j \in J$ do
    $\sigma' \leftarrow \sigma \circ (i\ j)$
    put $A'(s,t) = A(\sigma'(s), t)$ for all $(s,t) \in \mathbb{Z}_{i+1} \times \mathbb{Z}_n$
    minimize lexicographic rank of each column of $A'$ by value permutation
    sort minimized columns of $A'$ to increasing lexicographic order
    if $A' \leq A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_n}$ then
      if $A' < A|_{\mathbb{Z}_{i+1} \times \mathbb{Z}_n}$ then
        (* Counterexample found. *)
        return -1
      end if
      $d \leftarrow$ COSET-SEARCH($\sigma', i+1, l, M'$)
      if $d < 0 \vee (d = 0 \wedge i > l)$ then
        (* Counterexample found or Corollary 5.1.6 applies. *)
        return $d$
      end if
    end if
  end for
  return 1
function CANONICAL($A$ : an element of $\mathbb{Z}_q^{\mathbb{Z}_{M'} \times \mathbb{Z}_n}$) : boolean
  for $l = M', M'-1, \dots, 0$ do
    (* Test left cosets of the stabilizer of $0, 1, \dots, l$ in $S_{M'}$. *)
    if COSET-SEARCH(id, $0, l, M'$) $< 0$ then
      return FALSE
    end if
  end for
  return TRUE
```

We observe that the vertices of a clique of order $M - M'$ in $\mathscr{G}_d(A)$ clearly complete A to an ED$_m$-code. Conversely, if A has an extension in $P \cap C$, then the extending codewords appear as the vertices of an $(M - M')$-clique in $\mathscr{G}_d(A)$. Furthermore, if A satisfies the necessary conditions in Lemma 5.2.1, then a clique in $\mathscr{G}_d(A)$ has order at most $M - M'$; this is because ED$_m$-codes are optimal by Corollary 3.2.5. Thus, the second stage of the algorithm correctly produces all extensions of A in $P \cap C$.

5.3 A MCKAY-TYPE ALGORITHM FOR RESOLUTIONS

In this section we develop an alternative algorithm, based on the McKay framework, for generating all nonisomorphic resolutions of $RB(v, k, \lambda)$ designs. We continue to use the coding-theoretic framework, and assume that n, M, d, q constitute the parameters of an ED$_m$-code. The algorithm performs a backtrack search for all nonisomorphic $(n, M, d)_q$ codes coordinate by coordinate.

5.3.1 Embedding into the McKay framework

We embed coordinatewise code construction into the McKay framework as follows.

The set of labelled objects X is formed as the disjoint union $X = \cup_{s=1}^{n} X_s$, where $X_s = \mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_s}$. The group G acting on X is the (formal) direct product $G = \Pi_{s=1}^{n} G_s$, where $G_s = S_M \times (S_q \wr S_s) \times S_q$. The action of G_s on X_s is defined by

$$(\sigma, (\mu, \pi), \nu)A : (i, j) \mapsto \mu_j(A(\sigma^{-1}(i), \pi^{-1}(j))) \tag{5.6}$$

for all $(i, j) \in \mathbb{Z}_M \times \mathbb{Z}_s$, all $(\sigma, (\mu, \pi), \nu) \in G_s$, and all $A \in X_s$. (Note this is the same as action (3.4).)

The sets of upper and lower objects associated to an $A \in X_s$ are

$$L(A) = \begin{cases} \emptyset & \text{if } s = 1, \\ \{(A, t) \, : \, t \in \mathbb{Z}_s\} & \text{if } s = 2, 3, \ldots, n; \end{cases} \tag{5.7}$$

$$U(A) = \begin{cases} \{(A, c) \, : \, c \in \mathbb{Z}_q^{\mathbb{Z}_M}\} & \text{if } s = 1, 2, \ldots, n-1, \\ \emptyset & \text{if } s = n. \end{cases} \tag{5.8}$$

The intuition behind the lower objects is that the second component of a pair (A, t) indicates which coordinate is to be deleted from A to obtain a smaller labelled object. For an upper object (A, c), the second component $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$ describes the new coordinate to be added to A to obtain a larger labelled object. These collections of upper and lower objects are clearly disjoint for distinct A. For notational convenience we put $\check{X} = \cup_{s=1}^{n} \check{X}_s$ and $\hat{X} = \cup_{s=1}^{n} \hat{X}_s$, where $\check{X}_s = \cup_{A \in X_s} L(A)$, $\hat{X}_s = \cup_{A \in X_s} U(A)$.

The action of G_s is extended from X_s to $\check{X}_s$ and $\hat{X}_s$ by defining

$$(\sigma, (\mu, \pi), \nu)(A, t) = ((\sigma, (\mu, \pi))A, \pi(t)), \tag{5.9}$$

$$(\sigma, (\mu, \pi), \nu)(A, c) = ((\sigma, (\mu, \pi))A, \nu \circ c \circ \sigma^{-1}) \tag{5.10}$$

for all $(\sigma, (\mu, \pi), \nu) \in G_s$ and all $A \in X_s$, $t \in \mathbb{Z}_s$, and $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$.

The upper and lower objects are connected by defining $R \subseteq \check{X} \times \hat{X}$ to be the set that consists of all pairs $((A,t),(B,c)) \in \check{X}_{s+1} \times \hat{X}_s$, $s = 1,2,\ldots,n-1$, that satisfy

$$(\sigma,(\mu,\pi))B(i,j) = \begin{cases} A(i,j) & \text{if } j < t \\ A(i,j+1) & \text{if } j \geq t \end{cases}, \tag{5.11}$$

$$\nu \circ c \circ \sigma^{-1}(i) = A(i,t) \tag{5.12}$$

for some $(\sigma,(\mu,\pi),\nu) \in G_s$. Consequently, a lower object $(A,t) \in \check{X}_{s+1}$ is related through R to precisely those upper objects that can be obtained from A by separating coordinate $t \in \mathbb{Z}_{s+1}$ as $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$ and then relabelling the resulting pair (B,c) using a group element in G_s. Conversely, an upper object $(B,c) \in \hat{X}_s$ is related to precisely those lower objects that can be obtained by inserting c to any coordinate position $t \in \mathbb{Z}_{s+1}$ and then relabelling the resulting pair (A,t) using a group element G_{s+1}.

It is straightforward to verify that these definitions satisfy axioms (C1)-(C5) of the McKay framework.

We define the order of a labelled object to be $o(A) = s$ for every $A \in X_s \subseteq X$. Axioms (O1) and (O2) are then clearly satisfied.

It remains to define a function $m : X \to \mathscr{P}[\check{X}]$ that satisfies axioms (M1)-(M3). Our construction for an m-function was inspired by the one in [84], which was used to generate triangle-free graphs. Let $c : X \to X$ be an arbitrary canonical placement map with respect to G (recall Definition 4.0.1). For convenience we write A^* for the canonical representative associated to an $A \in X$ by c. A canonical placement map associates to every $A \in X$ a unique *canonical placement coset* $gG_A \in G/G_A$ that satisfies $gG_A A = A^*$ (cf. [3]), where G_A is the stabilizer of A in G as usual. Suppose now that $A \in X$ and that $gG_A \in G/G_A$ is the corresponding canonical placement coset. We define

$$m(A) = \begin{cases} \emptyset & \text{if } o(A) = 1, \\ G_A g^{-1}(A^*,0) & \text{if } o(A) > 1. \end{cases} \tag{5.13}$$

Lemma 5.3.1 *The function m is well-defined and satisfies axioms (M1)-(M3) of the McKay framework.*

Proof. Clearly, $L(A) = \emptyset$ if and only if $o(A) = 1$. Thus, axiom (M1) is satisfied. Henceforth we assume $o(A) > 1$. Select an $A \in X$, and suppose that $gG_A A = A^*$ for some $g \in G$. We first show that m is well-defined. It is clear by (5.9) that $m(A) \subseteq L(A)$ for all $A \in X$. Select any $g_1 \in G$ such that $g_1 G_A A = A^*$. Then we must have $g_1^{-1}g \in G_A$. Select any $(A,t) \in G_A g^{-1}(A^*,0)$ and denote by $h \in G_A$ a solution to $(A,t) = hg^{-1}(A^*,0)$. Then, $(A,t) = h(g_1^{-1}g)^{-1}g_1^{-1}(A^*,0) \in G_A g_1^{-1}(A^*,0)$. Thus, $G_A g^{-1}(A^*,0) \subseteq G_A g_1^{-1}(A^*,0)$. The proof of the reverse inclusion is similar, and m is well-defined. Axiom (M2) is satisfied by (5.13) because $g^{-1}(A^*,0) \in L(A)$ and $m(A) = G_A g^{-1}(A^*,0)$. For (M3), select any $\tilde{g} \in G$. Then, $g(\tilde{g}^{-1}\tilde{g})G_A(\tilde{g}^{-1}\tilde{g})A = (g\tilde{g}^{-1})G_{\tilde{g}A}(\tilde{g}A) = A^*$ since $G_{\tilde{g}A} = \tilde{g}G_A\tilde{g}^{-1}$. Consequently, the coset $(g\tilde{g}^{-1})G_{\tilde{g}A}$ is the canonical placement coset of $\tilde{g}A$, and $m(\tilde{g}A) = G_{\tilde{g}A}(g\tilde{g}^{-1})^{-1}(A^*,0) = \tilde{g}G_A\tilde{g}^{-1}\tilde{g}g^{-1}(A^*,0) = \tilde{g}m(A)$. ■

5.3.2 Top-level algorithm

The embedding into the McKay framework presented in the previous section leads to a straightforward backtrack search implementation of Algorithm 2.

The algorithm accepts as input a partial solution $A \in X_s = \mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_s}$, where $s = 1, \ldots, n$. Also input to the algorithm is a strong generating set for the *point automorphism group* of A:

$$\mathrm{Aut}_P(A) = \{\sigma \in S_M \, : \, \exists(\sigma, (\mu, \pi), \nu) \in G_s \; (\sigma, (\mu, \pi), \nu)A = A\}.$$

If $s < n$, the algorithm performs a backtrack search for coordinates $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$ that extend A and are the lexicographic minimum representatives of their G_A-orbits on $U(A)$. (This procedure is described in Section 5.3.3.) Whenever an extending coordinate c is found, the algorithm constructs $A' \in X_{s+1}$ defined by

$$A'(i,j) = \begin{cases} A(i,j) & \text{if } j \in \mathbb{Z}_s; \\ c(i) & \text{if } j = s \end{cases} \tag{5.14}$$

for all $(i,j) \in \mathbb{Z}_M \times \mathbb{Z}_{s+1}$, and then evaluates the m-function for A'. (The m-function implementation is described in 5.3.4) If $(A', s) \in m(A')$, then the algorithm proceeds recursively to extend A' and its point automorphism group, which was computed as a side effect of the m-function evaluation; otherwise the algorithm considers the next $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$.

5.3.3 Coordinate generation

Given $A \in X$, the coordinate generation procedure produces an orbit representative for each orbit of G_A on $U(A)$ (cf. Algorithm 2). By (5.10) this is equivalent to producing a transversal for the orbits $\mathrm{Aut}_P(A) \times S_q \backslash\backslash \mathbb{Z}_q^{\mathbb{Z}_M}$, where $(\sigma, \nu) \in \mathrm{Aut}_P(A) \times S_q$ acts on a $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$ by $(\sigma, \nu)c = \nu \circ c \circ \sigma^{-1}$. Our implementation is a Read–Faradžev-type orbit transversal algorithm based on backtrack search, which uses the lexicographic minimum element of each orbit in $\mathrm{Aut}_P(A) \times S_q \backslash\backslash \mathbb{Z}_q^{\mathbb{Z}_M}$ as the canonical representative.

Naturally, not all such orbit representatives need to be generated:

Lemma 5.3.2 *Let $1 \le s \le n-1$, $A \in X_s$, $c \in \mathbb{Z}_q^{\mathbb{Z}_M}$, and suppose A' is defined by (5.14). Then, A' can be extended to an $(n, M, d)_q$ code only if*

(i) for all $i_1, i_2 \in \mathbb{Z}_M$, if $d_H(A(i_1, \cdot), A(i_2, \cdot)) = d$, then $c(i_1) = c(i_2)$; and

(ii) for all $i_1, i_2 \in \mathbb{Z}_M$, if $d_H(A(i_1, \cdot), A(i_2, \cdot)) + n - s = d$ and $i_1 \neq i_2$, then $c(i_1) \neq c(i_2)$; and

(iii) $|\{i \in \mathbb{Z}_M \, : \, c(i) = l\}| = M/q$ for all $l \in \mathbb{Z}_q$.

Proof. If at least one condition does not hold, then A' and consequently all of its extensions violate Corollary 3.2.4. ■

Our backtrack search implementation proceeds by fixing first $c(0)$, then $c(1)$, and so forth. Each partial solution $c(0), c(1), \ldots, c(t)$ must satisfy conditions (i) and (ii) of Lemma 5.3.2; condition (iii) can naturally only partly be checked for a partial solution. After a value $c(t)$ has been fixed, the partial solution $c(0), c(1), \ldots, c(t)$ is tested for lexicographic minimality with respect to the subgroup $H \times S_q \leq \mathrm{Aut}_P(A) \times S_q$, where H is the pointwise stabilizer of $t+1, \ldots, M-1$ in $\mathrm{Aut}_P(A)$. (This test is equivalent to applying extract (4.3) to prune the search in the Read–Faradžev framework.) The minimality test is implemented as an exhaustive search through all elements of H (see [14, Ch. 10] on how to implement such a search). For each $\sigma \in H$, the lexicographic rank of $(\sigma, 1)c|_{\mathbb{Z}_t}$ is minimized with respect to permutation of the values $\mathbb{Z}_q$, and the result is compared with $c|_{\mathbb{Z}_t}$. As a result of the comparison, the partial solution is either rejected as nonminimal, or the search continued if no counterexample to minimality was found. (We point out that Lemma 5.1.5 can also be applied in this context.)

If $\mathrm{Aut}_P(A)$ has large order, then we apply the minimality test only up to a certain depth in the search so that the order of H will not grow too large (and the search too expensive). In this case we need to supplement the partial minimality test with explicit isomorphism testing (cf. Algorithm 3). Isomorphism testing is facilitated by the m-function implementation, which as a side effect computes a canonical representative for the extended code A'.

5.3.4 Canonical placement of codes using *nauty*

Our implementation of the m-function (5.13) uses the the graph canonical labelling package *nauty* [81, 82] to compute a canonical placement coset for $A \in X_s$.

We first transform an $A \in X_s$ into a graph. (This transformation is from [92].) The graph consists of $M + sq$ vertices. The first M vertices correspond to the M codewords in A, and the remaining sq vertices encode the q coordinate values in each of the s coordinates. Edges are inserted to the graph so that each of the q coordinate-value vertices in every coordinate is connected by an edge to every other coordinate-value vertex in the same coordinate. Furthermore, for every $(i, j) \in \mathbb{Z}_M \times \mathbb{Z}_s$, the codeword vertex that corresponds to codeword i is connected by an edge to the coordinate-value vertex that corresponds to the value $A(i, j)$ in coordinate j. Finally, the vertices are colored using one color for the M codeword vertices and another color for the sq coordinate-value vertices.

After the graph has been constructed, we apply *nauty* to compute the canonical placement coset of the graph, that is, a permutation of the vertices that transforms the graph into its canonical representative combined with a strong generating set for the full automorphism group of the graph.

In computing canonical placement and the full automorphism group, *nauty* never maps vertices of one color into vertices of the other. Consequently, *nauty* computes canonical placement for elements of X_s with respect to G_s. (Two graphs, constructed as above, have identical canonical representatives if and only if they are related by a permutation of the codeword vertices combined with a permutation of the coordinates and the coordinate values in each coordinate; cf. (5.6).)

The m-function test condition $(A, s-1) \in m(A)$ is easy to check because *nauty* also computes the automorphism-orbits of the vertices of the graph.

Furthermore, we obtain a strong generating set for $\mathrm{Aut}_P(A)$ by restricting the reported generator permutations for the full automorphism group to the codeword vertices.

6 RESULTS

We implemented the three exhaustive generation algorithms described in the previous chapter as C programs, and ran them on a number of different design parameter families. In the process we were able to produce new complete classifications for several families of block designs, resolvable block designs, and their resolutions. Additionally, we were able to corroborate several recent classification results of other researchers.

The new classification results and the verified old classifications are summarized in Tables 6.1 and 6.2 for ease of reference. Column "No" in the tables indicates the block design parameter family number in [78]; columns v, k, λ, r, b give the corresponding parameters. (A "-" in the "No" column indicates that the parameter family does not lie in the range $3 \leq r \leq 41$ covered by [78].) The number of nonisomorphic block designs, resolvable block designs, and resolutions in a family are given in columns "Nd", "Nrd", and "Nr", respectively.

The following results on block designs were obtained as a result of this work. The classifications of the $B(13, 6, 5)$ and the $B(14, 7, 6)$ designs are discussed in Sections 6.1 and 6.2. To the best of our knowledge, both of these classifications are new. Additionally, we verified the classification results of Spence on $B(31, 10, 3)$ designs [110] and on $B(23, 11, 5)$ designs [111]. (We remark that both Table 1 in [111] and table entry number 63 in [78] contain an error. The correct number of nonisomorphic $B(23, 11, 5)$ designs is 1106; this is implicit in [111, p. 192–196].) In both verifications we applied the block design generation algorithm described in Section 5.1. The $B(31, 10, 3)$ verification was conducted with $p = 10$ and the $B(23, 11, 5)$ verification with $p = 7$.

For resolvable block designs, we were able to settle the nonexistence of an $RB(15, 5, 4)$ design and produce complete classifications of the $RB(16, 4, 2)$, the affine $RB(24, 12, 11)$, and the $RB(14, 7, 12)$ designs. These are discussed in Sections 6.3, 6.4, 6.5, and 6.6, respectively. Additionally, the classifications of the resolutions of $RB(9, 3, \lambda)$ designs for $\lambda = 3, 4, 5$ and the $RB(9, 3, \lambda)$ designs for $\lambda = 3, 4$ have been submitted for separate publication [96].

We corroborate the following classification results on resolvable block designs obtained by other researchers. First, we confirm the result of Lam and Tonchev [70] that there are 68 nonisomorphic affine $RB(27, 9, 4)$ designs. Second, we verified the result of Kocay and van Rees [66] that there are 5 and 3 nonisomorphic $RB(16, 8, 7)$ and $RB(20, 10, 9)$ designs, respectively. Third, we confirm the number of nonisomorphic $RB(8, 4, 9)$ and $RB(8, 4, 12)$ designs as 10 and 31, respectively, and continue the enumeration of the $RB(8, 4, 3\lambda)$ design family up to $3\lambda = 21$. Fourth, we corroborate the recent classification result of Morales and Velarde [90] that there are 5 nonisomorphic $RB(12, 4, 3)$ designs, each of which has a unique resolution. Finally, we note that table entry number 195 in [78] contains an error; the correct number of nonisomorphic $RB(10, 5, 8)$ designs is 5.

The classification runs were distributed over a network of twenty 200–500MHz PCs using the batch system autoson [83]. The resolutions of the

Table 6.1: The $B(v, k, \lambda)$ design families classified.

No	v	k	λ	Nd	Comments
54	31	10	3	151	Classified in [110].
63	23	11	5	1106	Classified in [111] (see text).
77	13	6	5	19072802	New. See Section 6.1.
89	14	7	6	15111019	New. See Section 6.2.

Table 6.2: The $RB(v, k, \lambda)$ design families classified.

No	v	k	λ	Nr	Nrd	Comments
44	16	4	2	339592	325062	New. See Section 6.4.
56	12	4	3	5	5	Classified in [90]
66	9	3	3	426	395	New [96].
90	27	9	4	68	68	Classified in [70].
102	15	5	4	0	0	New. See Section 6.3.
130	16	8	7	5	5	Classified in [66].
145	9	3	4	149041	119985	New [96].
195	10	5	8	5	5	Error in [78].
224	20	10	9	3	3	Classified in [66].
235	9	3	5	203047732	?	New [96].
278	8	4	9	10	10	Classified by Spence [78].
346	24	12	11	130	130	See Section 6.5.
451	14	7	12	1363486	1363486	New. See Section 6.6.
524	8	4	12	31	31	Known [78].
819	8	4	15	82	82	New.
-	8	4	18	240	240	New.
-	8	4	21	650	650	New.

$RB(9,3,5)$ designs took the longest to classify (slightly over 100 days total CPU time). The second longest classification was that of the $RB(14,7,2)$ designs, which required six days total CPU time. All the other classifications were completed in total CPU time ranging from a few seconds to a few days.

Because the classifications of designs and resolutions presented in this chapter were obtained using computer search, the results are correct if the computer programs are correct and no hardware error occurs. To guard against hardware errors, all of the classification runs were conducted at least twice. Confidence in the correctness of the computer programs was acquired from the correct verifications of earlier results obtained. Nevertheless, an independent verification would naturally be desirable.

6.1 CLASSIFICATION OF $B(13,6,5)$ DESIGNS

The classification of the $B(14,7,6)$ designs was obtained using the block design generation algorithm from Section 5.1 with $p = 7$. Altogether there are 19072802 nonisomorphic $B(13,6,5)$ designs, of which 19063352 have a trivial full automorphism group.

In addition to enumerating the designs, the following additional data was computed. For each canonical representative generated by the algorithm, the order of its full automorphism group was determined from the strong generating set computed during the canonicity test. The orders of the automorphism groups of $B(13,6,5)$ designs are given in Table 6.3. Column "$|\operatorname{Aut}(\mathcal{B})|$" indicates the order of the full automorphism group, and column "Nd" indicates the number of nonisomorphic designs with this full automorphism group order.

Table 6.3: Properties of $B(13,6,5)$ designs.

$\|Aut(\mathcal{B})\|$	Nd
1	19063352
2	7619
3	1651
4	113
6	53
12	10
13	1
39	2
156	1
Total	19072802

We discuss the four $B(13,6,5)$ designs that have the largest automorphism groups as an example. All four designs are *transitive*, that is, their full automorphism group is transitive. (Based on Table 6.3, these are also the only transitive $B(13,6,5)$ designs.) The lexicographic maximum incidence matrices of the designs are listed below.

$$\mathcal{B}_1 : \begin{array}{l}
11111111111110000000000000\\
11111000000001111111000000\\
11000111000011100001111000\\
10100100110010011001100110\\
10010010101001010100011110\\
10000001011100101111010100\\
01100100001101010011010011\\
01010000110100111000111001\\
01001001101010001101001011\\
00110011010010000110110011\\
00101010100101100101100101\\
00011101000101001010101110\\
00001110011010110010001101
\end{array}
\quad
\mathcal{B}_2 : \begin{array}{l}
11111111111110000000000000\\
11111000000001111111000000\\
11110100000001000000111110\\
11000011100001110001110001\\
10100011010001001100001111\\
10010000101100001111101001\\
01000110011010101011000101\\
01000100011101110100101010\\
00101101001000011010110011\\
00101010100100110011001110\\
00011010010110010100110101\\
00010001101111101000010110\\
00001101110011000111011000
\end{array}$$

$$\mathcal{B}_3 : \begin{array}{l}
11111111111110000000000000\\
11111000000001111111000000\\
11100110000001100000111100\\
11000000111000011100111001 0\\
10011100000100010010110101 1\\
10000101100110110010001101\\
01010010001100011011011001\\
01010001010111000101000111\\
00101011010000101100011101\\
00101000111001010011001110\\
00100110100101011100100011\\
00010100111010001110110100\\
00001011001111100010110010
\end{array}
\quad
\mathcal{B}_4 : \begin{array}{l}
11111111111110000000000000\\
11111000000001111111000000\\
11100110000001100000111100\\
11010001100000110001110011\\
11000100011000101100001111\\
10000001110011000111100101 0\\
01000001001111001101011000 1\\
00101010010101110001001011\\
00100101100101011100100101\\
00100000111101110001001011 0\\
00011110000100100110110110\\
00011101001010010011001101\\
00011000111001001101111000
\end{array}$$

The automorphism groups of the designs are:

$$\begin{aligned}
\mathrm{Aut}(\mathcal{B}_1) &= \langle (1\ 4\ 5\ 6\ 10\ 9\ 2\ 8\ 7\ 3\ 12\ 11),\ (0\ 6\ 12\ 8\ 9\ 2\ 5\ 7\ 1\ 11\ 4\ 10\ 3) \rangle,\\
\mathrm{Aut}(\mathcal{B}_2) &= \langle (1\ 6\ 11)(2\ 7\ 5)(3\ 8\ 10)(4\ 12\ 9),\ (0\ 1\ 2)(3\ 4\ 5)(6\ 8\ 10)(7\ 9\ 11) \rangle,\\
\mathrm{Aut}(\mathcal{B}_3) &= \langle (1\ 3\ 8)(2\ 7\ 9)(4\ 5\ 12)(6\ 11\ 10),\ (0\ 1\ 4)(2\ 8\ 11)(3\ 12\ 6)(5\ 9\ 7) \rangle,\\
\mathrm{Aut}(\mathcal{B}_4) &= \langle (0\ 1\ 9\ 4\ 8\ 3\ 2\ 7\ 10\ 11\ 6\ 12\ 5) \rangle.
\end{aligned}$$

Group $\mathrm{Aut}(\mathcal{B}_1)$ is primitive, has order 156, and is isomorphic to the group $\mathrm{AGL}_1(13)$. Groups $\mathrm{Aut}(\mathcal{B}_2)$ and $\mathrm{Aut}(\mathcal{B}_3)$ are both primitive, have order 39, and are isomorphic to the Frobenius group $F_{13,3}$. Group $\mathrm{Aut}(\mathcal{B}_4)$ is transitive and is evidently isomorphic to the cyclic group $\mathbb{Z}_{13}$. (The group properties were determined using the `GAP` system [39] and its library of primitive groups. The group names are from [18].)

6.2 CLASSIFICATION OF $B(14,7,6)$ DESIGNS

The classification of the $B(14,7,6)$ designs was obtained using the block design generation algorithm from Section 5.1 with $p = 7$. Altogether there are 15111019 nonisomorphic $B(14,7,6)$ designs, of which 15097318 have a trivial full automorphism group.

The orders of the full automorphism groups of the $B(14,7,6)$ designs are listed in Table 6.4.

The seven $B(14,7,6)$ designs with the largest automorphism groups are shown below.

$$\mathcal{B}_1 : \begin{array}{l} 11111111111110000000000000 \\ 11111100000001111111000000 \\ 11110011000001100000111110 \\ 11001100110000011000111101 \\ 11000010101101000111110001 \\ 11000000011110110110001110 \\ 10101001001010001101101011 \\ 01000011110011111001000011 \\ 00110100110010100110110011 \\ 00110011100100011110001101 \\ 00110000011111011001110100 \\ 00011100101101100001001111 \\ 00001111010011000111011100 \\ 00001111001100111010110010 \end{array} \quad \mathcal{B}_2 : \begin{array}{l} 11111111111110000000000000 \\ 11111100000001111111000000 \\ 11110011000001100000111110 \\ 11100000111000011100111001 \\ 10011010100100011010100111 \\ 10001011010101000111011001 \\ 10000101001110111010011100 \\ 01010100010111010001110101 \\ 01001101101001001001001111 \\ 01001010101010100111110100 \\ 00110001110010101011010011 \\ 00101001001111110100100011 \\ 00100110110011010110001110 \\ 00010110011100101101101010 \end{array}$$

$$\mathcal{B}_3 : \begin{array}{l} 11111111111110000000000000 \\ 11111100000001111111000000 \\ 11110011000001100000111110 \\ 11100000111000011100111001 \\ 10011010100100011010100111 \\ 10001001011101100110010101 \\ 10000011011011011011001010 \\ 01011000010110110001011011 \\ 01001101100010001110011110 \\ 01000111001100101101100011 \\ 00110010101010100111001101 \\ 00101110010011001001110101 \\ 00100100101111110010110010 \\ 00010101110101010101101100 \end{array} \quad \mathcal{B}_4 : \begin{array}{l} 11111111111110000000000000 \\ 11111100000001111111000000 \\ 11110011000001100000111110 \\ 11001010110000011100111001 \\ 10101000101101000011110101 \\ 10010100011100110010101011 \\ 10000111001010011011011100 \\ 01100010001111001110001011 \\ 01010101100010000111110011 \\ 01000000111011111010100100 \\ 00110001111000101101001101 \\ 00101100100110110100011110 \\ 00011010011011010101100110 \\ 00000111101010110100101001 1 \end{array}$$

$$\mathcal{B}_5 : \begin{array}{l} 11111111111110000000000000 \\ 11111100000001111111000000 \\ 11100011100001110000111100 \\ 11010010011001001100110011 \\ 10101001010100001110101110 \\ 10001110001010110010101011 \\ 10000100111101101001001101 \\ 01100100000111001100111011 0 \\ 01010001101100110110000111 \\ 01001011000110101101011001 \\ 00110101100011001010011011 \\ 00100010110111010111100001 \\ 00011110110000010101011110 \\ 00011001011011100011110100 \end{array} \quad \mathcal{B}_6 : \begin{array}{l} 11111111111110000000000000 \\ 11111100000001111111000000 \\ 11110011000001100000111110 \\ 11001010110000011100111001 \\ 10101001101000010011100111 \\ 10010100011100001110110110 \\ 10000101110101101001001101 \\ 01100001010111000111110001 \\ 01010000101110110101101100 \\ 01000111001010111010010101 \\ 00101110001100100101011011 \\ 00100110110011010110001110 \\ 00011010100111101010100011 \\ 00011001011011011001011010 \end{array}$$

$$
\begin{array}{rl}
\mathcal{B}_7 : & 11111111111110000000000000 \\
& 11111100000001111111000000 \\
& 11100011100001110000111100 \\
& 11010010011001001100110011 \\
& 10110001010100100011101011 \\
& 10001110110000001011011110 \\
& 10001001101101010110000111 \\
& 01010101001010100110011110 \\
& 01001011000110011101101010 \\
& 01000000111111101011100100 \\
& 00111000101010011010111001 \\
& 00101101010011000101110101 \\
& 00100110011100111100001101 \\
& 00010110100111110001010011
\end{array}
$$

The automorphism groups of the designs are:

$$
\begin{aligned}
\mathrm{Aut}(\mathcal{B}_1) &= \langle (0\ 7\ 1\ 8\ 3\ 5\ 13\ 12\ 10\ 9\ 4\ 2\ 11),\ (1\ 10\ 13\ 2\ 5\ 12)(3\ 8\ 11\ 9\ 4\ 7) \rangle, \\
\mathrm{Aut}(\mathcal{B}_2) &= \langle (0\ 1\ 2)(4\ 7\ 10)(5\ 8\ 11)(6\ 9\ 12),\ (1\ 9\ 10)(2\ 8\ 12)(4\ 11\ 7)(5\ 6\ 13) \rangle, \\
\mathrm{Aut}(\mathcal{B}_3) &= \langle (0\ 1\ 2)(4\ 7\ 10)(5\ 9\ 11)(6\ 8\ 12),\ (1\ 5\ 12)(2\ 6\ 10)(4\ 9\ 11)(7\ 13\ 8) \rangle, \\
\mathrm{Aut}(\mathcal{B}_4) &= \langle (0\ 1\ 2)(3\ 4\ 7)(5\ 10\ 8)(6\ 11\ 9),\ (1\ 8\ 11)(2\ 9\ 4)(3\ 10\ 7)(5\ 6\ 13) \rangle. \\
\mathrm{Aut}(\mathcal{B}_5) &= \langle (1\ 6\ 13)(2\ 11\ 9)(3\ 12\ 5)(7\ 8\ 10),\ (0\ 1\ 5\ 13\ 11\ 3\ 12\ 8\ 7\ 6\ 9\ 10\ 2) \rangle. \\
\mathrm{Aut}(\mathcal{B}_6) &= \langle (0\ 1\ 7\ 5\ 2\ 11\ 3\ 4\ 10\ 8\ 12\ 13\ 6) \rangle. \\
\mathrm{Aut}(\mathcal{B}_7) &= \langle (0\ 1\ 4\ 7\ 5\ 2\ 13\ 6\ 12\ 3\ 9\ 8\ 11) \rangle.
\end{aligned}
$$

Group $\mathrm{Aut}(\mathcal{B}_1)$ has order 78 and is isomorphic to the Frobenius group $F_{13,6}$. Groups $\mathrm{Aut}(\mathcal{B}_2)$, $\mathrm{Aut}(\mathcal{B}_3)$, $\mathrm{Aut}(\mathcal{B}_4)$, and $\mathrm{Aut}(\mathcal{B}_5)$ have order 39 and are isomorphic to the Frobenius group $F_{13,3}$. Groups $\mathrm{Aut}(\mathcal{B}_6)$ and $\mathrm{Aut}(\mathcal{B}_7)$ are clearly isomorphic to the cyclic group C_{13}. Each of the automorphism groups above fixes exactly one point. The points fixed are 6,3,3,12,4,6, and 10, respectively.

Table 6.4: Properties of $B(14, 7, 6)$ designs.

$\lvert\mathrm{Aut}(\mathcal{B})\rvert$	Nd
1	15097318
2	10934
3	2514
4	143
6	98
12	5
13	2
39	4
78	1
Total	15111019

6.3 NONEXISTENCE OF $RB(15, 5, 4)$ DESIGNS

The Read–Faradžev-type algorithm for generation of resolutions of block designs from Section 5.2 was used to settle the nonexistence of an $RB(15, 5, 4)$ design by exhaustive search. The algorithm was used with $p = 5$ to generate the corresponding $(14, 15, 10)_3$-codes; none of the 74 compatilibity graphs generated contained a clique of order exceeding 7, so no $(14, 15, 10)_3$-code and hence no $RB(15, 5, 4)$ design can exist.

6.4 CLASSIFICATION OF $RB(16, 4, 2)$ DESIGNS

The Read–Faradžev-type algorithm for generation of resolutions of block designs from Section 5.2 was used to generate all $RB(16, 4, 2)$ designs and their resolutions as follows. First, all 339592 nonisomorphic resolutions of $RB(16, 4, 2)$ designs were generated and stored to disk using the algorithm with $p = 8$. Then, a postprocessing stage was invoked, which determined for each resolution the underlying block design and converted the block design to its canonical representative using *nauty* (cf. Section 5.3.4). The canonical representatives of the underlying block designs were then sorted to determine which of the resolutions had isomorphic underlying block designs, that is, identical canonical representatives for the block designs. In the process the orders of the full automorphism groups of both the resolutions and the underlying designs were determined.

Table 6.5 contains data on the $RB(16, 4, 2)$ designs and their resolutions. The leftmost table contains the full automorphism group orders for each of the 325062 $RB(16, 4, 2)$ designs. Column "$|\operatorname{Aut}(\mathcal{B})|$" in the table indicates the full automorphism group order, and column "Nrd" indicates the number of resolvable designs with that automorphism group order. The table in the middle lists the same information for the 339592 resolutions of $RB(16, 4, 2)$ designs. Finally, the rightmost table gives the number of nonisomorphic resolutions for each resolvable block design. Column "Nr" indicates the number of nonisomorphic resolutions and column "Nrd" gives the number of resolvable designs that have this number of nonisomorphic resolutions.

The unique $RB(16, 4, 2)$ designs with the largest full automorphism group (of order 5760) is formed by taking two copies of the well-known unique affine $RB(16, 4, 1)$ design, and therefore is not very interesting as an example. A more illustrative example is given by the resolvable design with the next largest full automorphism group (of order 1920).

The blocks of the lexicographic maximum representative of this resolvable

Table 6.5: Properties of $RB(16, 4, 2)$ designs.

$\lvert\mathrm{Aut}(\mathcal{B})\rvert$	Nrd
1	314263
2	9588
3	88
4	661
5	3
6	94
8	158
10	7
12	37
16	73
18	2
24	22
32	32
36	2
48	9
64	5
72	1
96	5
120	2
128	1
192	2
256	1
384	2
768	1
1152	1
1920	1
5760	1
Total	325062

$\lvert\mathrm{Aut}(\mathcal{R})\rvert$	Nr
1	325678
2	11977
3	89
4	1112
5	3
6	101
8	315
10	7
12	56
16	110
18	2
24	32
32	61
36	2
48	7
64	7
72	1
96	8
120	2
128	9
192	4
256	1
320	1
384	3
768	1
1152	1
1920	1
5760	1
Total	339592

Nr	Nrd
1	311819
2	12840
3	88
4	170
5	1
6	52
7	4
8	73
9	3
10	2
11	1
13	2
14	2
16	2
24	2
28	1
Total	325062

design are

$$\begin{array}{llllll}
0: & \{0,1,2,3\}, & 1: & \{0,1,4,5\}, & 2: & \{0,2,6,7\}, \\
3: & \{0,3,8,9\}, & 4: & \{0,4,10,11\}, & 5: & \{0,5,12,13\}, \\
6: & \{0,6,10,14\}, & 7: & \{0,7,12,15\}, & 8: & \{0,8,11,15\}, \\
9: & \{0,9,13,14\}, & 10: & \{1,2,10,12\}, & 11: & \{1,3,11,13\}, \\
12: & \{1,4,6,8\}, & 13: & \{1,5,7,9\}, & 14: & \{1,6,10,15\}, \\
15: & \{1,7,12,14\}, & 16: & \{1,8,11,14\}, & 17: & \{1,9,13,15\}, \\
18: & \{2,3,14,15\}, & 19: & \{2,4,6,9\}, & 20: & \{2,4,10,13\}, \\
21: & \{2,5,7,8\}, & 22: & \{2,5,11,12\}, & 23: & \{2,8,13,15\}, \\
24: & \{2,9,11,14\}, & 25: & \{3,4,7,8\}, & 26: & \{3,4,11,12\}, \\
27: & \{3,5,6,9\}, & 28: & \{3,5,10,13\}, & 29: & \{3,6,12,14\}, \\
30: & \{3,7,10,15\}, & 31: & \{4,5,14,15\}, & 32: & \{4,7,13,14\}, \\
33: & \{4,9,12,15\}, & 34: & \{5,6,11,15\}, & 35: & \{5,8,10,14\}, \\
36: & \{6,7,11,13\}, & 37: & \{6,8,12,13\}, & 38: & \{7,9,10,11\}, \\
39: & \{8,9,10,12\}. & & & &
\end{array}$$

The full automorphism group of the design is primitive and has order 1920. A set of generators for the group is given below.

$$\begin{aligned}\mathrm{Aut}(B) = \langle &(0\ 1\ 12)(2\ 7\ 5)(3\ 14\ 13)(4\ 10\ 15)(6\ 9\ 11), \\ &(0\ 14\ 4\ 9\ 5\ 2\ 1\ 11)(3\ 8\ 10\ 13\ 15\ 6\ 7\ 12)\rangle\end{aligned}$$

The group is isomorphic to the group “$\mathsf{ASL}(2,4){:}2$” in the GAP primitive group library, and to group number 16.14 in [18].

The design has 11 nonisomorphic resolutions: (Each column gives the labels of blocks that constitute a parallel class.)

$$\mathcal{R}_1: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 13 & 12 & 17 & 16 & 15 & 14 \\
36 & 37 & 33 & 32 & 23 & 24 & 21 & 19 & 20 & 22 \\
39 & 38 & 35 & 34 & 29 & 30 & 26 & 28 & 27 & 25
\end{array} \qquad
\mathcal{R}_2: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 13 & 12 & 17 & 16 & 15 & 14 \\
36 & 37 & 33 & 32 & 23 & 24 & 22 & 19 & 20 & 21 \\
39 & 38 & 35 & 34 & 29 & 30 & 25 & 28 & 27 & 26
\end{array}$$

$$\mathcal{R}_3: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 13 & 12 & 17 & 16 & 15 & 14 \\
36 & 37 & 33 & 32 & 23 & 24 & 22 & 20 & 19 & 21 \\
39 & 38 & 35 & 34 & 29 & 30 & 25 & 27 & 28 & 26
\end{array} \qquad
\mathcal{R}_4: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 13 & 12 & 17 & 16 & 15 & 14 \\
37 & 36 & 33 & 32 & 23 & 24 & 22 & 20 & 19 & 21 \\
38 & 39 & 35 & 34 & 29 & 30 & 25 & 27 & 28 & 26
\end{array}$$

$$\mathcal{R}_5: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 13 & 14 & 17 & 16 & 15 & 12 \\
36 & 37 & 33 & 32 & 23 & 24 & 21 & 20 & 19 & 22 \\
39 & 38 & 35 & 34 & 29 & 25 & 26 & 27 & 28 & 30
\end{array} \qquad
\mathcal{R}_6: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 13 & 14 & 17 & 16 & 15 & 12 \\
37 & 36 & 33 & 32 & 23 & 24 & 21 & 20 & 19 & 22 \\
38 & 39 & 35 & 34 & 29 & 25 & 26 & 27 & 28 & 30
\end{array}$$

$$\mathcal{R}_7: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 17 & 14 & 13 & 16 & 15 & 12 \\
36 & 37 & 33 & 32 & 21 & 24 & 23 & 20 & 19 & 22 \\
39 & 38 & 35 & 34 & 29 & 25 & 26 & 27 & 28 & 30
\end{array} \qquad
\mathcal{R}_8: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 17 & 14 & 13 & 16 & 15 & 12 \\
37 & 36 & 33 & 32 & 21 & 24 & 23 & 19 & 20 & 22 \\
38 & 39 & 35 & 34 & 29 & 25 & 26 & 28 & 27 & 30
\end{array}$$

$$\mathcal{R}_9: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 10 & 17 & 14 & 13 & 16 & 15 & 12 \\
37 & 36 & 33 & 32 & 21 & 24 & 23 & 20 & 19 & 22 \\
38 & 39 & 35 & 34 & 29 & 25 & 26 & 27 & 28 & 30
\end{array} \qquad
\mathcal{R}_{10}: \begin{array}{cccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
31 & 18 & 11 & 14 & 15 & 16 & 17 & 12 & 13 & 10 \\
37 & 36 & 33 & 22 & 23 & 19 & 21 & 24 & 20 & 25 \\
38 & 39 & 35 & 32 & 27 & 30 & 26 & 28 & 29 & 34
\end{array}$$

$$\mathcal{R}_{11}: \quad \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 31 & 18 & 17 & 14 & 15 & 16 & 11 & 12 & 13 & 10 \\ 37 & 36 & 26 & 22 & 23 & 19 & 21 & 24 & 20 & 25 \\ 38 & 39 & 35 & 32 & 27 & 30 & 33 & 28 & 29 & 34 \end{array}$$

The full automorphism groups of the resolutions are:

$$\begin{aligned}
\mathrm{Aut}(\mathcal{R}_1) = \langle &(0\ 1\ 12)(2\ 7\ 5)(3\ 14\ 13)(4\ 10\ 15)(6\ 9\ 11),\\
&(0\ 14\ 4\ 9\ 5\ 2\ 1\ 11)(3\ 8\ 10\ 13\ 15\ 6\ 7\ 12)\rangle.\\
\mathrm{Aut}(\mathcal{R}_2) = \langle &(0\ 2)(1\ 3)(4\ 15)(5\ 14)(6\ 7)(8\ 10)(9\ 12)(11\ 13),\\
&(1\ 15)(2\ 7)(3\ 12)(4\ 11)(5\ 8)(6)(9\ 13)(10)(14),\\
&(2\ 3)(4\ 5)(6\ 9)(7\ 8)(10\ 13)(11\ 12)\rangle.\\
\mathrm{Aut}(\mathcal{R}_3) = \langle &(0\ 7)(1\ 11)(2\ 6)(3\ 13)(4\ 9)(5\ 10)(8\ 14)(12\ 15),\\
&(2\ 4)(3\ 5)(6\ 10)(7\ 11)(8\ 12)(9\ 13),\\
&(2)(3)(4\ 5)(6\ 7)(8\ 9)(10\ 12)(11\ 13)(14\ 15)\rangle.\\
\mathrm{Aut}(\mathcal{R}_4) = \langle &(0\ 7\ 6)(1\ 8\ 9)(3\ 5\ 4)(10\ 15\ 11)(12\ 13\ 14),\\
&(0\ 11\ 5\ 13\ 1\ 7\ 4\ 6)(2\ 10\ 15\ 12\ 3\ 9\ 14\ 8)\rangle.
\end{aligned}$$

$$\begin{aligned}
\mathrm{Aut}(\mathcal{R}_5) = \langle &(0\ 4)(1\ 5)(2\ 15)(3\ 14)(6\ 12)(7\ 9)(8\ 13)(10\ 11),\\
&(0\ 1)(2\ 3)(4\ 5)(6\ 13)(7\ 11)(8\ 12)(9\ 10)(14\ 15),\\
&(1\ 2)(4\ 6)(5\ 7)(8\ 9)(10)(11\ 14)(12)(13\ 15)\rangle.\\
\mathrm{Aut}(\mathcal{R}_6) = \langle &(0\ 2)(1\ 3)(4\ 15)(5\ 14)(6\ 7)(8\ 10)(9\ 12)(11\ 13),\\
&(0\ 1)(2\ 3)(4\ 5)(6\ 13)(7\ 11)(8\ 12)(9\ 10)(14\ 15),\\
&(1\ 15)(2\ 11)(3\ 8)(4\ 7)(5\ 12)(6\ 10)(13)(14)\rangle.\\
\mathrm{Aut}(\mathcal{R}_7) = \langle &(1\ 3\ 2)(4\ 9\ 7)(5\ 8\ 6)(10\ 13\ 15)(11\ 14\ 12),\\
&(0\ 7\ 14\ 8)(1\ 11\ 15\ 12)(2\ 13\ 5\ 9)(3\ 6\ 4\ 10)\rangle.\\
\mathrm{Aut}(\mathcal{R}_8) = \langle &(0\ 5)(1\ 4)(2\ 14)(3\ 15)(6\ 8)(7\ 10)(9\ 11)(12\ 13),\\
&(0\ 12\ 14\ 11)(1\ 8\ 15\ 7)(2\ 10\ 5\ 6)(3\ 9\ 4\ 13),\\
&(0\ 15)(1\ 14)(2\ 5)(3\ 4)(6)(7\ 11)(8\ 12)(9)(10)(13)\rangle.\\
\mathrm{Aut}(\mathcal{R}_9) = \langle &(2\ 3)(4\ 5)(6\ 9)(7\ 8)(10\ 13)(11\ 12),\\
&(0\ 4)(1\ 5)(2\ 15)(3\ 14)(6\ 12)(7\ 9)(8\ 13)(10\ 11),\\
&(0\ 7)(1\ 11)(2\ 6)(3\ 13)(4\ 9)(5\ 10)(8\ 14)(12\ 15)\rangle.\\
\mathrm{Aut}(\mathcal{R}_{10}) = \langle &(1\ 11\ 12\ 9)(2\ 10\ 7\ 14)(3\ 4\ 15\ 13)(5\ 8),\\
&(0\ 1)(2\ 3)(4\ 5)(6\ 13)(7\ 11)(8\ 12)(9\ 10)(14\ 15)\rangle.\\
\mathrm{Aut}(\mathcal{R}_{11}) = \langle &(0\ 4\ 2\ 8)(1\ 6\ 7\ 3)(5\ 9)(10\ 13\ 15\ 11)(12)(14),\\
&(0\ 10\ 11\ 9\ 14)(1\ 3\ 5\ 15\ 12)(2\ 13\ 6\ 4\ 7)\rangle.
\end{aligned}$$

All of the full automorphism groups are transitive. Moreover, groups $\mathrm{Aut}(\mathcal{R}_1)$ and $\mathrm{Aut}(\mathcal{R}_{11})$ are primitive. Group $\mathrm{Aut}(\mathcal{R}_1)$ coincides with the full automorphism group of the design; group $\mathrm{Aut}(\mathcal{R}_{11})$ is isomorphic to group "`(2^4:5).4`" in the `GAP` primitive group library, which is group number 16.5

in [18]. The orders of the groups are

$$\begin{array}{lll}
\mathrm{Aut}(\mathcal{R}_1) = 1920, & \mathrm{Aut}(\mathcal{R}_2) = 128, & \mathrm{Aut}(\mathcal{R}_3) = 128, \\
\mathrm{Aut}(\mathcal{R}_4) = 384, & \mathrm{Aut}(\mathcal{R}_5) = 32, & \mathrm{Aut}(\mathcal{R}_6) = 32, \\
\mathrm{Aut}(\mathcal{R}_7) = 96, & \mathrm{Aut}(\mathcal{R}_8) = 32, & \mathrm{Aut}(\mathcal{R}_9) = 32, \\
\mathrm{Aut}(\mathcal{R}_{10}) = 64, & \mathrm{Aut}(\mathcal{R}_{11}) = 320. &
\end{array}$$

6.5 CLASSIFICATION OF $RB(24, 12, 11)$ DESIGNS

The affine $RB(24, 12, 11)$ designs have been studied in [56] by Ito, Leon, and Longyear. They determined 129 as the number of nonisomorphic $RB(24, 12, 11)$ designs, and obtained from these 59 nonisomorphic Hadamard matrices of order 24. However, it was later discovered by Kimura [64] that the actual number of nonisomorphic Hadamard matrices of order 24 was 60. (This has been independently confirmed by Spence [111].) Because the completeness of the classification of Hadamard matrices of order 24 in [56] was based on the assumption that the 129 $RB(24, 12, 11)$ designs form a complete classification, this naturally raises the question whether there exist additional $RB(24, 12, 11)$ designs.

To determine all the nonisomorphic $RB(24, 12, 11)$ designs, we constructed the resolutions of the designs using the Read–Faradžev-type algorithm from Section 5.2 with $p = 14$. The algorithm found 130 nonisomorphic resolutions, to which correspond 130 nonisomorphic $RB(24, 12, 11)$ designs by Theorem 2.3.19.

Based on the automorphism group sizes and other tabulated isomorphism invariants in [56], we suspect that the design/resolution missed in [56] was

$$\begin{aligned}
&\{\{0,1,2,3,4,5,6,7,8,9,10,11\},\{12,13,14,15,16,17,18,19,20,21,22,23\}\},\\
&\{\{0,1,2,3,4,5,12,13,14,15,16,17\},\{6,7,8,9,10,11,18,19,20,21,22,23\}\},\\
&\{\{0,1,2,3,6,7,12,13,18,19,20,21\},\{4,5,8,9,10,11,14,15,16,17,22,23\}\},\\
&\{\{0,1,2,3,8,9,14,15,18,19,22,23\},\{4,5,6,7,10,11,12,13,16,17,20,21\}\},\\
&\{\{0,1,2,3,10,11,16,17,20,21,22,23\},\{4,5,6,7,8,9,12,13,14,15,18,19\}\},\\
&\{\{0,1,4,5,6,7,14,16,18,20,22,23\},\{2,3,8,9,10,11,12,13,15,17,19,21\}\},\\
&\{\{0,1,4,5,8,10,12,17,18,19,21,22\},\{2,3,6,7,9,11,13,14,15,16,20,23\}\},\\
&\{\{0,1,4,6,8,9,13,15,17,20,21,23\},\{2,3,5,7,10,11,12,14,16,18,19,22\}\},\\
&\{\{0,1,5,7,10,11,13,14,15,19,21,23\},\{2,3,4,6,8,9,12,16,17,18,20,22\}\},\\
&\{\{0,1,6,7,9,11,12,15,16,17,19,22\},\{2,3,4,5,8,10,13,14,18,20,21,23\}\},\\
&\{\{0,1,8,9,10,11,12,13,14,16,18,20\},\{2,3,4,5,6,7,15,17,19,21,22,23\}\},\\
&\{\{0,2,4,5,9,11,12,13,19,20,22,23\},\{1,3,6,7,8,10,14,15,16,17,18,21\}\},\\
&\{\{0,2,4,6,10,11,14,15,17,18,19,20\},\{1,3,5,7,8,9,12,13,16,21,22,23\}\},\\
&\{\{0,2,4,7,9,10,12,15,16,18,21,23\},\{1,3,5,6,8,11,13,14,17,19,20,22\}\},\\
&\{\{0,2,5,6,8,11,13,15,16,18,21,22\},\{1,3,4,7,9,10,12,14,17,19,20,23\}\},\\
&\{\{0,2,5,7,8,9,14,16,17,19,20,21\},\{1,3,4,6,10,11,12,13,15,18,22,23\}\},
\end{aligned}$$

$$\{\{0,2,6,7,8,10,12,13,14,17,22,23\},\{1,3,4,5,9,11,15,16,18,19,20,21\}\},$$
$$\{\{0,3,4,6,8,11,12,14,16,19,21,23\},\{1,2,5,7,9,10,13,15,17,18,20,22\}\},$$
$$\{\{0,3,4,7,8,10,13,15,16,19,20,22\},\{1,2,5,6,9,11,12,14,17,18,21,23\}\},$$
$$\{\{0,3,4,7,9,11,13,14,17,18,21,22\},\{1,2,5,6,8,10,12,15,16,19,20,23\}\},$$
$$\{\{0,3,5,6,9,10,12,14,15,20,21,22\},\{1,2,4,7,8,11,13,16,17,18,19,23\}\},$$
$$\{\{0,3,5,6,9,10,13,16,17,18,19,23\},\{1,2,4,7,8,11,12,14,15,20,21,22\}\},$$
$$\{\{0,3,5,7,8,11,12,15,17,18,20,23\},\{1,2,4,6,9,10,13,14,16,19,21,22\}\}.$$

The following properties distinguish the design above from the 129 designs tabulated in [56]. First, the full automorphism group

$$\begin{aligned}\mathrm{Aut}(\mathcal{R}) = \langle &(0\ 1)(2\ 3)(4\ 9\ 7\ 10)(5\ 8\ 6\ 11)(12\ 23)(13\ 22)\cdots\\ &\cdots(14\ 18\ 20\ 16)(15\ 19\ 21\ 17),\\ &(0\ 2)(1\ 3)(4\ 5)(6\ 7)(8\ 10)(9\ 11)(12\ 13)(14\ 17)\cdots\\ &\cdots(15\ 16)(18\ 21)(19\ 20)(22\ 23)\rangle\end{aligned}$$

of the design has order 8. Second, the induced action of the automorphism group on the set of parallel classes has 9 orbits: four of cardinality 4, two of cardinality 2, and three of cardinality 1. Finally, the design has 12 missing quadliterals, that is, 4-subsets of $\mathbb{Z}_{24}$ that do not occur in any block.

6.6 CLASSIFICATION OF $RB(14,7,12)$ DESIGNS

The $RB(14,7,12)$ designs were classified using the Read–Faradžev algorithm for generation of resolutions of block designs with $p = 8$. The total number of resolutions obtained was 1363486, of which 1360800 have a trivial full automorphism group.

Each resolution is unique and the automorphism groups of the resolution and the underlying design agree by Theorem 2.3.15. Table 6.6 gives the orders of the automorphism groups of the $RB(14,7,12)$ designs/resolutions.

Table 6.6: Properties of $RB(14,7,12)$ designs.

$\lvert\mathrm{Aut}(\mathcal{B})\rvert$	Nrd
1	1360800
2	1819
3	748
4	63
6	37
8	1
12	13
13	1
24	1
39	2
156	1
Total	1363486

The lexicographic maximum representative of the $RB(14,7,2)$ design with the largest automorphism group is given below.

$$\begin{aligned}\mathcal{R} = \{&\{\{0,1,2,3,4,5,6\},\{7,8,9,10,11,12,13\}\},\\
&\{\{0,1,2,3,4,5,7\},\{6,8,9,10,11,12,13\}\},\\
&\{\{0,1,2,3,8,9,10\},\{4,5,6,7,11,12,13\}\},\\
&\{\{0,1,2,4,8,11,12\},\{3,5,6,7,9,10,13\}\},\\
&\{\{0,1,2,5,9,11,13\},\{3,4,6,7,8,10,12\}\},\\
&\{\{0,1,3,6,7,12,13\},\{2,4,5,8,9,10,11\}\},\\
&\{\{0,1,3,6,9,12,13\},\{2,4,5,7,8,10,11\}\},\\
&\{\{0,1,4,6,9,10,11\},\{2,3,5,7,8,12,13\}\},\\
&\{\{0,1,4,7,8,11,12\},\{2,3,5,6,9,10,13\}\},\\
&\{\{0,1,5,7,8,10,13\},\{2,3,4,6,9,11,12\}\},\\
&\{\{0,1,5,8,10,12,13\},\{2,3,4,6,7,9,11\}\},\\
&\{\{0,1,6,7,9,10,11\},\{2,3,4,5,8,12,13\}\},\\
&\{\{0,2,3,6,8,11,13\},\{1,4,5,7,9,10,12\}\},\\
&\{\{0,2,3,7,8,9,10\},\{1,4,5,6,11,12,13\}\},\\
&\{\{0,2,4,7,9,12,13\},\{1,3,5,6,8,10,11\}\},\\
&\{\{0,2,4,9,10,12,13\},\{1,3,5,6,7,8,11\}\},\\
&\{\{0,2,5,6,7,10,12\},\{1,3,4,8,9,11,13\}\},\\
&\{\{0,2,5,6,10,11,12\},\{1,3,4,7,8,9,13\}\},\\
&\{\{0,2,6,7,8,11,13\},\{1,3,4,5,9,10,12\}\},\\
&\{\{0,3,4,5,10,11,13\},\{1,2,6,7,8,9,12\}\},\\
&\{\{0,3,4,6,8,10,12\},\{1,2,5,7,9,11,13\}\},\\
&\{\{0,3,4,7,10,11,13\},\{1,2,5,6,8,9,12\}\},\\
&\{\{0,3,5,7,9,11,12\},\{1,2,4,6,8,10,13\}\},\\
&\{\{0,3,5,8,9,11,12\},\{1,2,4,6,7,10,13\}\},\\
&\{\{0,4,5,6,7,8,9\},\{1,2,3,10,11,12,13\}\},\\
&\{\{0,4,5,6,8,9,13\},\{1,2,3,7,10,11,12\}\}\}.\end{aligned}$$

The full automorphism group

$$\mathrm{Aut}(\mathcal{R}) = \langle (1\ 3\ 2\ 8\ 11\ 12\ 5\ 10\ 13\ 4\ 9\ 6),\ (0\ 3\ 13\ 12\ 8\ 11\ 1\ 5\ 9\ 4\ 6\ 2\ 10)\rangle$$

has order 156 and is isomorphic to the group $\mathrm{AGL}_1(13)$.

7 CONCLUSION

The nonexistence of RB(15,5,4) designs and the other classification results obtained in this report provide additional evidence that exhaustive search is a valuable tool for settling problems in design theory — the time efficiency of the search algorithm and the available computational resources being the limiting factors for such an approach.

An additional concern in the exhaustive approach is naturally the reliability of the results obtained, especially in the case of negative results. In particular, a result obtained using exhaustive search is correct if the search algorithm is correct and the implementation behaves correctly. The correct behaviour of the implementation depends in turn on the correctness of the source code-level implementation, the correctness of the compiler, the error-free operation of the hardware, and so on. Given such a multitude of possible sources of error, it is obvious that an independent verification is desirable for complete confidence in a result based on exhaustive search, even if the algorithm implementation correctly reproduces earlier research results. In this sense computer search for mathematical results should be regarded as an experimental science, where the experimental setup must be described to a sufficient degree to allow reproducibility, and where the results are generally considered valid only after at least one independent verification.

There are two central issues to consider in designing effective exhaustive generation algorithms for block designs and other combinatorial configurations: (i) how to generate the configurations efficiently; and, if elimination of isomorphic configurations is desired, (ii) how to detect isomorphic configurations. In this report the approach chosen to generate block designs was to formulate the generation problem in a group-theoretic framework and to use backtrack search combined with detection and elimination of isomorphic partial solutions to generate the desired orbit transversal. It is natural to ask whether the backtrack search approach—which is usually considered as a "last resort" option when other algorithmic disciplines fail—is the best available for solving such problems. Although we cannot give an affirmative answer to this question, we will attempt to argue that the generation of block designs is by no means an easy combinatorial problem, so the use of backtrack search is acceptable.

To put generation of block designs into perspective, we briefly discuss generation of other types of configurations. For some configurations such as permutations, set partitions, and combinations, very fast generation methods exist [34]. Generation up to isomorphism is possible with constant delay for trees [121] and with polynomial delay (in the number of vertices) for graphs [43, 45]. As far as we are aware, there is no published polynomial delay algorithm for generating up to isomorphism all set systems with a point set and cardinality fixed. It is our belief, however, that the easy–hard interleaving technique used in [43] to achieve polynomial delay could in some form be used to achieve polynomial delay for generation of set systems.

All of the generation problems in the previous paragraph can be regarded as having an efficient solution. What is more often required is the efficient generation of a subcollection of these configurations that satisfy an additional property the generation of, say, regular graphs [87] or block designs being good examples. Deciding whether a given problem instance has a particular property is a standard problem studied in computational complexity theory, so it is natural to attempt to investigate the complexity of generation problems by studying related decision problems. Such an attempt is made in Appendix C, where it is hoped that the reader will become at least partially convinced that generation of block designs and resolutions are both hard problems from a computational complexity viewpoint.

If we regard backtrack search as an acceptable method for solving the generation problem for block designs, then further effort should be directed to improve its performance. The collection of principles for designing fast backtrack algorithms presented in [80] are helpful in this regard. The first question to ask when designing any backtrack search algorithm is what constitutes a partial solution, and what can be done to facilitate an early and efficient detection of inextendible partial solutions to limit the number of partial solutions that must be traversed by the algorithm.

The constraints for a block design necessarily involve a consideration of the points in the blocks, so it is natural to proceed either point by point or block by block in a backtrack search. For the generation of block designs we chose to proceed point by point motivated by the success of similar algorithms in the past. Furthermore, the pairwise inner product constraint required for the rows of an incidence system could be formulated as a clique problem on the compatibility graph; the clique approach presents a performance advantage over standard backtrack search because it is not necessary to backtrack through the set of rows that extend a partial solution after every extension step. The block by block generation approach for block designs was discouraged by an **NP**-completeness result of Colbourn [19] (see Appendix C). For resolutions of block designs no such result exists as far as we are aware, so we chose to explore generating the codes that correspond to the resolutions both codeword by codeword and coordinate by coordinate.

Of the three algorithms described in Chapter 5, both of the Read–Faradžev-type algorithms were successful in producing new classification results. The McKay-type algorithm, however, proved to be too slow to produce any new results. The most likely reason why the algorithm is ineffective is the choice to construct codes coordinate by coordinate. (The main reason for this choice was to try out an alternative method for classification of codes of small length but large cardinality. For example, the $(10, 21, 9)_7$ ED_m-codes that correspond to the resolutions of Kirkman triple systems of order 21 are of this type.) Although the number of extension steps in the search is smaller when proceeding coordinate by coordinate, the number of inextendible solutions that need to be traversed by the McKay-type algorithm seems to be much larger than when proceeding codeword by codeword. There are at least two likely sources for this inefficiency, both of which increase the number of inextendible partial solutions that must be traversed by the algorithm. First, the necessary conditions for extendibility given in Lemma 5.3.2 are likely not to constrain the partial solutions as much as the conditions in Lemmata 5.2.1 and 5.2.2 do. Second, there seems to be no immediate counterpart to Lemmata 5.2.6 and 5.2.4 that would constrain the order in which coordinates are added to a partial solution in the McKay framework.

We conclude this discussion by indicating several topics for future research. First of all, the new classification results described Chapter 6 should be more thoroughly analyzed. This involves implementing a back-end to the generation algorithm that analyzes the generated designs for properties of interest and stores either part or all of the designs for subsequent use. (A preliminary version of such a back-end was implemented for classification of the $RB(16, 4, 2)$ designs as described in Section 6.4.)

Improving the efficiency of the Read–Faradžev-type backtrack search algorithms is also a topic of future research. In particular, the packing constraint from [29] and the strong partial isomorph rejection technique from [30] should improve the efficiency of the row and codeword generation procedures. Additionally, it is our belief that more advanced computational group theory methods (see, for example, [14]) could be used to improve the performance of the canonicity test procedures. In the least, the canonicity test procedures should take advantage of the automorphism groups computed earlier in the search. The clique search algorithm can also be improved to take into account the automorphism group of the partial solution, which induces a group of automorphisms for the compatibility graph (cf. [112]). Indeed, clique search in a graph with a known group of automorphisms is a research topic of independent interest.

On the theoretical side, Appendix C contains problems whose exact computational complexity has not been established as far as we are aware. Two examples are deciding resolvability of a given block design and deciding whether a given incidence system can be completed to a block design by addition of points.

Finally, we wish to mention open problems in design theory which could be accessible to exhaustive methods in the future, although the algorithms developed in this report are still too inefficient to settle them. Namely, the $B(22, 8, 4)$ designs constitute the smallest parameter family of block designs whose existence is at present unsettled. Furthermore, the families of $STS(19)$ and $KTS(21)$ await classification.

ACKNOWLEDGMENT

The research was supported by the Academy of Finland under grant 44517.

Bibliography

[1] W. O. Alltop, Extending t-designs, *J. Combin. Theory, Ser. A* **18** (1975), 177–186.

[2] M. Aschbacher. *Finite Group Theory,* 2nd edition. Cambridge Stud. Adv. Math. no. 10. Cambridge University Press, Cambridge, England, 2000.

[3] L. Babai and E. M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing,* pages 171–183. ACM, New York, 1983.

[4] T. Beth, D. Jungnickel, and H. Lenz. *Design Theory.* Cambridge University Press, Cambridge, England, 1993. Originally published by Bibliographisches Institut, Zürich, 1985.

[5] A. Betten, R. Laue, and A. Wassermann, Simple 7-designs with small parameters, *J. Combin. Des.* **7** (1999) no. 2, 79–94.

[6] A. Blass and Y. Gurevich, Equivalence relations, invariants, and normal forms, *SIAM J. Comput.* **13** (1984) no. 4, 682–689.

[7] G. T. Bogdanova, A. E. Brouwer, S. N. Kapralov, and P. R. J. Östergård, Error-correcting codes over an alphabet of four elements, *Des. Codes Cryptogr.* **23** (2001) no. 3, 333–342.

[8] R. C. Bose, A note on the resolvability of balanced incomplete block designs, *Sankhyā* **6** (1942), 105–110.

[9] R. C. Bose, Strongly regular graphs, partial geometries and partially balanced designs, *Pacific J. Math.* **13** (1963), 389–419.

[10] S. Brandt, G. Brinkmann, and T. Harmuth, The generation of maximal triangle-free graphs, *Graphs Combin.* **16** (2000) no. 2, 149–157.

[11] G. Brinkmann, Fast generation of cubic graphs, *J. Graph Theory* **23** (1996) no. 2, 139–149.

[12] G. Brinkmann. Isomorphism rejection in structure generation programs. In *Discrete Mathematical Chemistry* (P. Hansen, P. Fowler, and M. Zheng, eds.), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. no. 51, pages 25–38. Amer. Math. Soc., Providence, Rhode Island, 2000.

[13] A. E. Brouwer, J. B. Shearer, N. J. A. Sloane, and W. D. Smith, A new table of constant weight codes, *IEEE Trans. Inform. Theory* **36** (1990) no. 6, 1334–1380.

[14] G. Butler. *Fundamental Algorithms for Permutation Groups.* Lecture Notes in Computer Science no. 559. Springer-Verlag, Berlin-New York, 1991.

[15] P. J. Cameron. *Permutation Groups.* London Math. Soc. Stud. Texts no. 45. Cambridge University Press, Cambridge, England, 1999.

[16] P. J. Cameron and J. H. van Lint. *Designs, Graphs, Codes and their Links.* London Math. Soc. Stud. Texts no. 22. Cambridge University Press, Cambridge, England, 1991.

[17] R. Carraghan and P. M. Pardalos, An exact algorithm for the maximum clique problem, *Oper. Res. Lett.* **9** (1990) no. 6, 375–382.

[18] L. G. Chouinard II, R. Jajcay, and S. S. Magliveras. Finite groups and designs. In *The CRC Handbook of Combinatorial Designs* (C. J. Colbourn and J. H. Dinitz, eds.), pages 587–615. CRC Press, Boca Raton, Florida, 1996.

[19] C. J. Colbourn, Embedding partial Steiner triple systems is NP-complete, *J. Combin. Theory, Ser. A* **35** (1983) no. 1, 100–105.

[20] C. J. Colbourn and J. H. Dinitz, eds. *The CRC Handbook of Combinatorial Designs.* CRC Press, Boca Raton, Florida, 1996.

[21] C. J. Colbourn and J. H. Dinitz. Applications of combinatorial designs to communications, cryptography, and networking. In *Surveys in Combinatorics* (J. D. Lamb and D. A. Preece, eds.), London Math. Soc. Lecture Note Ser. no. 267, pages 37–100. Cambridge University Press, Cambridge, England, 1999.

[22] C. J. Colbourn and A. Rosa. *Triple Systems.* Oxford Mathematical Monographs. Clarendon Press, Oxford, England, 1999.

[23] C. J. Colbourn and P. C. van Oorschot, Applications of combinatorial designs in computer science, *ACM Computing Surv.* **21** (1989) no. 2, 223–249.

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

[25] M. J. Colbourn and C. J. Colbourn, Concerning the complexity of deciding isomorphism of block designs, *Discrete Appl. Math.* **3** (1981) no. 3, 155–162.

[26] I. Constantinescu and W. Heise, On the concept of code-isomorphy, *J. Geom.* **57** (1996) no. 1–2, 63–69.

[27] N. G. de Bruijn, Generalization of Pólya's fundamental theorem in enumerative combinatorial analysis, *Indag. Math.* **21** (1959), 59–69.

[28] N. G. de Bruijn, A survey of generalizations of Pólya's enumeration theorem, *Nieuw Arch. Wisk. (3)* **19** (1971), 89–112.

[29] P. C. Denny. Search and enumeration techniques for incidence structures. Master's thesis, University of Auckland, 1998. Published as CDMTCS Technical Report #85, available at ⟨URL: http://www.cs.auckland.ac.nz/CDMTCS/⟩.

[30] P. C. Denny and P. B. Gibbons, Case studies and new results in combinatorial enumeration, *J. Combin. Des.* **8** (2000) no. 4, 239–260.

[31] P. C. Denny and R. Mathon, A census of t-$(t+8, t+2, 4)$ designs, $2 \leq t \leq 4$, *J. Statist. Plann. Inference*, to appear.

[32] J. H. Dinitz, D. K. Garnick, and B. D. McKay, There are 526,915,620 nonisomorphic one-factorizations of K_{12}, *J. Combin. Des.* **2** (1994) no. 4, 273–285.

[33] J. D. Dixon and B. Mortimer. *Permutation Groups.* Graduate Texts in Mathematics no. 163. Springer-Verlag, Berlin-New York, 1996.

[34] G. Ehrlich, Loopless algorithms for generating permutations, combinations, and other combinatorial configurations, *J. ACM* **20** (1973) no. 3, 500–513.

[35] I. A. Faradžev. Constructive enumeration of combinatorial objects. In *Problèmes Combinatoires et Théorie des Graphes Colloque Internat. CNRS No. 260*, pages 131–135. CNRS, Paris, 1978.

[36] G. H. Fricke, S. T. Hedetniemi, and D. P. Jacobs, Independence and irredundance in k-regular graphs, *Ars. Combin.* **49** (1998), 271–279.

[37] S. Furino, Y. Miao, and J. Yin. *Frames and Resolvable Designs. Uses, Constructions, and Existence.* CRC Press, Boca Raton, Florida, 1996.

[38] M. Furst, J. Hopcroft, and E. M. Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science*, pages 36–41. IEEE Computer Society Press, Los Alamitos, California, 1980.

[39] The GAP Group, Aachen, St Andrews. *GAP – Groups, Algorithms, and Programming, Version 4.2*, 2000. Available electronically at ⟨URL: http://www-gap.dcs.st-and.ac.uk/~gap⟩.

[40] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to The Theory of NP-Completeness.* W.H. Freeman and Co., San Francisco, California, 1979.

[41] P. B. Gibbons. *Computing Techniques for the Construction and Analysis of Block Designs.* PhD thesis, University of Toronto, 1976.

[42] P. B. Gibbons. Computational methods in design theory. In *The CRC Handbook of Combinatorial Designs* (C. J. Colbourn and J. H. Dinitz, eds.), pages 718–740. CRC Press, Boca Raton, Florida, 1996.

[43] L. A. Goldberg, Efficient algorithms for listing unlabeled graphs, *J. Algorithms* **13** (1992), 128–143.

[44] L. A. Goldberg, Automating Pólya theory: The computational complexity of evaluating the cycle index polynomial, *Inform. and Comput.* **105** (1993) no. 2, 268–288.

[45] L. A. Goldberg. *Efficient Algorithms for Listing Combinatorial Structures.* Cambridge University Press, Cambridge, England, 1993.

[46] L. A. Goldberg. Computation in permutation groups: counting and randomly sampling orbits. In *Surveys in Combinatorics* (J. W. P. Hirschfeld, ed.), London Math. Soc. Lecture Note Ser. no. 288, pages 109–143. Cambridge University Press, Cambridge, England, 2001.

[47] L. A. Goldberg and M. Jerrum, Counting unlabelled subtrees of a tree is #P-complete, *LMS J. Comput. Math.* **3** (2000), 117–124.

[48] S. W. Golomb and L. D. Baumert, Backtrack programming, *J. ACM* **12** (1965) no. 4, 516–524.

[49] W. H. Greub. *Linear Algebra*, 3rd edition. Die Grundlehren der Mathematischen Wissenschaften no. 97. Springer-Verlag, Berlin-New York, 1967.

[50] T. Grüner, R. Laue, and M. Meringer. Algorithms for group actions applied to graph generation. In *Groups and Computation, II* (L. Finkelstein and W. M. Kantor, eds.), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. no. 28, pages 113–122. Amer. Math. Soc., Providence, Rhode Island, 1997.

[51] F. Harary and E. M. Palmer. *Graphical Enumeration.* Academic Press, New York-London, 1973.

[52] C. Y. Ho, Hamming spaces and maximal self dual codes over GF(q), q = odd, *Bol. Soc. Brasil. Mat.* **15** (1984) no. 1, 7–24.

[53] D. G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. A. Rodger, and J. R. Wall. *Coding Theory: The Essentials.* Monogr. Textbooks Pure Appl. Math. no. 150. Marcel Dekker, New York, 1991.

[54] D. R. Hughes and F. C. Piper. *Design Theory.* Cambridge University Press, Cambridge, England, 1985.

[55] J. F. Humphreys. *A Course in Group Theory.* Oxford University Press, Oxford, England, 1996.

[56] N. Ito, J. S. Leon, and J. Q. Longyear, Classification of $3-(24,12,5)$ designs and 24-dimensional Hadamard matrices, *J. Combin. Theory, Ser. A* **31** (1981) no. 1, 66–93.

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

[58] M. Jerrum, A compact representation for permutation groups, *J. Algorithms* 7 (1986) no. 1, 60–78.

[59] M. Jerrum. Computational Pólya theory. In *Surveys in Combinatorics* (P. Rowlinson, ed.), London Math. Soc. Lecture Note Ser. no. 218, pages 103–118. Cambridge University Press, Cambridge, England, 1995.

[60] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou, On generating all maximal independent sets, *Inform. Process. Lett.* **27** (1988) no. 3, 119–123.

[61] P. Kaski and P. R. J. Östergård, There exists no (15,5,4) RBIBD, *J. Combin. Des.* **9** (2001) no. 3, 227–232. Reprinted in correct form in [*J. Combin. Des.* **9** (2001) no. 5, 357–362].

[62] A. Kerber. *Applied Finite Group Actions*, 2nd edition. Algorithms Combin. no. 19. Springer-Verlag, Berlin-New York, 1999.

[63] A. Kerber and R. Laue, Group actions, double cosets, and homomorphisms: unifying concepts for the constructive theory of discrete structures, *Acta Appl. Math.* **52** (1998) no. 1–3, 63–90.

[64] H. Kimura, New Hadamard matrix of order 24, *Graphs Combin.* **5** (1989) no. 3, 235–242.

[65] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, Boston, Massachusetts, 1993.

[66] W. Kocay and G. H. J. van Rees, Some non-isomorphic $(4t+4, 8t+6, 4t+3, 2t+2, 2t+1)$-BIBD's, *Discrete Math.* **92** (1991) no. 1–3, 159–172.

[67] E. S. Kramer and D. M. Mesner, t-Designs on hypergraphs, *Discrete Math.* **15** (1976) no. 3, 263–296.

[68] D. L. Kreher and S. P. Radziszowski, Finding simple t-designs by using basis reduction, *Congr. Numer.* **55** (1986), 235–244.

[69] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration and Search*. CRC Press, Boca Raton, Florida, 1999.

[70] C. Lam and V. D. Tonchev, Classification of affine resolvable 2-$(27, 9, 4)$ designs, *J. Statist. Plann. Inference* **56** (1996) no. 2, 187–202. Corrigendum appears in [*J. Statist. Plann. Inference* **86** (2000) no. 1, 277–278].

[71] C. W. H. Lam, L. Thiel, and S. Swiercz, The nonexistence of finite projective planes of order 10, *Canad. J. Math.* **41** (1989) no. 6, 1117–1123.

[72] R. Laue, Construction of combinatorial objects – a tutorial, *Bayreuth. Math. Schr.* **43** (1993), 53–96.

[73] R. Laue. Constructing objects up to isomorphism, simple 9-designs with small parameters. In *Algebraic Combinatorics and Applications (Proc. ALCOMA, Sept. 12–19, Gößweinstein, Germany, 1999)* (A. Betten, A. Kohnert, R. Laue, and A. Wassermann, eds.), pages 232–260. Springer-Verlag, Berlin-Heidelberg, 2001.

[74] E. M. Luks. Permutation groups and polynomial-time computation. In *Groups and Computation* (L. Finkelstein and W. M. Kantor, eds.), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. no. 11, pages 139–175. Amer. Math. Soc., Providence, Rhode Island, 1993.

[75] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland Math. Library no. 16. North-Holland, New York, 1977.

[76] R. Mathon, A note on the graph isomorphism counting problem, *Inform. Process. Lett.* **8** (1979) no. 3, 131–132.

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

[78] R. Mathon and A. Rosa. 2-(v, k, λ) designs of small order. In *The CRC Handbook of Combinatorial Designs* (C. J. Colbourn and J. H. Dinitz, eds.), pages 3–41. CRC Press, Boca Raton, Florida, 1996.

[79] R. A. Mathon, K. T. Phelps, and A. Rosa, Small Steiner triple systems and their properties, *Ars. Combin.* **15** (1983), 3–110. Errata appears in [*Ars. Combin.* **16** (1983), 286].

[80] B. McKay, W. Myrvold, and J. Nadon. Fast backtracking principles applied to find new cages. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 188–191. ACM, New York, 1998.

[81] B. D. McKay, Practical graph isomorphism, *Congr. Numer.* **30** (1981), 45–87.

[82] B. D. McKay. *nauty* User's Guide (version 1.5). Tech. Rep. TR-CS-90-02, Computer Science Department, Australian National University, 1990.

[83] B. D. McKay. autoson – a distributed batch system for UNIX workstation networks (version 1.3). Technical report TR-CS-96-03, Computer Science Department, Australian National University, 1996.

[84] B. D. McKay, Isomorph-free exhaustive generation, *J. Algorithms* **26** (1998) no. 2, 306–324.

[85] B. D. McKay and S. P. Radziszowski. The nonexistence of 4-(12,6,6) designs. In *Computational and Constructive Design Theory* (W. Wallis, ed.), Math. Appl. no. 368, pages 177–188. Kluwer, Dordrecht, The Netherlands, 1996.

[86] M. Meringer. Erzeugung regulärer Graphen. Master's thesis, Universität Bayreuth, 1996. (in German).

[87] M. Meringer, Fast generation of regular graphs and construction of cages, *J. Graph Theory* **30** (1999) no. 2, 137–146.

[88] G. L. Miller. On the $n^{\log n}$ isomorphism technique. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 51–57. ACM, New York, 1978.

[89] T. Miyazaki. The complexity of McKay's canonical labeling algorithm. In *Groups and Computation, II* (L. Finkelstein and W. M. Kantor, eds.), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. no. 28, pages 239–256. Amer. Math. Soc., Providence, Rhode Island, 1997.

[90] L. B. Morales and C. Velarde, A complete classification of (12,4,3)-RBIBDs, *J. Combin. Des.* **9** (2001) no. 6, 385–400.

[91] P. M. Neumann, A lemma that is not Burnside's, *Math. Sci.* **4** (1979) no. 2, 133–141.

[92] P. R. J. Östergård, Classification of binary/ternary one-error-correcting codes, *Discrete Math.* **223** (2000) no. 1–3, 253–262.

[93] P. R. J. Östergård, Enumeration of 2-$(12, 3, 2)$ designs, *Australas. J. Combin.* **22** (2000), 227–231.

[94] P. R. J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Appl. Math.* **120** (2002), 195–205.

[95] P. R. J. Östergård, T. Baicheva, and E. Kolev, Optimal binary one-error-correcting codes of length 10 have 72 codewords, *IEEE Trans. Inform. Theory* **45** (1999) no. 4, 1229–1231.

[96] P. R. J. Östergård and P. Kaski, Enumeration of 2-$(9, 3, \lambda)$ designs and their resolutions, *Des. Codes Cryptogr.*, to appear.

[97] C. H. Papadimitriou, On the complexity of integer programming, *J. ACM* **28** (1981) no. 4, 765–768.

[98] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[99] C. Pietsch. *Über die Enumeration von Inzidenzstrukturen*. PhD thesis, Universität Rostock, 1993. (in German).

[100] G. Pólya, Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und Chemische Verbindungen, *Acta Math.* **68** (1937), 145–253.

[101] D. Raghavarao. *Constructions and Combinatorial Problems in Design of Experiments*. Wiley Ser. Probab. Math. Statist. Wiley, New York, 1971.

[102] D. K. Ray-Chaudhuri and R. M. Wilson. Solution of Kirkman's schoolgirl problem. In *Combinatorics (Proc. Sympos. Pure Math., Vol. XIX, Univ. California, Los Angeles, California, 1968)* (T. S. Motzkin, ed.), pages 187–203. Amer. Math. Soc., Providence, Rhode Island, 1971.

[103] R. C. Read, Every one a winner *or* How to avoid isomorphism search when cataloguing combinatorial configurations, *Ann. Discrete Math.* **2** (1978), 107–120.

[104] R. C. Read and D. G. Corneil, The graph isomorphism disease, *J. Graph Theory* **1** (1977) no. 4, 339–363.

[105] J. J. Rotman. *An Introduction to the Theory of Groups*, 4th edition. Graduate Texts in Mathematics no. 148. Springer-Verlag, New York, 1995.

[106] G. F. Royle, An orderly algorithm and some applications in finite geometry, *Discrete Math.* **185** (1998) no. 1–3, 105–115.

[107] B. Schmalz, Verwendung von Untergruppenleitern zur Bestimmung von Doppelnebenklassen, *Bayreuth. Math. Schr.* **31** (1990), 109–143. (in German).

[108] B. Schmalz, The t-designs with prescribed automorphism group, new simple 6-designs, *J. Combin. Des.* **1** (1993) no. 2, 125–170.

[109] N. V. Semakov and V. A. Zinov'ev, Equidistant q-ary codes with maximal distance and resolvable balanced incomplete block designs, *Problemy Peredačhi Informacii* **4** (1968) no. 2, 3–10. Translated from Russian in [*Problems of Information Transmission* **4** (1968) no. 2, 1–7].

[110] E. Spence, A complete classification of symmetric (31,10,3) designs, *Des. Codes Cryptogr.* **2** (1992) no. 2, 127–136.

[111] E. Spence, Classification of Hadamard matrices of order 24 and 28, *Discrete Math.* **140** (1995) no. 1–3, 185–243.

[112] E. Spence, The complete classification of Steiner systems $S(2, 4, 25)$, *J. Combin. Des.* **4** (1996) no. 4, 295–300.

[113] D. A. Spielman. Faster isomorphism testing of strongly regular graphs. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 576–584. ACM, New York, 1996.

[114] A. P. Street and D. J. Street. *Combinatorics of Experimental Design*. Clarendon Press, Oxford, England, 1987.

[115] L. G. Valiant, The complexity of computing the permanent, *Theoret. Comput. Sci.* **8** (1979) no. 2, 189–201.

[116] L. G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput.* **8** (1979) no. 3, 410–421.

[117] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics*. Cambridge University Press, Cambridge, England, 1992.

[118] A. Wassermann, Finding simple t-designs with enumeration techniques, *J. Combin. Des.* **6** (1998) no. 2, 79–90.

[119] R. M. Wilson, Nonisomorphic Steiner triple systems, *Math. Z.* **135** (1974), 303–313.

[120] P. M. Winkler, Isometric embeddings in products of complete graphs, *Discrete Appl. Math.* 7 (1984) no. 2, 221–225.

[121] R. A. Wright, B. Richmond, A. Odlyzko, and B. D. McKay, Constant time generation of free trees, *SIAM J. Comput.* **15** (1986) no. 2, 540–548.

A NOTATION INDEX

Sets

$x \in X$	x is an element of X, 92
$x \notin X$	x is not an element of X, 92
$\lvert X \rvert$	cardinality of X, 92
$Y \subseteq X$	Y is a subset of X, 92
$Y \subset X$	Y is a proper subset of X, 92
$X \cup Y$	union of X and Y, 92
$X \cap Y$	intersection of X and Y, 92
$X \setminus Y$	difference of X and Y, 92
$X \times Y$	Cartesian product of X and Y, 92
$\{x \in X \,:\, P\}$	the set of all $x \in X$ with property P, 92
$\mathscr{P}[X]$	the set of all subsets of X, 92
$\mathscr{M}[X]$	the set of all multisets over X, 4
$\emptyset$	the empty set, 92
$\mathbb{N} = \{0, 1, 2, \ldots\}$	the set of nonnegative integers, 92
$\mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$	the set of integers, 92
$\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$	the set of the first q nonnegative integers, 92
$(x_0, x_1, \ldots, x_{n-1})$	ordered n-tuple, 92

Functions

$f : X \to Y$	f is a function of X into Y, 92
$f : x \mapsto y$	x maps to y under f, 92
Y^X	the set of all functions of X into Y, 92
$f \circ g$	composition of f and g, 93
f^{-1}	inverse of f, 93
$f\vert_Z$	restriction of f to Z, 93

Groups and group actions

$H \leq G$	H is a subgroup of G, 95
G/H	the set of all left cosets of H in G, 96
$H \backslash G$	the set of all right cosets of H in G, 96
$[G : H]$	index of H in G, 96
$G \mathbin{\backslash\!\backslash} X$	the set of all orbits of G on X, 97
Gx	orbit of x under action of G, 97
G_x	stabilizer of x in G, 97
$\mathrm{fix}_X(g)$	number of $x \in X$ fixed by g, 98
$G \times H$	direct product of G and H, 98
$K \rtimes Q$	semidirect product of K by Q, 98
$D \wr Q$	wreath product of D by Q, 100
$\mathrm{Sym}(X)$	symmetric group on X, 94
S_n	symmetric group on $\mathbb{Z}_n$, 94
$(i_0 \;\, i_1 \;\, \cdots \;\, i_{r-1})$	r-cycle with elements $i_0, i_1, \ldots, i_{r-1}$, 94

$\mathrm{Aut}(G)$	full automorphism group of G, 96
$\langle S \rangle$	group generated by S, 95

Block designs and resolutions

$\mathscr{B}(v,k,\lambda)$	the set of all $B(v,k,\lambda)$ designs over $\mathbb{Z}_v$, 8
$\mathscr{RB}(v,k,\lambda)$	the set of all $RB(v,k,\lambda)$ designs over $\mathbb{Z}_v$, 11
$\mathscr{R}(v,k,\lambda)$	the set of all resolutions of $RB(v,k,\lambda)$ designs over $\mathbb{Z}_v$, 11
$\mathscr{R}(\mathcal{B})$	the set of all resolutions of $\mathcal{B} \in \mathscr{RB}(v,k,\lambda)$, 14
$\mathrm{Aut}(\mathcal{B})$	full automorphism group of $\mathcal{B} \in \mathscr{B}(v,k,\lambda)$, 7
$\mathrm{Aut}(\mathcal{R})$	full automorphism group of $\mathcal{R} \in \mathscr{R}(v,k,\lambda)$, 13

Incidence systems

$\mathscr{IB}(v,k,\lambda)$	the set of all incidence systems that correspond to $B(v,k,\lambda)$ designs, 18
$\mathscr{IR}(v,k,\lambda)$	the set of resolved incidence systems in $\mathscr{IB}(v,k,\lambda)$, 20
$\mathrm{Aut}_P(A)$	point automorphism group of an $A \in \mathbb{Z}_2^{\mathbb{Z}_w \times \mathbb{Z}_b}$, 51

Error-correcting codes

d_H	the Hamming metric, 23
$A_q(n,d)$	the maximum cardinality of an $(n,d)_q$ code, 24
$\mathrm{Iso}(X,d)$	the set of all isometries of a metric space (X,d), 25
$\mathscr{C}_q(n,M,d)$	the set of all $(n,M,d)_q$ codes in $\mathscr{P}[\mathbb{Z}_q^n]$, 27
$\mathscr{LC}_q(n,M,d)$	the set of all $M \times n$ matrices over $\mathbb{Z}_q$ whose rows form an $(n,M,d)_q$ code, 28
$\mathrm{Aut}_P(A)$	point automorphism group of an $A \in \mathbb{Z}_q^{\mathbb{Z}_M \times \mathbb{Z}_s}$, 62

Miscellaneous

$\Leftrightarrow$	if and only if, 5
$A(i,\cdot)$	row i of a matrix A, 49
$A(\cdot,j)$	column j of a matrix A, 49
$\lfloor x \rfloor$	the greatest integer n for which $n \leq x$, 24
$[a,b]$	greatest common divisor of $a,b \in \mathbb{N}$, 28

B MATHEMATICAL PRELIMINARIES

B.1 TERMINOLOGY AND NOTATION

B.1.1 Sets, functions and relations

We use the following standard notation and terminology for sets. We write $x \in X$ to indicate that x is an element of the set X. Conversely, $x \notin X$ indicates that x is not an element of X. The cardinality of a finite set X is denoted by $|X|$. We write $Y \subseteq X$ to indicate that Y is a subset of X, and $Y \subset X$ to indicate that Y is a proper subset of X. Furthermore, we use the standard notation $X \cup Y$, $X \cap Y$ and $X \setminus Y$ for the union, intersection, and difference of X and Y, respectively.

The following sets are used frequently in this report and therefore deserve their own symbols. The empty set is denoted by $\emptyset$. The set of nonnegative integers is denoted by $\mathbb{N} = \{0, 1, 2, \dots\}$, and the set of integers by $\mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$. The set of the first q nonnegative integers is denoted by $\mathbb{Z}_q = \{0, 1, \dots, q-1\}$.

Two sets are said to be *disjoint* if their intersection is empty. A collection of sets $X_1, \dots, X_n$ is *pairwise disjoint* if $X_i \cap X_j = \emptyset$ for all $1 \le i < j \le n$. A set of pairwise disjoint nonempty subsets of a set X whose union is X is a *partition* of X. The elements of a partition are called *cells*.

We denote by $\{x \in X \ : \ P\}$ the set of all $x \in X$ with property P. The set of all subsets of a set X, including the empty set, is denoted by $\mathscr{P}[X]$. Additionally, we denote by $X \times Y$ the *Cartesian product* of X and Y. The n-fold Cartesian product of $X_0, X_1, \dots, X_{n-1}$ is denoted by $X_0 \times X_1 \times \cdots \times X_{n-1}$. We denote an ordered n-tuple in $X_0 \times X_1 \times \cdots \times X_{n-1}$ by $(x_0, x_1, \dots, x_{n-1})$, where $x_i \in X_i$ for all $i \in \mathbb{Z}_n$. The elements $x_0, x_1, \dots, x_{n-1}$ are called *coordinates* of the n-tuple. For notational convenience we abbreviate $\underbrace{X \times X \times \cdots \times X}_{n}$ as X^n.

A binary relation $R \subseteq X \times X$ is *reflexive* if $(x, x) \in R$ for all $x \in X$. (For notational convenience we shall henceforth write xRy to indicate $(x, y) \in R$.) If xRy implies yRx for all $x, y \in X$, then R is *symmetric*. Conversely, R is *antisymmetric* if, for all $x, y \in X$, xRy and yRx imply $x = y$. Furthermore, R is *transitive* if xRy and yRz imply xRz for all $x, y, z \in X$. If R is reflexive, symmetric, and transitive then R is an *equivalence relation* on X. An equivalence relation R on X partitions X into *equivalence classes*, which are defined by the rule that $x, y \in X$ are in the same cell (equivalence class) of the partition if and only if xRy.

For functions we use the standard notation $f : X \to Y$ to indicate that f is a function from X to Y. Moreover, we write either $f(x) = y$ or $f : x \mapsto y$ to indicate that $x \in X$ maps to $y \in Y$ under f. If the function is clear from the context, we often omit the "f :" and write simply $x \mapsto y$.

We write Y^X for the set of all functions from X to Y. This notation allows an easy identification of elements of the n-fold Cartesian product $Y^n = Y \times Y \times \cdots \times Y$ with functions in $Y^{\mathbb{Z}_n}$ in which an n-tuple $y = (y_0, y_1, \dots, y_{n-1}) \in Y^n$ corresponds to the function $y \in Y^{\mathbb{Z}_n}$ defined by

$y : i \mapsto y_i$ for all $i \in \mathbb{Z}_n$.

The *composition* of functions $f : Y \to Z$ and $g : X \to Y$ is the function $f \circ g : X \to Z$ defined by $x \mapsto f(g(x))$ for all $x \in X$. The *restriction* of a function $f : X \to Y$ into $Z \subseteq X$ is denoted by $f|_Z$. The *inverse* of a bijective function f is denoted by f^{-1}.

B.1.2 Lexicographic order

Let X be a nonempty set. A reflexive, antisymmetric and transitive binary relation $R \subseteq X \times X$ is a *partial order* on X. A partial order R is a *total order* if, for all $x, y \in X$, either xRy or yRx. We use $\leq$ to denote a total order relation. The symbol $<$ is used as a shorthand for $x \leq y$ and $x \neq y$.

Let $X_0, X_1, \ldots, X_{n-1}$ be totally ordered sets, and let $x = (x_0, x_1, \ldots, x_{n-1})$ and $y = (y_0, y_1, \ldots, y_{n-1})$ be elements of $X_0 \times X_1 \times \cdots \times X_{n-1}$. *Lexicographic order* on $X_0 \times X_1 \times \cdots \times X_{n-1}$ is defined by the rule $x \leq y$ if and only if either $x_i = y_i$ for all $i \in \mathbb{Z}_n$, or there exists a $j \in \mathbb{Z}_n$ such that $x_j < y_j$ and $x_i = y_i$ for all $i \in \mathbb{Z}_j$.

Let X and Y be totally ordered sets and let $f, g : X \to Y$. Lexicographic order on Y^X is defined by the rule $f \leq g$ if and only if either $f(x) = g(x)$ for all $x \in X$, or there exists a $x \in X$ such that $f(x) < g(x)$ and $f(z) = g(z)$ for all $z \in X$ that satisfy $z < x$.

B.2 GROUP THEORY

Our definitions for groups are based on [105] (cf. [2, 15, 33, 55]).

We use the standard multiplicative notation $g_1 * g_2$ for the image of an ordered pair (g_1, g_2) under $*$ instead of the more cumbersome $*(g_1, g_2)$.

Definition B.2.1 A set G is a *group* under a mapping $* : G \times G \to G$, called the *group operation* of G, if $*$ has the following properties:

(i) $(g_1 * g_2) * g_3 = g_1 * (g_2 * g_3)$ for all $g_1, g_2, g_3 \in G$; and

(ii) there exists a $1 \in G$ such that $g * 1 = g$ for all $g \in G$; and

(iii) for all $g \in G$ there exists a $g^{-1} \in G$ such that $g * g^{-1} = 1$.

Conditions (ii) and (iii) of the previous definition can be stated in a stronger form as (ii') and (iii').

Theorem B.2.2 *Let G be a group under $* : G \times G \to G$. Then,*

(ii') there exists a unique $1 \in G$, called the identity element *of G, such that $g * 1 = 1 * g = g$ for all $g \in G$; and*

(iii') for all $g \in G$ there exists a unique $g^{-1} \in G$, called the inverse *of g, such that $g * g^{-1} = g^{-1} * g = 1$.*

We write $(G, *)$ to indicate that G is a group under $*$ if explication of the group operation is necessary for clarity; if the group operation is clear from the context, then we write simply $g_1 g_2$ instead of $g_1 * g_2$.

A group G is *finite* if G is a finite set. The *order* of a finite group G is its cardinality.

B.2.1 Permutations, the symmetric group

Definition B.2.3 A *permutation* of a nonempty set X is a bijection of X onto itself.

We write $\mathrm{Sym}(X)$ for the set of all permutations of X.

Theorem B.2.4 $\mathrm{Sym}(X)$ *is a group under the composition operation* $\circ$.

The group $\mathrm{Sym}(X)$ is called the *symmetric group* on X. We abbreviate $\mathrm{Sym}(\mathbb{Z}_n)$ to S_n.

Definition B.2.5 Let $x \in X$ and $\pi \in \mathrm{Sym}(X)$. We say that π *fixes* x if $\pi(x) = x$; otherwise π *moves* x. Two permutations are *disjoint* if every point moved by one is fixed by the other.

Definition B.2.6 Let $i_0, i_1, \ldots, i_{r-1}$ be distinct integers in $\mathbb{Z}_n$. If $\pi \in S_n$ fixes the remaining $n - r$ integers and if

$$\pi(i_0) = i_1,\ \pi(i_1) = i_2,\ \ldots,\ \pi(i_{r-2}) = \pi(i_{r-1}),\ \pi(i_{r-1}) = i_0,$$

then π is an *r-cycle*, and is denoted by $(i_0\ \ i_1\ \ \cdots\ \ i_{r-1})$.

Example B.2.7 The permutation $\pi \in S_8$ with $\pi(0) = 5$, $\pi(1) = 1$, $\pi(2) = 6$, $\pi(3) = 2$, $\pi(4) = 0$, $\pi(5) = 4$, $\pi(6) = 3$, and $\pi(7) = 7$ can be written as a product of cycles $\pi = (0\;5\;4)(2\;6\;3)$. A pictorial representation of π in terms of its cycles is given below.

$$\begin{tikzcd}
0 \arrow[rr, bend left] & & 5 \arrow[dl, bend left] & & 6 \arrow[rr, bend left] & & 3 \arrow[dl, bend left] & \\
& 4 \arrow[ul, bend left] & & 1 \arrow[loop above] & & 2 \arrow[ul, bend left] & & 7 \arrow[loop above]
\end{tikzcd}$$

$\diamondsuit$

Theorem B.2.8 *Every permutation in $\pi \in S_n$ is either a cycle or a product of pairwise disjoint cycles.*

This cycle representation of $\pi \in S_n$ is unique up to ordering of the cycles if all points fixed by π are included in the representation as 1-cycles.

B.2.2 Subgroups, cosets and Lagrange's theorem

Definition B.2.9 A nonempty subset H of a group G is a *subgroup* of G if (i) $h^{-1} \in H$ for all $h \in H$; and (ii) $hh' \in H$ for all $h, h' \in H$.

We write $H \leq G$ to indicate that H is a subgroup of G.

Theorem B.2.10 *If $(G, *)$ is a group and $H \leq G$, then H is a group under the restriction of $*$ to $H \times H$.*

For a finite group G, condition (i) of Definition B.2.9 can be omitted:

Theorem B.2.11 *If G is a finite group and $H \subseteq G$, then $H \leq G$ if and only if $hh' \in H$ for all $h, h' \in H$.*

Example B.2.12 Let X be a nonempty set. A *permutation group* on X is a subgroup of $\mathrm{Sym}(X)$. The *degree* of a permutation group on X is $|X|$. $\diamondsuit$

Theorem B.2.13 *If G is a group, then the intersection of any collection of subgroups of G is a subgroup of G.*

Corollary B.2.14 *Let G be a group and let $S \subseteq G$. Then the set*

$$\langle S \rangle = \bigcap \{H \,:\, S \subseteq H \leq G\}$$

is a subgroup of G.

The group $\langle S \rangle$ is called the group *generated* by S and the elements of S are called *generators* of $\langle S \rangle$.

Example B.2.15 The 6-cycle $(0\;1\;2\;3\;4\;5)$ generates a subgroup of S_6 with 6 elements:

$$\begin{array}{ccc}
(0)(1)(2)(3)(4)(5), & (0\;1\;2\;3\;4\;5), & (0\;2\;4)(1\;3\;5), \\
(0\;3)(1\;4)(2\;5), & (0\;4\;2)(1\;5\;3), & (0\;5\;4\;3\;2\;1).
\end{array}$$

$\diamondsuit$

Definition B.2.16 Let G be a group and let $H \leq G$. For a $g \in G$ the set $gH = \{gh : h \in H\}$ is called the *left coset* of H in G which contains g. Similarly, $Hg = \{hg : h \in h\}$ is the *right coset*.

We write G/H and $H \backslash G$ for the sets of left and right cosets of H in G, respectively.

Theorem B.2.17 *For a group G and a $H \leq G$, any two left (right) cosets of H in G are either identical or disjoint, and the number of left cosets of H in G is equal to the number of right cosets of H in G.*

Definition B.2.18 For a group G and a $H \leq G$, the *index* of H in G, denoted by $[G : H]$, is the number of left (right) cosets of H in G.

Theorem B.2.19 (Lagrange) *If G is a finite group and $H \leq G$, then $|H|$ divides $|G|$ and $[G : H] = |G|/|H|$.*

Definition B.2.20 For a group G and a $H \leq G$, a subset $T \subseteq G$ is a *left (right) transversal* of H in G if T contains exactly one element from each left (right) coset of H in G.

B.2.3 Homomorphism and isomorphism

Definition B.2.21 Let $(G, *)$ and $(H, \circledast)$ be groups. A mapping $\vartheta : G \to H$ is a group *homomorphism* if $\vartheta(g_0 * g_1) = \vartheta(g_0) \circledast \vartheta(g_1)$ for all $g_0, g_1 \in G$.

Theorem B.2.22 *Let G and H be groups and let $\vartheta : G \to H$ be a group homomorphism. Then (i) $\vartheta(1_G) = 1_H$; and (ii) $\vartheta(g^{-1}) = \vartheta(g)^{-1}$ for all $g \in G$.*

Definition B.2.23 A group *isomorphism* is a bijective homomorphism. An isomorphism of G onto itself is an *automorphism*.

Definition B.2.24 The set of all automorphisms of G is a subgroup of $\mathrm{Sym}(G)$ that is called the *(full) automorphism group* of G and denoted by $\mathrm{Aut}(G)$.

B.2.4 Permutation representations and group actions

Definition B.2.25 For a group G and a nonempty set X, a homomorphism $\rho : G \to \mathrm{Sym}(X)$ is called a *permutation representation* of G on X. A permutation representation is *faithful* if it is injective.

Permutation representations are often more conveniently described in terms of group actions. Theorem B.2.27 below shows that these descriptions are equivalent.

Definition B.2.26 Let $(G, *)$ be a group, and let X be a nonempty set. A function $\cdot : G \times X \to X$ is called a *(left) action* of G on X if

(i) $1 \cdot x = x$ for all $x \in X$; and

(ii) $(g_0 * g_1) \cdot x = g_0 \cdot (g_1 \cdot x)$ for all $g_0, g_1 \in G$, and all $x \in X$.

Theorem B.2.27 *Let $\cdot : G \times X \to X$ be a group action. Then, the mapping $\rho : G \to \mathrm{Sym}(X)$ defined by $\rho : g \mapsto \rho_g$ and $\rho_g : x \mapsto g \cdot x$ for all $x \in X$ and $g \in G$ is a group homomorphism. Conversely, a homomorphism $\rho : G \to \mathrm{Sym}(X)$ defines an action on X by $(g, x) \mapsto \rho_g(x)$.*

To indicate the presence of a group action of G on X without explicating the action we say that G *acts* on X. For notational convenience we shall omit the "$\cdot$" and write simply gx whenever there is no danger of confusion.

Example B.2.28 Let $G \le \mathrm{Sym}(X)$ be a permutation group on X. Then the *induced action* of G on X is defined by $(\pi, x) \mapsto \pi(x)$ for all $\pi \in G$ and $x \in X$. $\diamondsuit$

The induced action of a permutation group can be extended to $\mathscr{P}[X]$ and other constructions involving subsets of X (such as set systems, cf. Definition 2.1.4) using Lemmata 2.1.9 and 2.1.10. It is customary to speak of the induced action also in this case.

B.2.5 Orbits and stabilizers

Definition B.2.29 Let G act on X. For every $x \in X$, the set

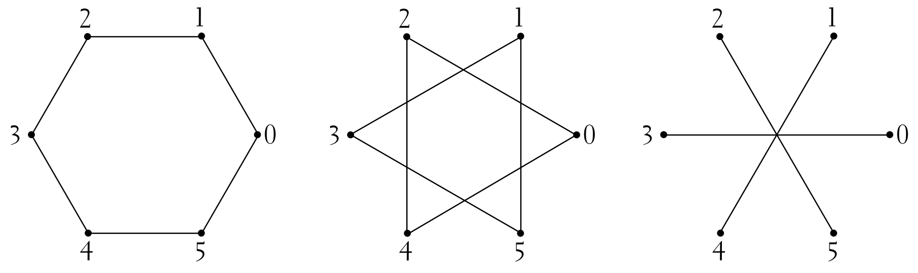
$$Gx = \{gx \,:\, g \in G\}$$

is called the *orbit* of x under G.

We write $G \setminus\!\!\setminus X$ for the set of all orbits of G on X.

Theorem B.2.30 *$G \setminus\!\!\setminus X$ is a partition of X.*

Example B.2.31 The induced action of $\langle(0\ 1\ 2\ 3\ 4\ 5)\rangle$ on the set of 2-subsets of $\mathbb{Z}_6$ produces the following partition into three orbits:



(Each straight line connecting two points represents a 2-subset.) $\diamondsuit$

Definition B.2.32 Let G act on X. For every $x \in X$, the *stabilizer* of x in G, denoted by G_x, is the subgroup

$$G_x = \{g \in G \,:\, gx = x\} \le G.$$

Let $x \in X$. The orbit Gx and the corresponding stabilizer subgroup G_x are connected by the following theorem.

Theorem B.2.33 (Orbit-Stabilizer Theorem) *Let G act on X and suppose $x \in X$. If T is a left transversal of G_x in G, then the mapping defined by $t \mapsto tx$ for all $t \in T$ is a bijection of T onto Gx.*

Corollary B.2.34 *Let G be a finite group acting on a finite set X. Then, for every $x \in X$, $|Gx| = [G : G_x]$.*

Theorem B.2.35 (Cauchy-Frobenius Lemma) *Let G be a finite group acting on a finite set X. Define $\mathrm{fix}_X(g) = |\{x \in X \ : \ gx = x\}|$ for all $g \in G$. Then,*

$$|G \setminus\!\!\setminus X| = \frac{1}{|G|} \sum_{g \in G} \mathrm{fix}_X(g). \tag{B.1}$$

The previous theorem is also known as *Burnside's Lemma*, however, see [91].

Definition B.2.36 The action of G on X is *transitive* if G has only one orbit on X.

Definition B.2.37 Let G act transitively on X. A nonempty subset $\Delta \subseteq X$ is a *block* of the action of G on X if, for all $g \in G$, either $g\Delta = \Delta$ or $g\Delta \cap \Delta = \emptyset$. A block is *trivial* if either $\Delta = \{x\}$ for some $x \in X$ or $\Delta = X$.

Definition B.2.38 Let G act transitively on X. The action of G on X is *primitive* if it has only trivial blocks.

Definition B.2.39 A permutation group on X is *transitive* (respectively, *primitive*) if its induced action on X is transitive (respectively, primitive).

B.2.6 Direct, semidirect and wreath products

Definition B.2.40 The Cartesian product $G \times H$ of groups $(G, *)$ and $(H, \circledast)$ is a group under the operation defined by $((g_1, h_1), (g_2, h_2)) \mapsto (g_1 * g_2, h_1 \circledast h_2)$ for all $(g_1, h_1), (g_2, h_2) \in G \times H$ and is called the *(external) direct product* of G and H.

The *n-fold direct product* for groups $G_1, \dots , G_n$ is defined analogously.

The group constructed in the following theorem is called a *semidirect product* of K by Q (realizing ϑ) and is denoted by $K \rtimes_\vartheta Q$.

Theorem B.2.41 *Let K and Q be groups and let $\vartheta : Q \to \mathrm{Aut}(K)$ where $q \mapsto \vartheta_q$ be a homomorphism. Then, $K \times Q$ is a group under the operation defined by*

$$(k_1, q_1), (k_2, q_2) \mapsto (k_1 \vartheta_{q_1}(k_2), q_1 q_2).$$

for all $(k_1, q_1), (k_2, q_2) \in K \times Q$.

Proof. We verify conditions (i)-(iii) of Definition B.2.1. For condition (i), let $(k_1, q_1), (k_2, q_2), (k_3, q_3) \in K \times Q$. From the definition we obtain

$$\begin{aligned}&((k_1, q_1)(k_2, q_2))(k_3, q_3) = \\ &\quad = (k_1\vartheta_{q_1}(k_2), q_1q_2)(k_3, q_3) = (k_1\vartheta_{q_1}(k_2)\vartheta_{q_1q_2}(k_3), q_1q_2q_3).\end{aligned}$$

Because ϑ is a homomorphism of Q into $\mathrm{Aut}(K)$, we have $\vartheta_{q_1q_2}(k_3) = \vartheta_{q_1}(\vartheta_{q_2}(k_3))$. Moreover, since ϑ_{q_1} is a homomorphism of K onto K, $\vartheta_{q_1}(k_2)\vartheta_{q_1}(\vartheta_{q_2}(k_3)) = \vartheta_{q_1}(k_2\vartheta_{q_2}(k_3))$. Thus,

$$\begin{aligned}&(k_1\vartheta_{q_1}(k_2)\vartheta_{q_1q_2}(k_3), q_1q_2q_3) = \\ &\quad = (k_1\vartheta_{q_1}(k_2\vartheta_{q_2}(k_3)), q_1q_2q_3) = (k_1, q_1)(k_2\vartheta_{q_2}(k_3), q_2q_3) = \\ &\quad = (k_1, q_2)((k_2, q_2)(k_3, q_3)).\end{aligned}$$

Condition (ii) holds for the element $(1_K, 1_Q) \in K \times Q$. To see this, note that $\vartheta_q(1_K) = 1_K$ for all $q \in Q$ by Theorem B.2.22. Thus, $(k, q)(1_K, 1_Q) = (k\vartheta_q(1_K), q1_Q) = (k, q)$ for all $(k, q) \in K \times Q$. For condition (iii), let $(k, q) \in K \times Q$ and put $(k, q)^{-1} = (\vartheta_{q^{-1}}(k^{-1}), q^{-1})$. Now, by Theorem B.2.22,

$$\begin{aligned}&(k, q)(\vartheta_{q^{-1}}(k^{-1}), q^{-1}) = \\ &\quad = (k\vartheta_q(\vartheta_{q^{-1}}(k^{-1})), qq^{-1}) = (k\vartheta_{qq^{-1}}(k^{-1}), 1_Q) = (k\vartheta_{1_Q}(k^{-1}), 1_Q) = \\ &\quad = (kk^{-1}, 1_Q) = (1_K, 1_Q),\end{aligned}$$

which establishes condition (iii) and completes the proof. ■

We require some preliminaries before we define the wreath product (which is a special case of a semidirect product). Let D be a group and suppose X is a finite nonempty set. Then we can give D^X a direct product structure by defining a group operation $(d_1, d_2) \mapsto d_1d_2$ on D^X by the rule

$$d_1d_2 : x \mapsto d_1(x)d_2(x)$$

for all $x \in X$ and $d_1, d_2 \in D^X$.

Theorem B.2.42 *Let Q act on a finite set X, and let D be a group. If D^X has the direct product structure, then the mapping $\vartheta : Q \to \mathrm{Aut}(D^X)$ defined by $\vartheta_q(d) : x \mapsto d(q^{-1}x)$ for all $x \in X$, $d \in D^X$, and $q \in Q$ is a well-defined homomorphism.*

Proof. We first show that ϑ_q is an automorphism of D^X for all $q \in Q$. Select a $q \in Q$. We note that ϑ_q simply maps a $d : x \mapsto d(x)$ to $\vartheta_q(d) : x \mapsto d(q^{-1}x)$. Thus, because $x \mapsto q^{-1}x$ is a bijection, ϑ_q is a bijection and its inverse is $\vartheta_{q^{-1}}$. Let $d_1, d_2 \in D^X$, and select an $x \in X$. Then,

$$\vartheta_q(d_1d_2) : x \mapsto d_1d_2(q^{-1}x) = d_1(q^{-1}x)d_2(q^{-1}x)$$

by definition of the direct product structure on D^X. Also, $\vartheta_q(d_1)\vartheta_q(d_2) : x \mapsto d_1(q^{-1}x)d_2(q^{-1}x)$. Since $x \in X$ was arbitrary, we have $\vartheta_q(d_1d_2) = \vartheta_q(d_1)\vartheta_q(d_2)$. We still have to show that ϑ is a homomorphism. Let $q_1, q_2 \in Q$ and $d \in D^X$. Then,

$$\begin{aligned}\vartheta_{q_1q_2}(d) &: x \mapsto d((q_1q_2)^{-1}x) = d(q_2^{-1}q_1^{-1}x), \\ \vartheta_{q_1}(\vartheta_{q_2}(d)) &: x \mapsto \vartheta_{q_2}(d)(q_1^{-1}x) = d(q_2^{-1}q_1^{-1}x)\end{aligned}$$

for all $x \in X$. Consequently, $\vartheta_{q_1q_2}(d) = \vartheta_{q_1}(\vartheta_{q_2}(d))$. ■

Definition B.2.43 Let D and Q be groups, and let Q act on a finite set X. The *wreath product* of D by Q, denoted by $D \wr Q$, is the semidirect product $D^X \rtimes_\vartheta Q$, where D^X has the direct product structure and $\vartheta : Q \to \mathrm{Aut}(D^X)$ is the homomorphism of Theorem B.2.42.

A wreath product $D \wr Q$ is clearly dependent on the action of Q on X. An important special case occurs with $Q \leq \mathrm{Sym}(X)$, where the action of Q on X is the induced action. In this case we say that $D \wr Q$ is the *permutation wreath product* of D by Q.

Example B.2.44 Suppose $S_q \wr S_n$ is the permutation wreath product of S_q by S_n. The elements of $S_q \wr S_n$ are then ordered pairs (μ, π), where $\mu \in S_q^{\mathbb{Z}_n}$ and $\pi \in S_n$. For notational convenience, we shall identify the mapping μ with the ordered n-tuple $(\mu_0, \dots , \mu_{n-1}) \in S_q^n$, where $\mu_i = \mu(i) \in S_q$ for all $i \in \mathbb{Z}_n$. The product $(\hat{\hat{\mu}}, \hat{\hat{\pi}})$ of two elements $(\mu, \pi), (\hat{\mu}, \hat{\pi})$ in $S_q \wr S_n$ is then defined by $\hat{\hat{\pi}} = \pi\hat{\pi}$ and $\hat{\hat{\mu}}_i = \mu_i \hat{\mu}_{\pi^{-1}(i)}$ for all $i \in \mathbb{Z}_n$. $\diamondsuit$

C COMPUTATIONAL COMPLEXITY

In this appendix we discuss computational complexity aspects of block designs and their resolutions motivated by surveys of Colbourn [24], Gibbons [42], and the recent survey of Goldberg [46]. (For reference on computational complexity theory and the complexity classes **P**, **NP**, #**P**, and #$\mathbf{P}_1$, see [40, 98, 116].) In particular, we shall consider decision and counting problems related to generation of block designs and resolutions. Furthermore, we briefly discuss the complexity of algorithms that output all solutions to a given problem instance.

C.1 EXISTENCE AND COMPLETION

First we discuss decision problems related to generation of block designs. A natural decision problem to consider is

Problem C.1.1 (BLOCK DESIGN EXISTENCE) Given parameters v,k,λ input as unary integers, decide whether a $B(v, k, \lambda)$ design exists.

The existence problem is obviously in **NP**, because a nondeterministic Turing machine can in time polynomial in the input size nondeterministically guess an incidence matrix and then verify that it corresponds to a block design. Due to the unary problem instance encoding, the problem is unlikely to be **NP**-complete (cf. [98, Theorem 14.3]), although it is by no means trivial. For example, the existence of a $B(22, 8, 4)$ design is at present undecided.

If we constrain the existence problem to the problem of completing a given partial design, then the problem becomes **NP**-complete.

Problem C.1.2 (PARTIAL DESIGN COMPLETION) Given parameters v,k,λ of a block design and a collection of blocks of cardinality k, decide whether the collection can be completed to a $B(v, k, \lambda)$ design.

(We assume that the collection of blocks is always input using a $v \times b$ incidence matrix to guarantee that the problem is in **NP**.) By a result of Colbourn [19], the problem of completing partial Steiner triple systems is **NP**-complete. Consequently, PARTIAL DESIGN COMPLETION is **NP**-complete, and it is likely that no efficient (polynomial-time) procedure exists for deciding whether a partial design can be completed to a block design by addition of blocks.

Given this result, it is natural to consider the complexity of the problem of completing a $w \times b$ incidence system to a block design by addition of points instead of blocks. The complexity of this problem is, to the best of our knowledge, undetermined.

A related problem is to decide whether a single extending row exists, that is, (recall Section 5.1.2) deciding whether the integer equation system (5.3) has a $\{0, 1\}$-solution, a problem clearly in **NP**. In the general case, the problem of deciding whether an equation system $Ax = y$ has a nonnegative integer solution $x \in \mathbb{N}^n$ for a given $m \times n$ integer matrix A and an m-vector y

is **NP**-complete [97]. It would be interesting to know whether the restriction from the generic case to systems similar to (5.3) remains **NP**-complete.

Another related problem is deciding, given all solutions to (5.3), whether there exists a collection of rows that completes the incidence system to a block design. Recalling Section 5.1.4, this problem reduces to deciding whether the corresponding compatibility graph contains a clique of appropriate size. The problem of deciding whether an arbitrary graph contains a clique of a given size is **NP**-complete [98, p. 190].

C.2 RESOLVABILITY

The second problem we discuss is deciding whether a block design is resolvable.

Problem C.2.1 (BLOCK DESIGN RESOLVABILITY) Decide whether a given $B(v, k, \lambda)$ design is resolvable.

The exact complexity of this problem is to our knowledge undetermined, although it is clearly in **NP**. (Nondeterministically guess a partition of the blocks and then verify in polynomial time that each cell of the partition is a parallel class.)

Deciding resolvability reduces to deciding whether a graph related to the design has a partition into cliques of fixed size.

Definition C.2.2 The *block intersection graph* of a $B(v, k, \lambda)$ design has as vertices all the blocks of the design. Two vertices are connected by an edge if and only if the corresponding blocks are disjoint.

A clique of order v/k in the block intersection graph of a $B(v, k, \lambda)$ design clearly corresponds to a parallel class. Thus, a partition of the block intersection graph into cliques of order v/k corresponds to a resolution of the design. More specifically, a resolution corresponds to a partition of the vertex set of the block intersection graph into sets of size v/k such that the subgraph induced by each of the sets is a v/k-clique.

For arbitrary graphs, deciding whether a partition of the above kind into m-cliques exists is **NP**-complete for $m \geq 3$. (The clique partition problem is a form of the **NP**-complete problem PARTITION INTO ISOMORPHIC SUBGRAPHS, see [40, p. 193].)

For $\lambda = 1$ the block intersection graphs of $B(v, k, \lambda)$ designs belong to a class of graphs of separate interest (see [16, Ch. 5]).

Definition C.2.3 A *strongly regular graph* with parameters (n, d, p, q) is a graph with n vertices, where each pair of vertices x, y is adjacent to d, p, or q common vertices according as x and y are equal, adjacent, or non-adjacent, respectively.

Theorem C.2.4 *The block intersection graph of a $B(v, k, 1)$ design is strongly regular with parameters*

$$n = b, \qquad d = k(r-1), \qquad p = k^2, \qquad q = (r-2) + (k-1)^2. \quad \text{(C.1)}$$

Proof. Note that two blocks of a $B(v, k, 1)$ design are either disjoint or intersect in exactly one point, and apply straightforward counting arguments. ■

By a result of Bose [9], the converse also holds for large enough n, that is, any strongly regular graph with appropriate parameters and a large enough n is a block intersection graph of a $B(v, k, 1)$ design. Furthermore, the design can be constructed up to isomorphism in polynomial time from the block intersection graph [113]. So, deciding resolvability of, for example, a Steiner triple system of large enough order is polynomial-time equivalent to deciding whether a corresponding strongly regular graph can be partitioned into $v/3$-cliques.

A related problem to BLOCK DESIGN RESOLVABILITY is deciding whether the design contains a parallel class.

Problem C.2.5 (PARALLEL CLASS EXISTENCE) Decide whether a given $B(v, k, \lambda)$ design contains a parallel class.

This naturally reduces to deciding whether the block intersection graph contains a v/k-clique. Again, the exact complexity of this problem is, to our knowledge, undetermined.

Deciding whether a regular graph contains a clique of given size is **NP**-complete. More precisely, the problem INDEPENDENT SET OF d-REGULAR GRAPH is **NP**-complete for $d > 2$ [36]. (A graph is *d-regular* if each of its vertices is adjacent to exactly d vertices.)

Problem C.2.6 (INDEPENDENT SET OF d-REGULAR GRAPH) Given a d-regular graph, decide whether it contains an *independent set* (a set of vertices in which no pair of vertices is connected by an edge) of a given size.

Strongly regular graphs are clearly a much more restricted class of graphs than regular graphs, so this **NP**-completeness result on regular graphs unfortunately does not tell us very much about the corresponding problem for strongly regular graphs.

For $B(v, 3, \lambda)$ designs PARALLEL CLASS EXISTENCE is a restriction of an **NP**-complete problem EXACT COVER BY 3-SETS.

Problem C.2.7 (EXACT COVER BY 3-SETS) Decide whether a collection of n 3-subsets of a $3m$-set contains m sets which partition the $3m$-set.

C.3 ISOMORPHISM TESTING AND CANONICAL PLACEMENT

The complexity of deciding isomorphism of two block designs and the complexity of computing canonical placement is clearly of interest in the context of isomorph-free generation of block designs.

Problem C.3.1 (BLOCK DESIGN ISOMORPHISM) Given two $B(v, k, \lambda)$ designs, decide whether they are isomorphic.

By a result of Colbourn [25], BLOCK DESIGN ISOMORPHISM is polynomial-time equivalent to graph isomorphism.

A related problem is deciding whether two resolutions are isomorphic.

Problem C.3.2 (Resolution Isomorphism) Given two resolutions of $RB(v,k,\lambda)$ designs, decide whether they are isomorphic.

Resolution Isomorphism can clearly be reduced in polynomial time to graph isomorphism. To our knowledge the converse has not been established.

The graph isomorphism problem and canonical placement of graphs have both received a lot of attention (see [65, 104]). The time complexity of McKay's graph canonical placement software *nauty* [81, 82] is discussed in [89]. Miller [88] develops a $v^{\log v+O(1)}$ time algorithm for canonical placement of Steiner systems. Babai and Luks [3] generalize the algorithm to handle $B(v,k,\lambda)$ designs. They also develop a polynomial-time algorithm for computing canonical placement of graphs with bounded vertex degree, and demonstrate that finding the graph with the lexicographic minimum/maximum adjacency matrix isomorphic to a given graph is **NP**-hard. A related theoretical result of Blass and Gurevich [6] states that, for certain polynomial time computable equivalence relations, determining the lexicographic minimum/maximum representative of an equivalence class is strictly harder than computing canonical placement unless $\mathbf{P} = \mathbf{NP}$.

C.4 LABELLED AND UNLABELLED COUNTING

A natural prerequisite to listing all solutions to a given problem is the ability to count them. Valiant's complexity classes $\#\mathbf{P}$ [115] and $\#\mathbf{P}_1$ [116] are suitable for discussing the computational complexity of counting problems. The complexity class $\#\mathbf{P}$ consists of problems solvable by a polynomial-time nondeterministic Turing machine in the following sense: The solution to a problem instance is the number of accepting computations of the Turing machine when input with the problem instance. The class $\#\mathbf{P}_1$ is the restriction of $\#\mathbf{P}$ to problems with a unary input alphabet. Both $\#\mathbf{P}$-complete and $\#\mathbf{P}_1$-complete problems exist, see [116].

We will first consider labelled counting of block designs and resolutions in this context.

Problem C.4.1 (#Labelled Block Designs) Given parameters v, k, λ input as unary integers, compute the number of labelled $B(v,k,\lambda)$ designs, that is, $|\mathscr{B}(v,k,\lambda)|$.

This problem is in $\#\mathbf{P}$, because a nondeterministic Turing machine can in polynomial time guess a column-ordered $v \times b$ incidence matrix and accept if and only if it corresponds to a block design. If either k, λ or v, k are fixed, then the problem is clearly in $\#\mathbf{P}_1$. For example, the problem of counting all labelled $STS(v)$ is in $\#\mathbf{P}_1$.

The problem of counting resolutions of $RB(v,k,\lambda)$ designs with unary parameter input is clearly also in $\#\mathbf{P}$. Furthermore, so is the problem of counting labelled resolutions of a given block design.

Problem C.4.2 (#Labelled Resolutions of Design) Given a $B(v,k,\lambda)$ design, compute the number of distinct labelled resolutions of the design.

The group-theoretic framework is natural for unlabelled counting. Let G be a group acting on a finite set X. Recall that the number of orbits in $G \backslash\!\backslash X$ can be counted using the Cauchy-Frobenius Lemma (Theorem B.2.35) as

$$|G \backslash\!\backslash X| = \frac{1}{|G|} \sum_{g \in G} \operatorname{fix}_X(g),$$

or, more conveniently, $|G||G \backslash\!\backslash X| = \sum_{g \in G} \operatorname{fix}_X(g)$. Based on the discussion in Chapter 2, counting the unlabelled block designs can be formulated as:

Problem C.4.3 (#UNLABELLED BLOCK DESIGNS) Given parameters v, k, λ input as unary integers, compute $|S_v||S_v \backslash\!\backslash \mathscr{B}(v,k,\lambda)|$.

We will argue that #UNLABELLED BLOCK DESIGNS is in $\#\mathbf{P}$. By the Cauchy-Frobenius Lemma, solving #UNLABELLED BLOCK DESIGNS is equivalent to computing $\sum_{\sigma \in S_v} \operatorname{fix}_{\mathscr{B}(v,k,\lambda)}(\sigma)$. Consider a nondeterministic Turing machine which, given v, k, λ input as unary integers, computes as follows. First, the machine guesses nondeterministically an arbitrary permutation $\sigma \in S_v$ and an arbitrary $\mathcal{B} \in \mathscr{B}(v,k,\lambda)$ (which is, for example, represented as a $v \times b$ column-ordered incidence matrix). The machine then accepts if and only if $\sigma\mathcal{B} = \mathcal{B}$. It is straightforward to design the machine so that the number of accepting computations is $\sum_{\sigma \in S_v} \operatorname{fix}_{\mathscr{B}(v,k,\lambda)}(\sigma)$, and that all computations either accept or reject in time polynomial in $v+k+\lambda$. Consequently, #UNLABELLED BLOCK DESIGNS is in $\#\mathbf{P}$, and any restriction to an unary input alphabet is in $\#\mathbf{P}_1$. Furthermore, the following observations are evident. First, since $|S_v| = v!$ is easy to compute, determining $|S_v \backslash\!\backslash \mathscr{B}(v,k,\lambda)|$ is hard. (Unless, of course, #UNLABELLED BLOCK DESIGNS is computable in polynomial time.) Second, the problem of computing the nonisomorphic resolutions in a given parameter family, that is, computing $|S_v||S_v \backslash\!\backslash \mathscr{R}(v,k,\lambda)|$ can be shown to be in $\#\mathbf{P}$ using a similar construction.

A related unlabelled counting problem is to determine the number of unlabelled resolutions of a given block design. Recall that by Corollary 2.3.21 $|\operatorname{Aut}(\mathcal{B}) \backslash\!\backslash \mathscr{R}(\mathcal{B})|$ is the number of nonisomorphic resolutions of a $\mathcal{B} \in \mathscr{B}(v,k,\lambda)$. An analogous application of the Cauchy-Frobenius Lemma as above establishes that the following problem is in $\#\mathbf{P}$.

Problem C.4.4 (#UNLABELLED RESOLUTIONS OF DESIGN) Given a $B(v,k,\lambda)$ design $\mathcal{B} \in \mathscr{B}(v,k,\lambda)$, compute $|\operatorname{Aut}(\mathcal{B})||\operatorname{Aut}(\mathcal{B}) \backslash\!\backslash \mathscr{R}(\mathcal{B})|$.

Computing $|\operatorname{Aut}(\mathcal{B})|$ is likely not to be easy due to the polynomial time equivalence between isomorphism testing of graphs and of block designs [25] and the fact that computing $|\operatorname{Aut}(\mathcal{G})|$ is polynomial-time equivalent to deciding graph isomorphism [76].

We conclude this section with two generic observations on unlabelled counting.

First, in general, counting the orbits of a group action on a set can be very hard ($\#\mathbf{P}$-complete). Suppose $G \leq \operatorname{Sym}(X)$ is a permutation group on a finite set X. We assume that G is given as a set of generator permutations. (Any permutation group on X can be described using at most $|X|-1$ generator permutations. Moreover, these can be computed from an arbitrary set of generator permutations in polynomial time; see for example [58].) The orbits of the induced action of G on X can be computed in polynomial time from the generator permutations using, for example, a transitive closure algorithm [74].

Not much needs to be changed in the situation above and the orbit counting problem becomes #**P**-complete. Let Y be a fixed finite set with $|Y| > 1$, and let G act on Y^X by $(\pi, \varphi) \mapsto \varphi \circ \pi^{-1}$ for all $\pi \in G$ and $\varphi \in Y^X$.

Problem C.4.5 (#PÓLYAORBITS [46]) Given a set of generator permutations for G, compute $|G||G \setminus\!\!\setminus Y^X|$.

The problem #PÓLYAORBITS is #**P**-complete [46, Theorem 3.1] (cf. [44, 59]) under polynomial-time Turing reductions with a function oracle. The order $|G|$ can be computed in polynomial time from the generator permutations (see [38, 74]), so determining $|G \setminus\!\!\setminus Y^X|$ is hard.

The second observation we make is that counting the unlabelled substructures of a given structure can be a lot harder than counting all the unlabelled structures (cf. counting all unlabelled resolutions of a design vs. counting all unlabelled resolutions in a parameter family). An example is provided by the fact that counting the unlabelled subtrees of a tree is #**P**-complete under polynomial-time Turing reductions with a function oracle [47], yet counting unlabelled trees with a given number of vertices is possible in time polynomial in the number of vertices [51].

C.5 LISTING ORBIT REPRESENTATIVES

The problem of generating orbit representatives was discussed in an abstract setting in Chapter 4, however, little attention was paid to computational complexity issues and the efficiency of the algorithms. It is not our intention to analyze the algorithms in detail here either, but only to illustrate how the efficiency of such algorithms could be measured, and to give pointers to literature of interest.

It is customary to regard as "efficient" an algorithm that computes the solution to a decision problem in time bounded by a polynomial in the input size. However, this notion of efficiency is not applicable to algorithms that output all solutions to a given problem instance simply because the number of solutions to the problem may be exponential in the input size. For example, the number of nonisomorphic $STS(v)$ has an exponential lower bound in v (Theorem 2.2.10).

Notions of efficiency for algorithms that generate all solutions to a given problem are discussed in [60]. We will consider two of these, namely polynomial space and polynomial delay.

An algorithm for generating all solutions to a problem instance uses *polynomial space* if the algorithm operates within a polynomial space bound in the input size. (It is assumed that a structure output by the algorithm is beyond its reach as soon as it is completely output. For a rigorous definition of space-bounded computation, see for example [98, p. 34–35].)

We note that the three algorithms described in Chapter 5 all operate within a polynomial space bound in $v + k + \lambda$. To see this, note that the first stage of both of the Read–Faradžev algorithms can be applied throughout the search, so the potentially space-demanding second stage of the algorithms need not be used. Furthermore, the McKay-type algorithm for code generation is guaranteed to operate within a polynomial space bound if explicit isomorphism testing is not used and the canonical placement algorithm has a polynomial space bound.

An algorithm for generating all solutions to a problem instance exhibits *polynomial delay* if the delay until the first solution is output, and thereafter the delay between any two successive solutions output, is bounded by a polynomial in the input size. This notion of efficiency is very strong in the sense that the existence of a polynomial-delay algorithm that generates all solutions to a problem for which the corresponding decision problem is $\mathbf{NP}$-complete (say, the problem of generating all cliques of given size for a given graph) clearly implies $\mathbf{P} = \mathbf{NP}$.

Polynomial delay algorithms are interesting in at least two respects. First, the order in which the structures are generated is significant. For example, Johnson *et al.* [60] present a polynomial delay algorithm for generating all maximal cliques of a graph in lexicographic order, yet they show that no polynomial delay algorithm for generating the maximal cliques of a graph in reverse lexicographic order exists unless $\mathbf{P} = \mathbf{NP}$. Second, polynomial delay algorithms exist for nontrivial problems, such as listing all the unlabelled graphs with a given number of vertices [43] (see also [45]).

It would be interesting to know whether it is possible to construct a polynomial delay algorithm for listing unlabelled block designs and/or resolutions with given parameters. However, because block designs are much more constrained structures than arbitrary graphs, and even BLOCK DESIGN EXISTENCE seems to be a hard problem, the existence of a polynomial delay algorithm seems very remote.

ISBN 951-22-5815-3
ISSN 1457-7615