

CLASSIFICATION OF RESOLVABLE BALANCED INCOMPLETE BLOCK DESIGNS — THE UNITALS ON 28 POINTS

PETTERI KASKI* — PATRIC R. J. ÖSTERGÅRD**

Dedicated to Alex Rosa on the occasion of his seventieth birthday

(Communicated by Peter Horák)

ABSTRACT. Approaches for classifying resolvable balanced incomplete block designs (RBIBDs) are surveyed. The main approaches can roughly be divided into two types: those building up a design parallel class by parallel class and those proceeding point by point. With an algorithm of the latter type — and by refining ideas dating back to 1917 and the doctoral thesis by Pieter Mulder — it is shown that the list of seven known resolutions of 2 -(28, 4, 1) designs is complete; these objects are also known as the resolutions of unital on 28 points.

©2009

Mathematical Institute
Slovak Academy of Sciences

1. Introduction

A 2 -(v, k, λ) *balanced incomplete block design* (BIBD) is a v -set V of *points* together with a collection of k -subsets of V , called *blocks*, with the property that every 2-subset of points is contained in exactly λ blocks. Two more parameters are associated to a BIBD: the number of blocks, b , and the number of blocks in which each point occurs, r . These central parameters of BIBDs are interrelated by

$$vr = bk, \quad r(k-1) = \lambda(v-1). \quad (1)$$

2000 Mathematics Subject Classification: Primary 05B05.

Keywords: classification, RBIBD, resolution, Steiner system, unital.

This research was supported in part by the Academy of Finland, Grants No. 107493, 110196, and 117499.

A *resolution* of a design is a partition of the blocks into *parallel classes*, which in turn partition the point set. A design with a resolution is said to be *resolvable*. Two resolutions of designs are *isomorphic* if there exists a bijection between the point sets of the resolutions that takes the parallel classes of one resolution onto the parallel classes of the other.

The most basic question with respect to resolvability — given a set of parameters, v , k , and λ — is whether 2 -(v, k, λ) resolvable BIBDs (RBIBDs for short) exist. Obvious necessary conditions for existence are that (1) has an integer solution and that k divides v . A comprehensive account of constructions and existence results for RBIBDs is [8]. If the existence question is answered in the affirmative, one may proceed and classify the RBIBDs and/or their resolutions up to isomorphism (RBIBDs may have several nonisomorphic resolutions).

In the 1980s, Mathon and Rosa [22], [23] collected and published the then known classification results for BIBDs and resolutions of RBIBDs (exact values and bounds) — an important undertaking, since the number of papers published on these topics had increased at the same speed as computers had become available to design theorists. Earlier surveys had been solely devoted to the existence problem. The seminal survey, with its range $r \leq 41$, became the standard reference, and updated versions have later been published by Mathon and Rosa in [24], [25], [26]. Tables of classification results for RBIBDs have later also been published in [2], [9], [15].

The first part of this paper, Section 2, gives a survey of the main computational approaches used to classify RBIBDs. These can roughly be divided into two types: those building up a design parallel class by parallel class and those proceeding point by point. In Section 3 we use an algorithm of the latter type to classify the 2 -(28, 4, 1) RBIBDs and show that the list of seven known resolutions of 2 -(28, 4, 1) designs is complete. These objects are also known as the resolutions of unitals on 28 points. The paper is concluded in Section 4.

2. Classification of RBIBDs

The survey of classification methods for RBIBDs to be presented here will be on a rather high level, emphasizing the main principles rather than details (in particular, implementational ones). The interested reader is encouraged to consult [15] for an in-depth treatment of any aspects of such a classification.

One obvious way of carrying out a classification of RBIBDs is to extract the desired objects from an exhaustive catalogue of BIBDs. However, since RBIBDs have more restrictions than BIBDs, direct construction of RBIBDs — to be more

precise, resolutions of RBIBDs — is to be preferred for computational reasons. Hence we do not consider this indirect approach further but refer the reader to [15, Sect. 6.3.1]. Modern classification results along this line include [20], [21], [32], [35], [36]. For certain sets of parameters, every BIBD is uniquely resolvable; see [15, Theorems 2.62, 2.63, 2.120, 2.127], and [19] for the particular case of 2-(27, 9, 4) RBIBDs.

Accidentally, from the early history of design theory and the first decades of the 20th century, there is a concrete example of obtaining resolutions either directly or indirectly: Cole [4] used a classification [5], [40] of Steiner triple systems of order 15 to get the Kirkman triple systems of order 15 — that is, the resolutions of the 2-(15, 3, 1) RBIBDs — whereas Mulder [33] used a direct approach.

On a general level, the computational problem of classifying RBIBDs is about backtrack search with isomorph rejection. The connection between resolutions and codes to be discussed next is very useful when setting up a framework for the computational work.

2.1. Resolutions are codes

We first recall some coding-theoretic definitions and notations. Let Z_q be an alphabet of q symbols; here $Z_q = \{0, 1, \dots, q-1\}$. A *word* of length n is an n -tuple $x = (x_1, x_2, \dots, x_n) \in Z_q^n$. The (*Hamming*) *distance* $d(x, y)$ between two words $x, y \in Z_q^n$ is the number of coordinates in which the words differ. Formally, $d(x, y) = |\{j \in \{1, 2, \dots, n\} : x_j \neq y_j\}|$.

An $(n, M, d)_q$ *code* is a set of M words of length n over Z_q , with the property that the distance between any two distinct words is at least d . A code is said to be *equidistant* if any two distinct codewords have the same mutual distance.

A *post* is a pair (j, v) where $1 \leq j \leq n$ is a coordinate and $v \in Z_q$ is an alphabet symbol. A post (j, v) is *flagged* by a word $x \in Z_q^n$ if $x_j = v$. In particular, every word $x \in Z_q^n$ flags exactly n posts. A code is *equireplicate* if every post is flagged by the same number of codewords.

Two codes are *equivalent* if one can be transformed into the other by a permutation of the coordinates followed by permutations of the coordinate values, independently for each coordinate.

Semakov and Zinov'ev [39] made the following observation.

THEOREM 1. *There is a one-to-one correspondence between the resolutions of 2-(qk, k, λ) designs and the $(r, qk, r - \lambda)_q$ codes, where $r = \lambda(qk - 1)/(k - 1)$ and $q > 1$. Such codes are necessarily equidistant and equireplicate.*

This correspondence is one-to-one in the strongest sense: nonisomorphic resolutions lead to inequivalent codes and vice versa; cf. [15, Theorem 3.82]. The correspondence is illustrated by the following example.

Example. Consider the Kirkman triple system on 9 points, that is, the unique resolution of the unique 2 -(9, 3, 1) BIBD. Construct a 9×4 matrix with one column for each parallel class and one row for each point. The three blocks of each parallel class are indicated by a number from $\{0, 1, 2\}$.

$$\begin{pmatrix} 0000 \\ 0111 \\ 0222 \\ 1012 \\ 1120 \\ 1201 \\ 2021 \\ 2102 \\ 2210 \end{pmatrix} \quad (2)$$

The rows of (2) form the codewords of an equidistant and equireplicate $(4, 9, 3)_3$ code, as anticipated by Theorem 1. This code is known as the ternary Hamming code of length 4.

2.2. Classification point by point

There are two main approaches for building up resolutions of designs when carrying out a classification. The two approaches proceed point by point and parallel class by parallel class, or, in the framework of codes, codeword by codeword and coordinate by coordinate, respectively. The design parameters restrict the candidates that need to be considered in both approaches. When proceeding point by point (in other words, codeword by codeword), the restrictions have a notable effect on the branching of the search tree from the very first levels; indeed, this approach is the one that has been used more often in the literature. Finally, adding isomorph rejection to this scheme gives a complete classification algorithm.

From here on, we use the framework of codes. When building up a code with the parameters given by Theorem 1 codeword by codeword, the only restriction applied is that the distance between codewords must be $r - \lambda$. Isomorph rejection can be accomplished by any of three main methods: recorded canonical forms of objects (not feasible if there are too many objects to store in memory), orderly generation ([7], [38]), and canonical augmentation ([29]); see also [15, Sect. 4.2].

Orderly generation, first used for classifying RBIBDs in [11], is the method commonly used in recent studies: then, after adding a row using a representation

as in the earlier example, the matrix is accepted iff it is minimal in its equivalence class. Equivalent matrices are obtained by permuting rows, columns, and symbols column-wise; cf. the definition of equivalent codes. In the minimality test, the strings obtained by concatenating the rows are compared. The rows are concatenated starting from the top row, and the first symbol is the most significant. See [15, Sect. 7.1.2] for details.

The calculations related to the isomorph rejection part use most of the CPU time in a classification of RBIBDs using orderly generation. One may therefore consider omitting isomorph rejection on certain levels of the search tree. If isomorph rejection is omitted on all remaining levels from a given level, then the search problem can be viewed as a clique problem in the graph with one vertex for each word that is at distance $r - \lambda$ from each of the fixed codewords, inserting edges between vertices with mutual distance $r - \lambda$.

Orderly generation codeword by codeword implies that the value of the first coordinate will be 0 in the first k codewords, 1 in the next k codewords, and so on. To minimize the size of the search tree, it is worth considering alternative ways of building up the RBIBD. Instead of considering candidates with a given value in the first coordinate, one may in fact consider any post that has been flagged less than k times and corresponding candidates. The choice with the smallest number of candidates should be preferred; it turns out that it is generally a good choice to consider posts that have been flagged $k - 1$ times. This idea is in fact implicitly used already in Pieter Mulder's doctoral thesis [33], published in 1917, where the seven nonisomorphic Kirkman triple systems of order 15 are classified by hand calculation. In the classification in Section 3, this idea will also be employed. Note that other isomorph rejection methods than orderly generation are required in this case.

In the following studies, RBIBDs have been classified using a point-by-point approach: [10], [11], [12], [36].

2.3. Classification parallel class by parallel class

The constraints when classifying RBIBDs parallel class by parallel class — coordinate by coordinate for codes — are rather weak in the beginning of the search. Favorable parameters are small λ , especially $\lambda = 1$, and small v , perhaps together with large k (for a small number of candidate parallel classes).

However, the main reason for using this approach has been the possibility of letting combinatorial properties of the resolutions restrict the search space, as shown by Morales and Velarde [30], [31] and briefly discussed in the following.

Let $Q = \{B_1, B_2, \dots, B_q\}$ and $Q' = \{B'_1, B'_2, \dots, B'_q\}$ be two different parallel classes in a resolution of a $2-(qk, k, \lambda)$ design. The (Q, Q') *parallel class intersection matrix* is the $q \times q$ matrix $A = (a_{ij})$ with entries given by $a_{ij} = |B_i \cap B'_j|$ for all $1 \leq i, j \leq q$.

If $v/k = q = 2$, then the parallel class intersection matrices are of the form

$$\begin{bmatrix} k-i & i \\ i & k-i \end{bmatrix},$$

where we without loss of generality may assume that $k \geq 2i$. For an arbitrary parallel class Q , we denote by n_i the corresponding number of matrices with parameter i associated with Q . Then

$$\sum_{i=0}^{\lfloor k/2 \rfloor} n_i = r - 1. \quad (3)$$

Moreover, by counting pairs of elements in any block of Q , we get that

$$\sum_{i=0}^{\lfloor k/2 \rfloor} \left[\binom{k-i}{2} + \binom{i}{2} \right] n_i = \binom{k}{2} (\lambda - 1). \quad (4)$$

The integer solutions of (3), (4) can now be used to restrict the search.

Analogous restrictions are obtained for larger q , and the approach is still feasible for $q = 3$ and small v ([30]).

In the following studies, RBIBDs have been classified parallel class by parallel class: [30], [31] and partially [10].

3. The resolutions of unitals on 28 points

A *unital* is a $2-(q^3 + 1, q + 1, 1)$ design. For $q = 3$, we get $2-(28, 4, 1)$ BIBDs, the resolutions of which are $(9, 28, 8)_7$ codes by Theorem 1. This section documents the seven known nonisomorphic resolutions of unitals on 28 points and the classification algorithm used to establish that the known isomorphism classes in fact constitute a complete set. Our aim is to include all details needed to repeat the search and to check the results for consistency; without the latter objective, presentation of several implementational details could have been omitted.

CLASSIFICATION OF RESOLVABLE BALANCED INCOMPLETE BLOCK DESIGNS

| R_1 | R_2 | R_3 | R_4 | R_5 | R_6 | R_7 |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 000000000 | 000000000 | 000000000 | 000000000 | 000000000 | 000000000 | 000000000 |
| 011111111 | 011111111 | 011111111 | 011111111 | 011111111 | 011111111 | 011111111 |
| 022222222 | 022222222 | 022222222 | 022222222 | 022222222 | 022222222 | 022222222 |
| 033333333 | 033333333 | 033333333 | 033333333 | 033333333 | 033333333 | 033333333 |
| 101234444 | 101234444 | 101234444 | 101234444 | 101234444 | 101234444 | 101234444 |
| 202315555 | 202315555 | 202143555 | 202143555 | 202441355 | 202341555 | 202143555 |
| 303126666 | 303126666 | 303425166 | 304412366 | 303555512 | 303452166 | 304415236 |
| 241440236 | 241440236 | 414523604 | 440155324 | 244205631 | 440535125 | 145542031 |
| 431552560 | 421553560 | 525603416 | 555024351 | 355213056 | 552036616 | 451626530 |
| 153650125 | 153650125 | 645021345 | 125166063 | 132106566 | 662660134 | 361052653 |
| 256523401 | 136506251 | 166502135 | 136405152 | 145450123 | 163516502 | 136165620 |
| 316054253 | 164142503 | 244636120 | 254256130 | 164643210 | 220656341 | 160421365 |
| 362560314 | 214604362 | 250312646 | 265301426 | 220563164 | 330164652 | 214662304 |
| 426616034 | 255263601 | 365216503 | 326350541 | 256026413 | 356205531 | 240316642 |
| 440364621 | 325405143 | 456035512 | 343046412 | 312664603 | 451643062 | 255404123 |
| 512643640 | 346351402 | 463640241 | 434061245 | 366302145 | 466354210 | 326536105 |
| 560155432 | 362560314 | 536241660 | 462436601 | 614536025 | 513625450 | 352360461 |
| 623461450 | 415346024 | 543514052 | 513265506 | 636415204 | 546142304 | 425150346 |
| 646102345 | 456012346 | 610446432 | 615440234 | 643162450 | 636521046 | 434501452 |
| 650246513 | 460625431 | 654104263 | 641362650 | 651340662 | 641406253 | 443064215 |
| 124045361 | 513462450 | 115355250 | 153510625 | 415042534 | 144320661 | 515035562 |
| 165412603 | 534255016 | 142460613 | 246614203 | 424351406 | 155461320 | 532456014 |
| 235066142 | 540514623 | 226454301 | 361625035 | 440616342 | 235015264 | 553241606 |
| 344631502 | 565031265 | 334062451 | 416523460 | 463424061 | 264103426 | 566603241 |
| 464203156 | 626164035 | 352651034 | 524603614 | 526630551 | 325540413 | 616324056 |
| 534424015 | 632641640 | 421366065 | 560542143 | 535521640 | 414062543 | 620645413 |
| 555301264 | 644023154 | 560165324 | 632554016 | 550154235 | 524414035 | 644253160 |
| 615535026 | 650436512 | 631550526 | 650631562 | 561065326 | 615253605 | 663510524 |

FIGURE 1. The seven nonisomorphic resolutions of unitals on 28 points

3.1. The seven resolutions

For completeness, we list the seven known resolutions of unitals on 28 points in Figure 1. Each resolution is given as a 28×9 array; each row of an array is a codeword of a corresponding $(9, 28, 8)_7$ code. The rows are listed in the order they are constructed by the algorithm used in this work.

These seven resolutions are presented by Mathon and Rosa in [22]. Brouwer [3] obtained all but one (R_7) of the resolutions and underlying resolvable unitals a few years earlier. Penttinen and Royle [37] classified all unitals on 28 points that admit embedding in a projective plane of order 9, finding four nonisomorphic resolvable unitals and resolutions. Krčadinac [17] classified all unitals on 28 points admitting a nontrivial automorphism, finding six nonisomorphic resolvable unitals and seven resolutions.

The automorphism groups of the seven resolutions are, in the order they appear in Fig. 1, 432, 216, 48, 48, 1512, 168, and 72, and the underlying unitals have automorphism groups of order 12096, 216, 48, 48, 1512, 1512, and 72, respectively.

The resolution R_1 is the unique resolution of the classical (hermitian) unital in $\text{PG}(2, 9)$.

A succinct description of R_2 is given by Hall [9]: expand the three base blocks

$$\begin{aligned} &\{(0, 2, 1), (0, 0, 1), (1, 1, 2), (1, 1, 0)\}, \\ &\{(0, 2, 0), (1, 2, 2), (0, 0, 2), (1, 0, 0)\}, \\ &\{\infty, (0, 1, 1), (1, 1, 1), (2, 1, 1)\} \end{aligned}$$

modulo $(3, 3, 3)$ while keeping the special point ∞ fixed; a parallel class of the resolution is obtained by adding $(0, 0, 0)$, $(1, 0, 0)$, $(2, 0, 0)$ to the base blocks. The unital underlying R_2 admits an embedding into $\text{PG}(2, 9)$.

The resolutions R_3 and R_4 form a dual pair in terms of embeddability of their underlying unitals; the unital underlying R_3 embeds into the Hall plane of order 9, whereas its nonisomorphic dual (underlying R_4) embeds into the dual Hall plane of order 9.

The resolutions R_5 and R_6 are the two nonisomorphic resolutions of the Ree unital on 28 points; this unital cannot be embedded in a projective plane of order 9.

A succinct description of R_7 is given by Mathon and Rosa [22] (who also present the other six resolutions in a similar, compact form): expand the parallel class

$$\begin{aligned} &\{(0, 1)_0, (2, 0)_0, (1, 0)_1, (0, 2)_2\}, \quad \{(1, 1)_0, (1, 2)_0, (2, 1)_0, (2, 2)_1\}, \\ &\{(0, 1)_1, (2, 0)_1, (1, 0)_2, (0, 2)_0\}, \quad \{(1, 1)_1, (1, 2)_1, (2, 1)_1, (2, 2)_2\}, \\ &\{(0, 1)_2, (2, 0)_2, (1, 0)_0, (0, 2)_1\}, \quad \{(1, 1)_2, (1, 2)_2, (2, 1)_2, (2, 2)_0\}, \\ &\{\infty, (0, 0)_0, (0, 0)_1, (0, 0)_2\} \end{aligned}$$

modulo $(3, 3)$ while keeping the special point ∞ fixed. The unital underlying R_7 cannot be embedded in a projective plane of order 9.

3.2. Classification

We rely on a point by point (that is, word by word) approach to classify the resolutions of unitals on 28 points. For reasons of performance, the search is divided into three phases:

- (a) word by word generation with isomorph rejection,

- (b) clique search in a compatibility graph,
- (c) completion to a resolution of a unital.

Before describing the individual phases in more detail, however, it is convenient to first outline the basic algorithm that generates individual words that extend a given partial resolution.

3.2.1. Generating compatible words

By virtue of Theorem 1 and the earlier discussion, we consider codes with $M = 28$ words of length $n = 9$ over an alphabet of $q = 7$ symbols, such that

- (i) any two distinct words have distance $d = 8$;
- (ii) every post is flagged exactly $M/q = 4$ times by the codewords.

Put otherwise, to extend a code $C \subseteq Z_q^n$ corresponding to a partial solution, it suffices to find all words $x \in Z_q^n$ such that

- (i) x has distance d to all the words in C ;
- (ii) each post is flagged at most M/q times by the words in $C \cup \{x\}$.

We say that such words x are *compatible* with C .

At implementation level, the algorithm for generating compatible words is a backtrack search that sets the values x_1, x_2, \dots, x_n one at a time so that all words x compatible with C are generated in lexicographical order from smallest to largest. Here we assume lexicographical ordering defined by the rule $x < y$ if and only if there exists an $1 \leq i \leq n$ such that both $x_i < y_i$ and $x_j = y_j$ holds for all $1 \leq j < i$.

As it turns out, the equireplication property of resolutions of unitals provides additional control over the compatible words that need to be considered. To see this, consider a code C corresponding to a partial solution. Because some words remain to be added to C , some posts are flagged by less than M/q words of C . To complete C , these posts must eventually be flagged by M/q words. Thus, when extending C with a new word, instead of considering every compatible word in turn, one can (arbitrarily) select one post that is not flagged M/q times, and consider only those compatible words that flag the selected post.

In essence, word by word generation of resolutions of unitals now amounts to recursively extending a given code with each compatible word that flags a selected post. However, to obtain a practical algorithm, equivalent codes need to be rejected during the first few recursive steps, which makes taking advantage of the equireplication property somewhat more elaborate.

3.2.2. Word by word generation with isomorph rejection

For codes with at most 9 words, we employ full rejection of equivalent codes. More precisely, at each recursive invocation, we check whether the input code is equivalent to a code encountered at an earlier invocation. If yes, the code is considered no further. At implementation level, the codes encountered are represented as colored graphs, whose canonical forms produced by *nauty* ([27]) are stored in memory; we refer to [15, Chaps. 3 and 4] for a detailed discussion.

To avoid generating an abundance of equivalent codes, we constrain the generation of compatible words. To motivate these constraints, we make the following observations. First of all, the first codeword is without loss of generality the all-zero word; in the sequel we therefore assume that $|C| \geq 1$. For a code $C \subseteq Z_q^n$, let x be a word compatible with C . If $y_i = y_{i+1}$ holds for every $y \in C$, then we can require $x_i \leq x_{i+1}$. Indeed, if $x_i > x_{i+1}$, then by virtue of lexicographical generation of compatible words, we have already encountered a code equivalent to $C \cup \{x\}$. Finally, by similar reasoning, we can require that $x_i \leq 1 + \max_{y \in C} y_i$.

It remains to develop these observations into an algorithm that also takes advantage of equireplication. Let a code $C \subseteq Z_q^n$ be given. Initially, let C consist of only the all-zero word. Call a post (j, v) *active* with respect to C if the following three properties hold:

- (ii') the words of C flag (j, v) less than M/q times;
- (iii') either $j = 1$ or there exists an $y \in C$ such that $y_{j-1} \neq y_j$; and
- (iv') $v \leq 1 + \max_{y \in C} y_j$.

For each active post (j, v) , the algorithm generates, in lexicographical order, all words $x \in Z_q^n$ that satisfy the following four properties:

- (i) x has distance d to all the words in C ;
- (ii⁺) x flags (j, v) and $C \cup \{x\}$ flags every post at most M/q times;
- (iii) for all $1 \leq i < n$ with $i \neq j$ it holds that $y_i = y_{i+1}$ for all $y \in C$ implies $x_i \leq x_{i+1}$; and
- (iv) for all $1 \leq i \leq n$ it holds that $x_i \leq 1 + \max_{y \in C} y_i$.

We say that these words are *associated with* (j, v) . When all active posts have been considered, we select an active post with the minimum number of associated words. If there are many such active posts, then we select the minimum one with respect to lexicographical ordering. Lexicographical ordering of posts is defined by the rule $(j_1, v_1) < (j_2, v_2)$ if and only if either $j_1 < j_2$ or both $j_1 = j_2$ and $v_1 < v_2$. Once an active post has been selected, the algorithm

considers, in lexicographical order, each word x associated with the selected post and recursively invokes itself with the input $C \cup \{x\}$.

The algorithm produces 140297 inequivalent codes with 9 words; in what follows we call these codes *seeds*. Before proceeding to the next phase of the search, the same algorithm is used to extend the seeds by one more word, but the resulting 10-word codes are no longer checked for equivalence against codes encountered earlier. (Note, however, that the constraints (iii), (iv), (iii'), and (iv') still eliminate some equivalent codes.)

3.2.3. Clique search

The next phase of the search takes as input the 10-word codes obtained by the approach described in the previous section. For each such code C , we first generate all words compatible with C . For a post (j, v) , denote by $f(j, v)$ the number of times (j, v) is flagged by the words in C . Denote by $N(j, v)$ the number of words compatible with C that flag (j, v) . Next, we find a pair of posts of the form $(j, v_1), (j, v_2)$, $v_1 \neq v_2$, such that $f(j, v_1) < M/q$, $f(j, v_2) < M/q$, and $N(j, v_1) + N(j, v_2)$ is maximized. If there is more than one such pair, then we select the pair with lexicographically minimum posts. Next, we construct a graph whose vertices are the words compatible with C that flag neither of the selected posts; an edge connects two words if and only if the words have mutual distance d . Once the graph is constructed, we find all its cliques of size

$$\begin{aligned} M - 10 - (M/q - f(j, v_1)) - (M/q - f(j, v_2)) \\ = M - 10 - 2M/q + f(j, v_1) + f(j, v_2) \end{aligned}$$

using the program Cliquer ([34]). Whenever such a clique D is found, we invoke the final phase with input $C \cup D$.

The final phase of the search completes a given code to a code corresponding to a resolution of a unital in all possible ways. This can again be done in a clique search, or, since this is a computationally easy task, by a routine backtrack search.

3.2.4. Search statistics

In total, the algorithm outputs 1284 codes corresponding to resolutions of unitals. After rejection of equivalent codes, seven inequivalent codes remain. The search was executed in batch jobs of 100 seeds using the batch system `autoson` ([28]) on a network of 139 Linux PCs, the majority of which are equipped with 2.8-GHz Pentium 4 CPUs and 512KB of in-processor cache. In total, about 1.5 years of CPU time was consumed.

3.3. Validation of the results

Validation of the results is of central importance in any computer-aided study; see [15, Chap. 10] for a survey of validation methods. In particular, consistency checking ([18]) in various form has turned out to be practicable and has been utilized frequently in recent work, including [13], [16].

Consistency checking of computational combinatorial results is almost without exception based on double counting. Occasionally, it is possible to use data obtained from the computations directly for this purpose. If the number of objects in the final classification is small, then a more careful analysis can be carried out to get an even higher confidence in the results than by mere double counting. We will now discuss how such an analysis can be done for the current classification; this analysis also illustrates the main ideas behind most of the validation methods of this type.

To give a rough intuition for the following more technical development, the idea is to dissect each of the seven inequivalent codes U' in all possible ways into $U' = S' \cup T'$, where S' is a code equivalent to a seed, and T' is the part that extends the seed. We then check whether the data obtained from this process is consistent with the 1284 codes produced in the search.

In precise terms, we count in two different ways all three-tuples (S, x, T) such that

- (a) S is one of the 140297 seeds;
- (b) $S \cup T$ is a completed code; and
- (c) $x \in T$ flags the active post selected by the classification algorithm when extending S .

The first count uses the 1284 codes obtained in the search. Indeed, observe that the search constructs three-tuples (S, x, T) satisfying (a), (b), and (c). The search has one additional restriction, however. Namely, x , the final word that is added in the first phase of the algorithm, always satisfies (iii) and (iv) with $C = S$. Thus, to arrive at the required three-tuples, all one has to do is to relabel each (S, x, T) constructed in the search in all possible ways so that S remains fixed and x flags the selected active post. In this way 2790 three-tuples are obtained.

The second count relies on dissecting the seven classified codes and uses the orbit-stabilizer theorem. To this end, it is convenient to work with a group action that captures equivalence of codes. Denote by S_n the symmetric group of degree n , and denote by $S_q \wr S_n$ the wreath product group that acts on Z_q^n by permuting the coordinates and the symbols independently in each coordinate. Extend the action to an elementwise action on sets and tuples constructed from words in Z_q^n .

For such an object, X , denote by $\text{Aut}(X)$ the stabilizer subgroup of X under the action of $S_q \wr S_n$. For example, for a code $C \subseteq Z_q^n$, $\text{Aut}(C) = \{g \in S_q \wr S_n : C^g = C\}$. For a pair of codes, $\text{Aut}(C_1, C_2) = \{g \in S_q \wr S_n : C_1^g = C_1, C_2^g = C_2\}$.

The second count is now carried out as follows. Each of the seven codes U' contains $\binom{28}{9} = 6906900$ subcodes with 9 words. Starting with the zero count and considering each U' in turn, we dissect U' in all possible ways into $U' = S' \cup T'$, where $|S'| = 9$ and $|T'| = 19$. Whenever S' is equivalent to a seed S , we accumulate the count as follows. First, we determine the number $h(S)$ of words in S that flag the active post selected by the classification algorithm when extending S . Next, we accumulate the count by $|\text{Aut}(S')|(M/q - h(S))/|\text{Aut}(U')|$; note that this number is in general not an integer. Once all codes U' and all dissections are considered, we output the accumulated count.

To see that the correct count is obtained, consider an arbitrary seed S and its extension T . Each such pair (S, T) needs to be counted $M/q - h(S)$ times to obtain the correct count. By the orbit-stabilizer theorem, the $\text{Aut}(S)$ -orbit of (S, T) has exactly $|\text{Aut}(S)|/|\text{Aut}(S, T)|$ pairs. Thus, it suffices to accumulate the count by $|\text{Aut}(S)|/|\text{Aut}(S, T)|(M/q - h(S))$ units for each such $\text{Aut}(S)$ -orbit. By the orbit stabilizer theorem, a pair from this $\text{Aut}(S)$ -orbit is encountered exactly $|\text{Aut}(U)|/|\text{Aut}(S, T)|$ times while dissecting (any code U' equivalent to) $U = S \cup T$. In particular, as each encounter accumulates the count by $|\text{Aut}(S)|(M/q - h(S))/|\text{Aut}(U)|$, the correct count is obtained.

As it turns out, the second count is identical to the first count. Furthermore, as an additional validation measure, both authors independently implemented the algorithm for classifying the seeds, with identical results. This gives us confidence that the classification is correct.

4. Conclusions

In this paper general approaches for classifying RBIBDs have been outlined, and the particular case of resolutions of unitals on 28 points has been studied and settled. In the future, with increasing computing speeds, a few more instances can certainly be settled without any fundamentally new ideas. For parameters with a prohibitively large number of nonisomorphic RBIBDs there is perhaps not so much more to hope for. For parameters with very few RBIBDs — or even none — on the other hand, the prospects of a breakthrough are much better. Throwing out an idea, perhaps linear programming could be useful in proving nonexistence; cf. [1].

Finally, note that 1-factorizations of complete graphs can be viewed as resolutions of $2-(v, 2, 1)$ designs. Recent computational studies on the problem of classifying such objects include [6], [14].

REFERENCES

- [1] APPA, G.—MAGOS, D.—MOURTOS, I.: *An LP-based proof for the non-existence of a pair of orthogonal Latin squares of order 6*, Oper. Res. Lett. **32** (2004), 336–344.
- [2] BETH, T.—JUNGNICKEL, D.—LENZ, H.: *Design Theory, Vol. I, II* (2nd ed.), Cambridge University Press, Cambridge, 1999.
- [3] BROUWER, A. E.: *Some unitals on 28 points and their embeddings in projective planes of order 9*. In: Geometries and Groups (M. Aigner, D. Jungnickel, eds.). Lecture Notes in Math. 893, Springer, Berlin, 1981, pp. 183–188.
- [4] COLE, F. N.: *Kirkman parades*, Bull. Amer. Math. Soc. **28** (1922), 435–437.
- [5] COLE, F. N.—CUMMINGS, L. D.—WHITE, H. S.: *The complete enumeration of triad systems in 15 elements*, Proc. Natl. Acad. Sci. USA. **3** (1917), 197–199.
- [6] DINITZ, J. H.—GARNICK, D. K.—MCKAY, B. D.: *There are 526,915,620 nonisomorphic one-factorizations of K_{12}* , J. Combin. Des. **2** (1994), 273–285.
- [7] FARADŽEV, I. A.: *Constructive enumeration of combinatorial objects*. In: Problèmes Combinatoires et Théorie des Graphes (Université d’Orsay, July 9–13, 1977), CNRS, Paris, 1978, pp. 131–135.
- [8] FURINO, S.—MIAO, Y.—YIN, J.: *Frames and Resolvable Designs. Uses, Constructions, and Existence*, CRC Press, Boca Raton, 1996.
- [9] HALL, M., Jr.: *Combinatorial Theory* (2nd ed.), Wiley, New York, 1986.
- [10] KASKI, P.—MORALES, L. B.—ÖSTERGÅRD, P. R. J.—ROSENBLUETH, D. A.—VELARDE, C.: *Classification of resolvable $2-(14, 7, 12)$ and $3-(14, 7, 5)$ designs*, J. Combin. Math. Combin. Comput. **47** (2003), 65–74.
- [11] KASKI, P.—ÖSTERGÅRD, P. R. J.: *There exists no $(15, 5, 4)$ RBIBD*, J. Combin. Des. **9** (2001), 357–362.
- [12] KASKI, P.—ÖSTERGÅRD, P. R. J.: *Miscellaneous classification results for 2-designs*, Discrete Math. **280** (2004), 65–75.
- [13] KASKI, P.—ÖSTERGÅRD, P. R. J.: *The Steiner triple systems of order 19*, Math. Comp. **73** (2004), 2075–2092.
- [14] KASKI, P.—ÖSTERGÅRD, P. R. J.: *One-factorizations of regular graphs of order 12*, Electron. J. Combin. **12** (2005) No. 1, #R2, 25pp.
- [15] KASKI, P.—ÖSTERGÅRD, P. R. J.: *Classification Algorithms for Codes and Designs*, Springer, Berlin, 2006.
- [16] KASKI, P.—ÖSTERGÅRD, P. R. J.—POTTONEN, O.: *The Steiner quadruple systems of order 16*, J. Combin. Theory Ser. A **113** (2006), 1764–1770.
- [17] KRČADINAC, V.: *Steiner 2-designs $S(2, 4, 28)$ with nontrivial automorphisms*, Glas. Mat. Ser. III **37(57)** (2002), 259–268.
- [18] LAM, C. W. H.—THIEL, L.: *Backtrack search with isomorph rejection and consistency check*, J. Symbolic Comput. **7** (1989), 473–485.
- [19] LAM, C.—TONCHEV, V. D.: *Classification of affine resolvable $2-(27, 9, 4)$ designs*, J. Statist. Plann. Inference **56; 86** (1996; 2000), 187–202; 277–278.

CLASSIFICATION OF RESOLVABLE BALANCED INCOMPLETE BLOCK DESIGNS

- [20] MATHON, R.—LOMAS, D.: *A census of 2-(9,3,3) designs*, Australas. J. Combin. **5** (1992), 145–158.
- [21] MATHON, R.—ROSA, A.: *A census of Mendelsohn triple systems of order nine*, Ars Combin. **4** (1977), 309–315.
- [22] MATHON, R.—ROSA, A.: *Some results on the existence and enumeration of BIBD's*. Mathematics Report 125-Dec-1985, Department of Mathematics and Statistics, McMaster University, Hamilton, 1985.
- [23] MATHON, R.—ROSA, A.: *Tables of parameters of BIBDs with $r \leq 41$ including existence, enumeration, and resolvability results*, Ann. Discrete Math. **26** (1985), 275–307.
- [24] MATHON, R.—ROSA, A.: *Tables of parameters of BIBDs with $r \leq 41$ including existence, enumeration and resolvability results: An update*, Ars Combin. **30** (1990), 65–96.
- [25] MATHON, R.—ROSA, A.: *2-(v,k, λ) designs of small order*. In: The CRC Handbook of Combinatorial Designs (C. J. Colbourn, J. H. Dinitz, eds.), CRC Press, Boca Raton, 1996, pp. 3–41.
- [26] MATHON, R.—ROSA, A.: *2-(v,k, λ) designs of small order*. In: Handbook of Combinatorial Designs (C. J. Colbourn, J. H. Dinitz, eds.; 2nd ed.), Chapman & Hall/CRC Press, Boca Raton, 2007, pp. 25–58.
- [27] MCKAY, B. D.: *nauty User's guide (version 1.5)*. Technical Report TR-CS-90-02, Computer Science Department, Australian National University, Canberra, 1990.
- [28] MCKAY, B. D.: *autoson — A distributed batch system for UNIX workstation networks (version 1.3)*. Technical Report TR-CS-96-03, Computer Science Department, Australian National University, Canberra, 1996.
- [29] MCKAY, B. D.: *Isomorph-free exhaustive generation*, J. Algorithms **26** (1998), 306–324.
- [30] MORALES, L. B.—VELARDE, C.: *A complete classification of (12,4,3)-RBIBDs*, J. Combin. Des. **9** (2001), 385–400.
- [31] MORALES, L. B.—VELARDE, C.: *Enumeration of resolvable 2-(10,5,16) and 3-(10,5,6) designs*, J. Combin. Des. **13** (2005), 108–119.
- [32] MORGAN, E. J.: *Some small quasi-multiple designs*, Ars Combin. **3** (1977), 233–250.
- [33] MULDER, P.: *Kirkman-Systemen*. PhD Thesis, Rijksuniversiteit Groningen, 1917.
- [34] NISKANEN, S.—ÖSTERGÅRD, P. R. J.: *Cliquer user's guide, version 1.0*. Technical Report T48, Communications Laboratory, Helsinki University of Technology, Espoo, 2003.
- [35] ÖSTERGÅRD, P. R. J.: *Enumeration of 2-(12,3,2) designs*, Australas. J. Combin. **22** (2000), 227–231.
- [36] ÖSTERGÅRD, P. R. J.—KASKI, P.: *Enumeration of 2-(9,3, λ) designs and their resolutions*, Des. Codes Cryptogr. **27** (2002), 131–137.
- [37] PENTTILÄ, T.—ROYLE, G. F.: *Sets of type (m,n) in the affine and projective planes of order nine*, Des. Codes Cryptogr. **6** (1995), 229–245.
- [38] READ, R. C.: *Every one a winner; or, How to avoid isomorphism search when cataloguing combinatorial configurations*, Ann. Discrete Math. **2** (1978), 107–120.
- [39] SEMAKOV, N. V.—ZINOV'EV, V. A.: *Equidistant q-ary codes with maximal distance and resolvable balanced incomplete block designs*, Problemy Peredachi Informatsii **4** (1968), No. 2, 3–10 (Russian). [English translation: Probl. Inf. Transm. **4** (1968), No. 2, 1–7].

- [40] WHITE, H. S.—COLE, F. N.—CUMMINGS, L. D.: *Complete classification of triad systems on fifteen elements*, Memoirs Natl. Acad. Sci. USA. **27** (1919) No. 2, 1–89.

Received 23. 8. 2007

**Department of Computer Science
Helsinki Institute for Information Technology HIIT
University of Helsinki
P.O. Box 68, 00014 University of Helsinki
FINLAND
E-mail: petteri.kaski@cs.helsinki.fi*

***Department of Electrical and Communications Engineering
Helsinki University of Technology
P.O. Box 3000, 02015 TKK
FINLAND
E-mail: patric.ostergard@tkk.fi*