Revista digital
Matemática, Educación e Internet
https://revistas.tec.ac.cr/index.php/matematica
Vol 27, No 1. Agosto, 2026 – Febrero, 2027

ISSN 1659 -0643
DOI: XXX

Construction of Balanced Incomplete Block Designs

| Construcción de Diseños de Bloques Incompletos Balanceados |

| Construção de Desenhos de Blocos Incompletos Balanceados |

Eduardo Piza Volio
eduardo.piza@ucr.ac.cr
Universidad de Costa Rica
San José, Costa Rica

Esteban Segura Ugalde
esteban.seguraugalde@ucr.ac.cr
Universidad de Costa Rica
San José, Costa Rica

Recibido: 29 enero 2025 Aceptado: 30 mayo 2026

Abstract: In this review article, we study balanced incomplete block designs (BIBDs) and test a heuristic algorithm for generating these designs. We first provide a brief theoretical exposition of BIBDs, the mathematical properties of which are of extreme importance due to their deep theoretical applications in non-Euclidean geometry (construction of finite projective and finite affine planes) and in combinatorics. We use a simulated annealing algorithm to find BIBDs of moderate size. In particular, we discover a new, unpublished solution for the BIBD(16,56,21,6,7), for which only one other solution was known, even though the theoretical existence of more solutions was certain. Our algorithms were compared with other heuristic methods used in previous research and consistently outperformed all other heuristic methods.

Keywords: balanced incomplete blocks designs, combinatorics, simulated annealing.

Resumen: En este artículo estudiamos los *diseños de bloques balanceados incompletos* (BIBDs) y ponemos a prueba un algorítmo heurístico para generar estos diseños. Realizamos primero una breve exposición teórica de los BIBDs, cuyas propiedades matemáticas son de extrema importancia por sus profundas aplicaciones teóricas en el campo de la geometría no euclideana (construcción de planos proyectivos finitos y planos afines finitos) y en el campo de la combinatoria. Utilizamos heurísticas de *recocido simulado* para hallar BIBDs de un tamaño moderado. En particular hallamos una nueva solución inédita para el BIBD(16,56,21,6,7), para el cual solamente se conocía otra solución, aunque se sabía con certeza de la existencia teórica de más soluciones. Nuestros algoritmos fueron comparados contra otros métodos heurísticos empleados en anteriores investigaciones, superando ampliamente a todos los otros métodos heurísticos.

Palabras Clave: diseños de bloques incompletos balanceados, combinatoria, recocido simulado.

[1]Researcher at the Center for Research in Pure and Applied Mathematics (CIMPA), University of Costa Rica. Research City, San Pedro de Montes de Oca, San José, Costa Rica. Postal code: 2060 San José. E-mail: eduardo.piza@ucr.ac.cr.
[2]Researcher at the Center for Research in Pure and Applied Mathematics (CIMPA), University of Costa Rica. Research City, San Pedro de Montes de Oca, San José, Costa Rica. Postal code: 2060 San José. E-mail: esteban.seguraugalde@ucr.ac.cr.

Resumo: Neste artigo de revisão, estudamos os desenhos de blocos incompletos balanceados (BIBDs) e testamos um algoritmo heurístico para gerar esses desenhos. Inicialmente, apresentamos uma breve exposição teórica sobre os BIBDs, cujas propriedades matemáticas são de extrema importância devido às suas profundas aplicações teóricas na geometria não euclidiana, especialmente na construção de planos projetivos finitos e planos afins finitos, bem como na combinatória. Utilizamos um algoritmo de recozimento simulado para encontrar BIBDs de tamanho moderado. Em particular, descobrimos uma nova solução, ainda não publicada, para o BIBD(16,56,21,6,7), para o qual apenas uma outra solução era conhecida, embora a existência teórica de mais soluções fosse certa. Nossos algoritmos foram comparados com outros métodos heurísticos utilizados em pesquisas anteriores e superaram consistentemente todos esses métodos.

Palavras-chave: desenhos de blocos incompletos balanceados; combinatória; recozimento simulado.

1. Balanced Incomplete Block Designs

One of the key notions in the theory of combinatorial designs is the concept of pairwise equilibrium, which finds its roots in the theory of statistical designs of experiments. When combined with the requirements of a specific type of regularity, it gives raise to one of the most common forms of combinatorial designs. There are excellent introductory references to balanced incomplete block designs, and among them, we recommend the works of Gross (2007), Godsil (2019) and Rosa (1987).

In this presentation, we introduce some elementary concepts of the theory and also contribute a rather modest addition to the topic.

Definición 1

A balanced incomplete block design (BIBD) with parameters (v, b, r, k, λ) is an ordered pair $(V, \mathcal{B})$, where V is a finite set of v elements or points, $\mathcal{B}$ is a family of b subsets of V, called blocks, each of cardinality k, such that every point of V is contained in exactly r blocks, and every two distinct points of V are contained in exactly λ blocks.

When $k = v$, we refer to the blocks as *complete*. In this scenario, every block encompasses all elements of V, resulting in equal blocks. Conversely, in the general case, where $k < v$, we characterize the blocks of the combinatorial design as *incomplete*.

In general, the five parameters v, b, r, k, λ are not independent. In fact, a simple counting argument provide us with two basic relations:

$$vr = bk, \tag{1}$$
$$\lambda(v-1) = r(k-1). \tag{2}$$

BIBDs are sometimes denoted as BIBD(v, k, λ), reducing the number of parameters from 5 to 3, as the other two parameters, b and r, can be derived from the above equations. In fact, solving equations (1) and (2) for b and r, we obtain $b = \dfrac{\lambda v(v-1)}{k(k-1)}$ and $r = \dfrac{\lambda(v-1)}{k-1}$. Since both r and b are integers, we derive the following necessary condition (3) (and its direct consequence (4)) for the existence of a BIBD(v, b, r, k, λ):

$$\lambda(v-1) \equiv 0 \mod(k-1), \tag{3}$$
$$\lambda v(v-1) \equiv 0 \mod(k-1). \tag{4}$$

If $(V, \mathcal{B})$ is a BIBD(v, b, r, k, λ), with points $V = \{x_i : 1 \leq i \leq v\}$ and blocks $B = \{B_j : 1 \leq j \leq b\}$, then we denote $M = \{m_{ij}\}$ the *incidence matrix* of the BIBD as that binary matriz of size $v \times b$, where

$$m_{ij} = \begin{cases} 1, & \text{if } x_i \in B_j, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

It is easily verified (see Anderson y Honkala (2012)) that the incidence matrix always satisfies the relation $MM^t = (r - \lambda)I_v + \lambda J$, where I_v is the identity matrix of size v and J is the $v \times v$ matrix of 1's. That is, in the MM^t matrix, every element on the diagonal is r and every element off the diagonal is λ.

We say that a BIBD is *symmetric* when $v = b$ (and therefore $r = k$).

Example 1. An interesting symmetric BIBD(7,7,3,3,1) is the famous *Fano plane*, i.e. the *projective plane*[1] of order 2, as shown in Figure 1, where $V = \{1, 2, \ldots, 7\}$ are the *points* of the plane and the blocks (the *lines* of the plane) are[2]:

132 145 167 246 257 347 356. □

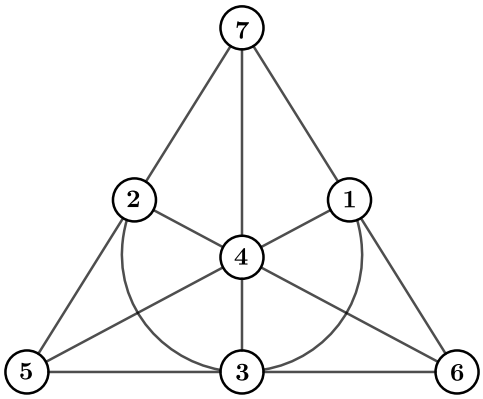

Figura 1: The Fano plane. Source: Prepared by the authors.

The BIBD incidence matrix corresponding to the Fano plane is as follows:

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}, \tag{6}$$

a 7×7 binary matrix whose rows all add up to $r = 3$, its columns all add up to $k = 3$, and the inner product between two different rows always equals $\lambda = 1$. From a computational point of view, it is easiest to conceptualize a BIBD by its incidence matrix $M = \{m_{ij}\}$.

[1] A *finite projective plane of order* n, where $n > 0$, is a collection of $n^2 + n + 1$ lines and $n^2 + n + 1$ points satisfying the following axioms: *i*) Every line contains $n + 1$ points; *ii*) Every point lies on $n + 1$ lines; *iii*) Two different lines always intersect at 1 point; *iv*) Two different points always lie on a line. One of the interesting aspects of finite projective planes is that the notions of "lines" and "points" can be interchanged, obtaining basically the same 4 axioms.

[2] See Gross (2007) y Kalmanovich (2005) for an introduction to finite projective planes.

Example 2. The following is a symmetric BIBD$(15, 15, 7, 7, 3)$, where the elements are $V = \{0, 1, \ldots, 14\}$, with the following 15 blocks:

$$
\begin{array}{lll}
B_0 = \{0,1,2,3,4,5,6\} & B_1 = \{0,1,2,7,8,9,10\} & B_2 = \{0,1,2,11,12,13,14\} \\
B_3 = \{0,3,4,7,8,11,12\} & B_4 = \{0,3,4,9,10,13,14\} & B_5 = \{0,5,6,7,8,13,14\} \\
B_6 = \{0,5,6,9,10,11,12\} & B_7 = \{1,3,5,7,9,11,13\} & B_8 = \{1,3,6,7,10,12,14\} \\
B_9 = \{1,4,5,8,10,11,14\} & B_{10} = \{1,4,6,8,9,12,13\} & B_{11} = \{2,3,5,8,10,12,13\} \\
B_{12} = \{2,3,6,8,9,11,14\} & B_{13} = \{2,4,5,7,9,12,14\} & B_{14} = \{2,4,6,7,10,11,13\}.
\end{array}
$$

We can number the BIBD blocks with subscripts in any convenient way, starting the numbering at 0, at 1, or at any other number. □

Example 3. The following BIBD$(9, 12, 4, 3, 1)$ is also very famous: it is the *3rd order affine plane*, as shown in Figure 2. Here we have $V = \{1, 2, \ldots, 9\}$ (points of the affine plane) and the blocks (lines of the affine plane):

123 456 789 147 158 169 248 259 267 349 357 368. □

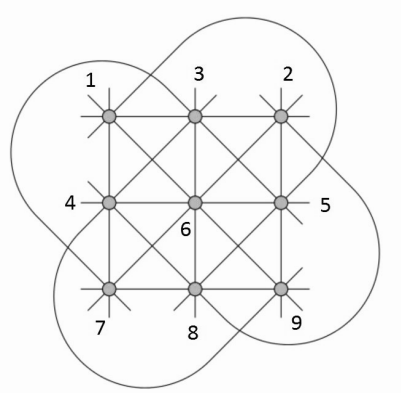

Figura 2: The affine plane of order 3. Source: Prepared by the authors.

The BIBDs in Examples 1 and 2 are symmetric, while the BIBD in Example 3 is not. In a symmetric BIBD, any two blocks have exactly λ elements in common.

Example 4. The *3rd order projective plane* is a symmetric BIBD(13,13,4,4,1) as shown in Figure 3. Here the "points" are $V = \{2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A\}$, like the numbers and figures on the cards, while the blocks ("lines") are:

2 3 4 J	2 5 8 Q	2 6 10 A	2 7 9 K	3 5 10 K	3 6 9 Q	3 7 8 A
4 5 9 A	4 6 8 K	4 7 10 Q	5 6 7 J	8 9 10 J	J Q K A	

□

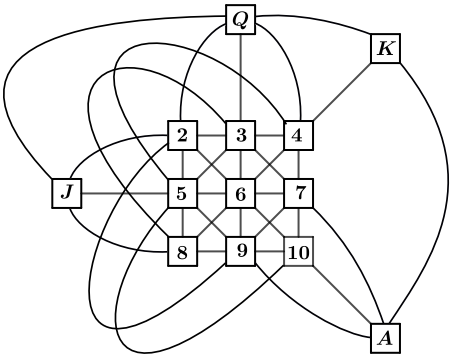

Figura 3: The projective plane of order 3. Source: Prepared by the authors.

Given $(V, \mathcal{B})$, a symmetric BIBD(v, b, r, k, λ) with blocks

$$\mathcal{B} = \{B_0, B_1, \ldots, B_{b-1}\},$$

we can fix one of the blocks, for example the block B_0, and obtain the new $b-1$ blocks B_i^* of the *derived design with respect to* B_0, denoted Der$(V, \mathcal{B}; B_0)$, where $B_i^* = B_0 \cap B_i$. The parameters $(v^*, b^*, r^*, k^*, \lambda^*)$ of the derived design with respect to the original design are $v^* = k$, $b^* = b - 1$, $r^* = r - 1$, $k^* = \lambda$, $\lambda^* = \lambda - 1$.

Example 5. Starting with the symmetric design of Example 2, we get a *derived design* with respect to B_0, which gives us a BIBD$(7, 14, 6, 3, 2)$, which is *non-symmetric*. This is

$$\begin{array}{lllll} B_1^* = \{0,1,2\} & B_2^* = \{0,1,2\} & B_3^* = \{0,3,4\} & B_4^* = \{0,3,4\} & B_5^* = \{0,5,6\} \\ B_6^* = \{0,5,6\} & B_7^* = \{1,3,5\} & B_8^* = \{1,3,6\} & B_9^* = \{1,4,5\} & B_{10}^* = \{1,4,6\} \\ B_{11}^* = \{2,3,5\} & B_{12}^* = \{2,3,6\} & B_{13}^* = \{2,4,5\} & B_{14}^* = \{2,4,6\} & \end{array}$$

□

Note that in the above design, three blocks are repeated once. A BIBD that has no repeated blocks is called a *simple* combinatorial design. The above BIBD is not simple.

The same symmetric design of Example 2 yields a *derived design* with respect to B_{14} that is also a *non-symmetric*, but *simple* BIBD$(7, 14, 6, 3, 2)$, essentially different from the previous one:

$$\begin{array}{lllll} B_0^* = \{2,4,6\} & B_1^* = \{2,7,10\} & B_2^* = \{2,11,13\} & B_3^* = \{4,7,11\} & B_4^* = \{4,10,13\} \\ B_5^* = \{6,7,13\} & B_6^* = \{6,10,11\} & B_7^* = \{7,11,13\} & B_8^* = \{6,7,10\} & B_9^* = \{4,10,11\} \\ B_{10}^* = \{4,6,13\} & B_{11}^* = \{2,10,13\} & B_{12}^* = \{2,6,11\} & B_{13}^* = \{2,4,7\} & \end{array}$$

Given $(V, \mathcal{B})$ a symmetric BIBD(v, b, r, k, λ), if we fix any block B_0 and suppress that block and its elements in all other blocks, we obtain a *residual* combinatorial design BIBD$(v', b', r', k', \lambda')$, denoted Res$(V, \mathcal{B}; B_0)$, whose parameters are $v' = v - k$, $b' = b - 1$, $r' = r$, $k' = k - \lambda$, $\lambda' = \lambda$.

Example 6. From the symmetric design of Example 2, by removing the block $B_1 : 0123456$, we obtain the following *residual* combinatorial design BIBD(8, 14, 7, 4, 3), which is obviously *non-symmetric*:

$$\begin{array}{llll} B_1^* = \{7,8,9,10\} & B_2^* = \{11,12,13,14\} & B_3^* = \{7,8,11,12\} & B_4^* = \{9,10,13,14\} \\ B_5^* = \{7,8,13,14\} & B_6^* = \{9,10,11,12\} & B_7^* = \{7,9,11,13\} & B_8^* = \{7,10,12,14\} \\ B_9^* = \{8,10,11,14\} & B_{10}^* = \{8,9,12,13\} & B_{11}^* = \{8,10,12,13\} & B_{12}^* = \{8,9,11,14\} \\ B_{13}^* = \{7,9,12,14\} & B_{14}^* = \{7,10,11,13\}. & & \end{array}$$

□

2. Fisher's inequality

Another necessary condition for the existence of a BIBD(v, b, r, k, λ) is given by the following result, due to Fisher:

Teorema 1

(Fisher's inequality, 1940) For any BIBD(v, b, r, k, λ) we will have

$$b \geq v. \tag{7}$$

Demonstration: Following Fisher (1940), we evaluate the determinant of AA', where A' denotes the transpose of A:

$$\det AA' = (r-\lambda)^{v-1}(v\lambda - \lambda + r).$$

To perform this calculation, we subtract the first column in AA' from all remaining columns and then add the rows 2 through v to the first row. As a result, all elements above the main diagonal cancel out. On the main diagonal of AA', the first entry will be $r + (v-1)\lambda$, while the remaining entries on the diagonal will be $r - \lambda$.

We have that $r > \lambda$, since $r = \lambda$ would imply that each element is paired with each other when it is part of a block, i.e., each block would contain all v elements. Therefore, AA' is nonsingular and A has rank at most equal to b. Now, AA' has rank v, but the rank of the product of matrices cannot exceed the rank of their factors, whence we obtain that $b \geq v$ (which further implies that $r \geq k$).

■

An example of an application of Fisher's inequality is that a combinatorial design BIBD(21,6,1) cannot exist, since $b = \dfrac{\lambda v(v-1)}{k(k-1)} = \dfrac{1 \cdot 21 \cdot 20}{6 \cdot 5} = 14 < v = 21$, violating Fisher's inequality. Note that in this case the other two necessary conditions (3) and (4) for the existence of combinatorial design are satisfied.

3. Concatenation of compatible BIBDs

In his paper, Fisher (1940) also noted that if we have the designs of one BIBD$(v, b_1, r_1, k, \lambda_1)$ and another BIBD$(v, b_2, r_2, k, \lambda_2)$, then both designs can be combined by juxtaposing them (juxtaposing their incidence matrices, since both have the same number v of rows) to obtain a larger BIBD design with parameters $(v, b_1 + b_2, r_1 + r_2, k, \lambda_1 + \lambda_2)$. This operation is called juxtaposing BIBDs and is denoted by

$$\text{BIBD}(v, b_1, r_1, k, \lambda_1) \,||\, \text{BIBD}(v, b_2, r_2, k, \lambda_2) \rightarrowtail \text{BIBD}(v, b_1 + b_2, r_1 + r_2, k, \lambda_1 + \lambda_2).$$

Such source BIBDs are called *compatible*, as illustrated in Table 1.

BIBD(8,14,7,4,3)	BIBD(8,28,14,4,6)	BIBD(8,42,21,4,9)
00001101100111	01100110010111001001101001 01	00001101100111011001100101110010011010 0101
11011101001000	00001110101111111011011000000	11011101001000000011101011111011011000000
00010110111100	00100011111010000101010111 10	00010110111100001000111110100001010101111 0
10100001011110	11110101001000101110100011 00	10100001011110111101010010001011101000110 0
11010000110011	01010010000001011110011110 11	11010000110011010100100000010111100111101 1
01100111010001	10111001010110010010101100 10	01100111010001101110010101100100101011001 0
00111010001011	10011001100101101001010101 01	00111010001011100110011001011010001010101
11101010100100	11001100111000110001001010 11	11101010100100110011001110001100010010101 1
8 x 14 matrix row sum $r = 7$ col sum $k = 4$ $\lambda = 3$	8 x 28 matrix row sum $r = 14$ col sum $k = 4$ $\lambda = 6$	8 x 42 matrix row sum $r = 21$ col sum $k = 4$ $\lambda = 9$

Tabla 1: BIBD(8,14,7,4,3) —— BIBD(8,28,14,4,6) $\rightarrowtail$ BIBD(8,42,21,4,9). Note also that the design of BIBD(8,28,14,4,6) can be obtained by juxtaposing BIBD(8,14, 7,4,3) with itself. Then BIBD(8,14,7,4,3) —— BIBD(8,14,7,4,3) —— BIBD(8,14,7,4,3) $\rightarrowtail$ BIBD(8,42,21,4,9). In general, in this way we could construct the designs corresponding to any BIBD(8,$14n$,7n,4,3n), for any $n \in \mathbb{N}^*$. Source: Prepared by the authors.

4. Bruck-Ryser-Chowla theorem

Another necessary condition for the existence of *symmetric* combinatorial designs is given by the following theorem:

Teorema 2

(Bruck-Ryser-Chowla, 1949–1950) In a symmetric combinatorial design BIBD(v, b, r, k, λ):

a) If v is even, then $k - \lambda$ is a square.

b) If v is odd, then the equation $z^2 = (k-\lambda)x^2 + (-1)^{(v-1)/2}\lambda y^2$ has a solution in the integers x, y, z, not all of which are zero.

Demonstration: (a) In fact, in a symmetric combinatorial design, we have that $b = v$, so A is a square matrix. Then,

$$(\det A)^2 = \det AA' = (k-\lambda)^{v-1}(v\lambda - \lambda + k).$$

Since $k(k-1) = \lambda(v-1)$, we have that $v\lambda - \lambda + k = k(k-1) + k = k^2$. Then the other factor of $\det AA'$, namely $(k-\lambda)^{v-1}$, must also be a square. Since v is even, this implies that $k - \lambda$ must also be a square.

(b) It is complicated. See Bruck y Ryser (1949), Chowla y Ryser (1950) y Kalmanovich (2005).

■

Example 7. There is no symmetric combinatorial design BIBD$(22, 22, 7, 7, 2)$, since in this case $v = 22$ is even, and the Bruck-Ryser-Chowla theorem says that $k - \lambda$ must be a square, which is not satisfied in this case, since $k - \lambda = 5$. Note that in this example the first necessary conditions (3) and (4) are satisfied. □

Example 8. A symmetric BIBD$(43, 43, 7, 7, 1)$, i.e. a model of the projective plane of order 6, cannot exist (see Kalmanovich (2005)), since the condition of the Bruck-Ryser-Chowla theorem reduces here to the equation $z^2 + y^2 = 6x^2$, which has no solution for integers x, y, z not all equal to zero. In fact, suppose that the equation $z^2 + y^2 = 6x^2$ has a non-trivial solution. Let $x \geq 0$, $y \geq 0$, $z \geq 0$ be the nontrivial solution *whose sum* $x + y + z > 0$ *is minimal*. Obviously, y and z cannot have distinct parity. So we have two possibilities:

a) If both y and z components were odd, we would have $y^2 \equiv 1 \pmod 8$ and $z^2 \equiv 1 \pmod 8$, giving $z^2 + y^2 \equiv 2 \pmod 8$. However, $6x^2 \equiv 6 \pmod 8$ or $6x^2 \equiv 0 \pmod 8$, depending on whether x is odd or even. This leads to a contradiction.

b) If both components y and z were even, then we would have that $z^2 + y^2$ is a multiple of 4, from which also $6x^2$ must be a multiple of 4. This would imply that x must be even. Then the three components x, y, and z of the solution will be even, say $x = 2x'$, $y = 2y'$, $z = 2z'$. But then, dividing by 2, we would get that x', y', z' are also a solution of the original equation, with $x' + y' + z' < x + y + z$, which is also a contradiction.

Therefore, it is concluded that the equation $z^2 + y^2 = 6x^2$ has no non-trivial solution. □

Example 9. In general, when $\lambda = 1$, the Bruck-Ryser-Chowla theorem gives the following necessary condition for the existence of a *symmetric* BIBD$(v, v, k, k, 1)$, corresponding to a model of a projective plane of order $n = k - 1$. The necessary condition is that if $n \equiv 1, 2 \pmod 4$, there exist integers x, y such that $n = x^2 + y^2$. This implies that for infinitely many orders n the projective plane of order n cannot exist, as in the cases $n = 6, 14, 21, 22, \ldots$ The smallest order not covered by this theorem, and which is also not a prime number, is $n = 10$, corresponding to the assumed existence of a symmetric

BIBD$(111, 111, 11, 11, 1)$. Well, using computational search with brute force algorithms, Lam (1991) proved the non-existence of the projective plane of order 10, after about 9 years of calculations. □

All the necessary conditions described above are still not sufficient to guarantee the existence of certain combinatorial designs. In fact, the parameter sets $(v, b, r, k, \lambda) = (22, 33, 12, 8, 4)$ or (46, 69, 9, 6, 1) or (111, 111, 11, 11, 1) satisfy the *necessary conditions* studied above. However, in each of these cases it has been shown by detailed structural analysis combined with considerable exhaustive computational search that the corresponding BIBD does not exist at all. The search for (necessary and sufficient) conditions for the existence of BIBDs continues. Currently, the "smallest" parameter sets for which the existence of an associated BIBD is still uncertain are (51,85,10,6,1), (61,122,12,6,1), (40,52,13,10,3), and (85,170,14,7,1). There are extensive parameter tables of "small" BIBDs (e.g., those in which $r \leq 41$) that record, for each parameter set, whether the combinatorial design exists, does not exist, or whether its existence is still an open question, as well as miscellaneous information including enumerative results (see Colbourn y Dinitz (2006)).

5. Resolvable BIBD's

We say that a BIBD is *resolvable* if its blocks can be partitioned into subsets $R_1, \ldots, R_r$, called *parallel classes*, where each R_i consists of pairwise disjoint blocks whose union is equal to the set of all V elements.

For example, the BIBD(9,12,4,3,1) of example 3 is resolvable. Indeed, the 12 blocks can be partitioned into the following 4 subsets of 3 disjoint blocks each, whose union is equal to V: $R_1 = \{123, 456, 789\}$, $R_2 = \{147, 259, 368\}$, $R_3 = \{158, 267, 349\}$, $R_4 = \{169, 248, 357\}$.

We will usually be interested in BIBDs with $k \geq 3$ blocks, since a BIBD with only $k = 2$ blocks has the structure of a complete (or multi-complete) graph. In fact, a resolvable BIBD with $k = 2$ and $\lambda = 1$ is equivalent to the 1-factorization of a complete graph, and it is known that such a 1-factorization exists if and only if the number of elements (i.e., vertices of the complete graph) is even.

Given two BIBDs $(V, \mathcal{B})$ and $(W, \mathcal{C})$, a function $\alpha : V \mapsto W$ such that $\alpha V = W$ and $\alpha\mathcal{B} = \mathcal{C}$, is said to be a *isomorphism*: the two BIBDs are *isomorphic*. An isomorphism between a combinatorial design and itself is called a *automorphism*. The set of all automorphisms of $(V, \mathcal{B})$ forms under the composition of functions a group called the *complete group of automorphisms of* $(V, \mathcal{B})$. Each of its subgroups is a *group of automorphisms* of $(V, \mathcal{B})$.

As an example without proof, it is known that in the *Fano plane* (see Example 1) all BIBD(7,3,1) are isomorphic to each other and the order of the complete group of automorphisms of the Fano plane is 168 (see Hirschfeld (1998)).

Group theory and the existence of combinatorial designs are closely related, since a method often used to prove the existence of a design is to assume the existence of some other design with a particular automorphism group. This allows one to choose a small basic set of blocks (the *base*) that contains representatives of all block orbits in the group in question. The collection $\mathcal{B}$ of all blocks is then obtained by allowing the group to act on these base blocks.

For example, we say that a design-(v, k, λ) is *cyclic* if it admits a cyclic group of order v within its automorphism group. Alternatively, we say that it is cyclic if it admits an automorphism that permutes its elements in a simple cycle of length v. The elements of a cyclic design are usually identified as elements of $\mathbb{Z}_v$, with $\alpha : i \mapsto i + 1$ as the cyclic automorphism.

6. Steiner systems

Other more general types of balanced combinatorial designs are the so-called t-designs, which we define below.

Definición 2

For an integer $t \geq 2$, a t-combinatorial design t-(v, k, λ) is an ordered pair $(V, \mathcal{B})$, where V is a set of v *elements* and $\mathcal{B}$ is a collection of subsets of V called *blocks*, all of k elements, such that each subset of V with t elements is contained in exactly λ blocks.

A t-(v, k, λ) combinatorial design, where $\lambda = 1$, is called a *Steiner system* and is denoted by $S(t, k, v)$. A Steiner system $S(2, 3, v)$ is called a *Steiner system of triples*, while Steiner systems $S(3, 4, v)$ are called *Steiner systems of quadruples*.

Note that BIBDs are 2-designs and BIBDs with $\lambda = 1$ are Steiner systems $S(2, k, v)$. Currently, only a finite number of Steiner systems $S(t, k, v)$ with $t = 4$ or $t = 5$ are known, and none for $t \geq 6$. Among the best known Steiner systems for $t > 2$, there are the five so-called *Mathieu groups* M_{11}, M_{12}, M_{22}, M_{23}, M_{24}, which are actually$S(4, 5, 11)$, $S(5, 6, 12)$, $S(3, 6, 22)$, $S(4, 7, 23)$ and $S(5, 8, 24)$, respectively (see Rosa (1987)).

7. BIBDs and Hadamard matrices

In this section we will see a theoretical connection between Hadamard matrices and certain types of symmetric BIBDs. Recall that a Hadamard matrix H of order n is a matrix with entries in $\{-1, +1\}$, of size $n \times n$, such that $HH^t = nI_n$.

It is easy to show that if H is a Hadamard matrix of order n, then $n \in \{1, 2, 4m\}$, $m \in \mathbb{N}^*$. The converse of this property is the famous *Hadamard Conjecture*, still an open problem in mathematics after more than 130 years. The conjecture states that "for any positive multiple of 4, say 4m, there exists a Hadamard matrix of order $4m$". The Hadamard conjecture is far from being solved. The smallest positive multiple of 4 for which it is not yet known whether a Hadamard matrix of such order exists is $668 = 4 \cdot 167$ (see Piza (2011)).

Teorema 3

Suppose that $m > 1$. Then, there exists a Hadamard matrix of order $4m$ if and only if there exists a symmetric BIBD$(4m - 1, 2m - 1, m - 1)$.

The formal proof can be found in Stinson (2003). Basically, it consists in proving that if W is the incidence matrix of a symmetric BIBD$(4m - 1, 2m - 1, m - 1)$, then the matrix

$$\widetilde{H} = \left[\begin{array}{c|c} 1 & 1 \cdots 1 \\ \hline \vdots & B \\ 1 & \end{array}\right], \tag{8}$$

where B comes from the matrix W by exchanging each 0 for -1, is a Hadamard matrix of order $4m$. Conversely, if H is a Hadamard matrix of order $4m$, we can normalize it (by multiplying rows and columns by -1 and exchanging some rows and columns) so that the resulting matrix $\widetilde{H}$ is also a Hadamard matrix as in (8). Then in B we replace each -1 by a 0, and we get the incidence matrix W of a symmetric BIBD$(4m - 1, 2m - 1, m - 1)$.

For example, starting from the Fano plane, which is a symmetric BIBD(7,3,1) (here $m = 2$) whose incidence matrix W was illustrated in (6), we can find the following Hadamard matrix $\widetilde{H}$ of order 8 using this method:

$$W = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \Rightarrow \widetilde{H} = \left[\begin{array}{c|ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{array}\right].$$

This method is only of theoretical interest, as it is not very practical for obtaining Hadamard matrices or symmetric BIBDs of parameters $(4m-1, 2m-1, m-1)$. There are more efficient methods in practice, at least for cases where it is known that the problem has a solution (see Piza (2011)).

8. Construction of BIBDs using heuristics

The computational construction of balanced incomplete block designs is in general a well-known *NP-hard* problem (see Corneil y Mathon (1978)), providing an excellent benchmark. BIBDs exhibit a wide variaty of instances, ranging from the most elementary to the most challenging, making them a suitable environment for testing different techniques and algorithms. The scalability and difficulty of the problem make it an ideal candidate for evaluating the performance of various methods.

Historically, exhaustive search (brute-force) has been the most commonly used approach, but it becomes intractable even for relatively small design sizes (see Gibbons y Ostergård (1996)). The difficulty of the problem is illustrated by the fact that several instances of finding a BIBD remain unsolved, even for relatively small parameter values. This will be discussed in more detail in section 10. It is important to note that there might be cases where no solution exists. Given that the algorithmic generation of BIBDs is a general *NP-hard* problem, brute-force methods are inherently limited by the problem size. Hence, the use of metaheuristic algorithms appears more suitable for handling larger instances.

Various metaheuristic methods, including hill climbing (HC), tabu search (TS), genetic algorithms (GA) and neural network (NN) algorithms, have been employed to generate BIBDs with some relative success. Extensive comparisons of these methods have been conducted (see Bofill et al. (2003) y Prestwich (2003)), with genetic algorithms (GA) proving to be the best choice so far, prior to the introduction of our *simulated annealing* algorithm (SA).

9. Simulated annealing algorithms and BIBDs

In this section we describe a *simulated annealing* algorithm (see Aarts y Korst (1990)) to generate a balanced incomplete block design BIBD(v, b, r, k, λ), with parameters of size "small" ($vb \leq 1000$). The algorithm is as follows:

1. Given the parameters (v, b, r, k, λ) of the BIBD to be generated, we first analyze whether they satisfy the necessary conditions for the existence of the corresponding BIBD, known in the literature on the subject, which are:

   a) Equation (1): $vr = kb$.

b) Equation (2): $\lambda(v-1) = r(k-1)$.

c) Fisher's inequality (7): $b \geq v$.

d) If the BIBD is symmetric ($v = b$ y $r = k$), we have to check that the conditions of the theorem 4 of Bruck-Ryser-Chowla theorem 2 are fulfilled, i.e: (*i*) when v is even, then $k-\lambda$ must be a perfect square; (*ii*) when v is odd, then it must be satisfied that the equation $z^2 = (k-\lambda)\,x^2 + (-1)^{(v-1)/2}\,\lambda y^2$ has solution in the integers x, y, z, not all zero. **Note**: the verification of this last condition is not easy, so it is established by hand beforehand.

2. We randomly generate an initial incidence matrix M_{curr} of size $v \times b$, with binary entries in $\{0, 1\}$, where each row adds up to r (i.e., each row contains exactly r 1's). To generate it, we start with $[\mathbf{1}_{v\times r}|\mathbf{0}_{v\times(b-r)}]$, where $\mathbf{1}_{v\times r}$ is the 1's matrix. We then apply a random permutation within each row, using the Fisher-Yates algorithm (see Press et al. (2007)), to obtain our initial matrix M_{curr}. This matrix will evolve over the course of the algorithm into a BIBD(v, b, r, k, λ) incidence matrix, as in (5), should the algorithm become successful.

3. During the course of our algorithm, we will move through a *neighborhood structure* toward an incidence matrix of the BIBD(v, b, r, k, λ) under study. We define the neighborhood $\mathcal{V}(M_{\text{curr}})$ as all those matrices that can be obtained by swapping two entries of opposite signs of the *same row* of M_{curr}. Then the size of each neighborhood will always be $|\mathcal{V}(M_{\text{curr}})| = vr(b-r)$, since we will have v rows available to select one of them, and in that selected row we will have r possibilities to select a 1 and $b-r$ possibilities to select a 0. In the course of the algorithm, we will eventually move to some neighborhood $M_{\text{prop}} \in \mathcal{V}(M_{\text{curr}})$. In principle, all neighbors have an equal chance of being considered for a move to them, although the final decision of whether or not to accept a neighbor is determined by the so-called *Metropolis Rule*, which we will explain later.

4. We define the following objective function $f(M)$, which measures how far a binary matrix M of size $v \times b$, of rows with r 1's, is from actually being an incidence matrix for the BIBD(v, b, r, k, λ) we are looking for, i.e, where all rows must sum to r, all columns must sum to k, and the scalar product between two different rows must be λ.

   We first define the quantities $\phi_j(M) = \sum_{i=1}^{v} m_{ij}$, for $j \in \{1, \ldots, b\}$, which denotes the quantity of 1's in the j-th column of M. We also define the quantities $\varphi_{ij}(M) = \sum_{s=1}^{b} m_{is}m_{js}$, with $1 \leq i < j \leq v$, denoting the scalar product between the i-th and j-th rows of M. Then we define the objective function $f(M)$ using

$$f(M) = \sum_{j=1}^{b} |\,\phi_j(M) - k\,| + \sum_{i=1}^{v-1}\sum_{j=i+1}^{v} |\,\varphi_{ij}(M) - \lambda\,|.$$

   In this way the global minimum of the function f will have the value 0 and will be reached in an incidence matrix for the BIBD(v, b, r, k, λ), if it exists, which is exactly what we are looking for.

5. We compute and store in memory the quantities $\phi_j(M_{\text{curr}})$ and the quantities $\varphi_{ij}(M_{\text{curr}})$. Here $j \in \{1, \ldots, b\}$ and $1 \leq i < j \leq v$. We compute $f(M_{\text{curr}})$.

6. They are initialized: $M_{\text{best}} \leftarrow M_{\text{curr}}$ and $f(M_{\text{best}}) \leftarrow f(M_{\text{curr}})$. The matrix M_{best} will be the "best" matrix found by the algorithm at any given time.

7. We calculate the initial temperature t_0. For this purpose we ran nSim *dummy simulations* of random moves of M_{curr} toward a neighbor $M_{\text{prop}} \in \mathcal{V}(M_{\text{curr}})$ to calculate the average Δf_{mean} change in the objective function, for those moves that do not improve the criterion, i.e., that increase the value of the objective function. Empirically, we have successfully experimented with the value $\text{nSim} = 400$.

   Let $0 < \chi < 1$ be the *acceptance threshold* for bad moves, previously set by the user. We then compute

$$t_0 = \frac{-\Delta f_{\text{mean}}}{\ln \chi}.$$

Thus, the algorithm will initially accept about $100\,\%\chi$ of moves that degrade the optimization criterion. Empirically we often choose a value of $\chi = 0.6$ (i.e. we start with an *acceptance threshold* of 60 %).

8. $\mathrm{nTe} \leftarrow 0$, $\mathrm{nChain} \leftarrow 0$, $t \leftarrow t_0$, $\mathrm{WereMov} \leftarrow \mathrm{False}$. Here, nTe controls the steps of the temperature parameter t, while $\mathrm{WereMov}$ controls whether for this temperature t, M_{curr} moves were made to $M_{\mathrm{prop}} \in \mathcal{V}(M_{\mathrm{curr}})$. The counter nChain counts consecutive chains without movement.

9. Do while ($\mathrm{nTe} \leq \mathrm{nTemp}$ and $\mathrm{nChain} \leq \mathrm{maxChain}$):

   This is the cycle of the temperature parameter, which will keep changing following an *exponential-type cooling scheme*, i.e., $t \leftarrow t * \mathrm{tFactor}$, where $\mathrm{tFactor} < 1$ is a constant close to 1. Empirically we have successfully used values of $\mathrm{tFactor} \in [0.94, 0.98]$. A maximum of nTemp temperatures are used. An early termination criterion of the algorithm is also applied, which consists of checking if in the last consecutive $\mathrm{maxChain}$ Markov chains a new neighbor $M_{\mathrm{prop}} \in \mathcal{V}(M_{\mathrm{curr}})$ (improving or worsening the criterion) has not been accepted, in which case the algorithm terminates prematurely, assuming that improvements are unlikely to be found in the future. We have successfully used empirical values of $\mathrm{nTemp} = 300$ and $\mathrm{nChain} = 3$.

   9a. $\mathrm{nStep} \leftarrow 1$, $\mathrm{nBad} \leftarrow 0$.

   9b. Do While ($\mathrm{nStep} \leq \mathrm{nOver}$ and $\mathrm{nBad} \leq \mathrm{nBadMax}$):

   This is the cycle of the iterations of the Markov chain corresponding to the current temperature t. This cycle is usually long, because the goal is for the Markov chain to reach a certain equilibrium. The chain ends after nOver possible moves. However, if a maximum of $\mathrm{nBadMax}$ moves have already been made that worsen the criterion (i.e., increase the value of the objective function), then the chain is interrupted and continues to the next temperature. The cycle will also be interrupted in the lucky case of finding the global minimum, i.e. when M_{best} is such that $f(M_{\mathrm{best}}) = 0$. nOver is usually on the order of millions, while $\mathrm{nBadMax}$ is empirically set as 10 % of nOver.

   9b*i*. We randomly select $i_0 \in \{1, \ldots, v\}$. This is the index of the row of M_{curr}, where it is proposed to perform the exchange of two entries of opposite sign.

   9b*ii*. Two entries are randomly selected from the i_0-th row of M_{curr}: m_{i_0a}, with value 1, and m_{i_0c} with value 0. The suggestion is to perform a movement towards the matrix M_{prop}, resulting from the exchange of m_{i_0a} with m_{i_0c}.

   9b*iii*. We calculate $\Delta f \leftarrow f(M_{\mathrm{prop}}) - f(M_{\mathrm{curr}})$. This calculation is done recursively as follows:

$$\begin{aligned}\Delta f \;\leftarrow\; & |\phi_a - k - 1| - |\phi_a - k| + |\phi_c - k + 1| - |\phi_c - k| + \\ & + \sum_{\substack{j=1 \\ j \neq i_0}}^{v} \big(|\varphi_{i_0 j} + m_{jc} - m_{ja} - \lambda| - |\varphi_{i_0 j} - \lambda| \big),\end{aligned}$$

   where all evaluations of the functions ϕ_a, ϕ_c and $\varphi_{i_o j}$ are done in M_{curr}.

   9b*iv*. If ($\Delta f \leq 0$) then

   - The motion to the proposed neighbor is accepted: $M_{\mathrm{curr}} \leftarrow M_{\mathrm{prop}}$, $f(M_{\mathrm{curr}}) \leftarrow f(M_{\mathrm{prop}})$. The quantities ϕ_a, ϕ_c and $\varphi_{i_0 j}$ are updated: for $j \neq i_0$ do: $\phi_a \leftarrow \phi_a - 1$, $\phi_c \leftarrow \phi_c + 1$, $\varphi_{i_0 j} \leftarrow \varphi_{i_0 j} + m_{jc} - m_{ja}$.
   - $\mathrm{WereMov} \leftarrow \mathrm{True}$.
   - If ($f(M_{\mathrm{curr}}) < f(M_{\mathrm{best}})$) then
     - $M_{\mathrm{best}} \leftarrow M_{\mathrm{curr}}$, $f(M_{\mathrm{best}}) \leftarrow f(M_{\mathrm{curr}})$.
     - If ($f(M_{\mathrm{best}}) = 0$) then

       Successful completion of the algorithm!

       Endif

     Endif

else [*Metropolis Rule* applies]

- If ($e^{-\Delta f/t_n} > \text{Random}[0,1]$) then
  - ◇ The motion to the proposed neighbor is accepted: $M_{\text{curr}} \leftarrow M_{\text{prop}}$, $f(M_{\text{curr}}) \leftarrow f(M_{\text{prop}})$. The quantities ϕ_a, ϕ_c and $\varphi_{i_0 j}$ are updated, for $j \neq i_0$ do: $\phi_a \leftarrow \phi_a - 1$, $\phi_c \leftarrow \phi_c + 1$, $\varphi_{i_0 j} \leftarrow \varphi_{i_0 j} + m_{jc} - m_{ja}$.
  - ◇ nBad $\leftarrow$ nBad $+ 1$, WhereMov $\leftarrow$ True.

  Endif

Endif

9b*v*. nStep $\leftarrow$ nStep $+ 1$.

End While

9c. nTe $\leftarrow$ nTe $+ 1$, $t \leftarrow t *$ tFactor.

9e. If (WhereMov) then

nChain $\leftarrow 0$, HuboMov $\leftarrow$ False.

else nChain $\leftarrow$ nChain $+ 1$.

Endif.

End While

The algorithm concludes its execution with M_{best} as the solution, which may not achieve the global minimum. In practice, it is necessary to repeat the execution of the algorithm a certain number of times, nRepeat, in order to find the global minimum, perhaps experimenting with some changes in the parameters.

Simulated annealing methods converge to the global minimum under certain conditions. These conditions are met here. However, the discrete simulation of the continuous processes requires a lot of CPU time and some experimentation with the parameters to obtain satisfactory results.

10. Experimental results

Systematic experiments have been performed with different instances of 86 parameters of BIBDs(v, b, r, k, λ), taken from Bofill et al. (2003), Prestwich (2003) y Rodríguez et al. (2009), where $vb \leq 1000$ and $k \neq 3$. This corresponds to the most difficult examples of BIBDs, since the cases where $k = 3$ are easily solved. Table 2 summarizes the results obtained with the different algorithms compared to our algorithm. The first two columns, labeled NN and CLS, respectively, are the results obtained by the neural network algorithm NN (see Bofill et al. (2003)) and the local search with constraints CLS (see Prestwich (2003)). In both cases, NN and CLS only report whether they found the solution or not. NN was run 3 times in each case, while CLS was run 10 times.

Tables 2 and 3 show the results of three other algorithms used in Rodríguez et al. (2009): HC from *hill climbing*, TS from *tabu searc*h and GA from *genetic algorithms*. The researchers report the results of applying each algorithm 30 times to each instance of BIBDs. They report the lowest objective function value, the average and standard deviation, and the percent of exact solutions found.

Finally, we report the results of our *simulated annealing* SA algorithm. The format is similar to that used to report the results of HC, TS, and GA in Rodríguez et al. (2009), but also indicates the minimal time required by the SA algorithm to find the best solution. Five repetitions were run in each instance of the problem. A desktop computer with an Intel (R) Core (TM) i7-3770 CPU @ 3.40GHz was used.

Generation of Incomplete Balanced Incomplete Block Designs (BIBD) with metaheuristics

		BIBD					NN	CLS	HC				TS_{sw}				GA_{sw}				SA_{sw}				
Id	*vb*	*v*	*b*	*r*	*k*	λ			μ	σ	m	%	μ	σ	m	%	μ	σ	m	%	μ	σ	m	%	sec
1	112	8	14	7	4	3	yes	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
2	121	11	11	5	5	2	yes	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
3	150	10	15	6	4	2	yes	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
4	162	9	18	8	4	3	yes	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
5	169	13	13	4	4	1	yes	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
6	180	10	18	9	5	4	yes	yes	1	1.9	0	63	0	0	0	100	0	0	0	100	0	0	0	100	0.1
7	224	8	28	14	4	6	yes	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.0
8	225	15	15	7	7	3	yes	yes	4	5.2	0	57	0	0	0	100	0	0	0	100	0	0	0	100	0.1
9	242	11	22	10	5	4	yes	yes	4	0.8	0	3	0	0	0	100	0	0	0	100	0	0	0	100	0.2
10	256	16	16	6	6	2	yes	yes	4	5.7	0	60	0	0	0	100	0	0	0	100	0	0	0	100	0.1
11	264	12	22	11	6	5	no	yes	6	1.3	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0.1
12	300	10	30	12	4	4	yes	yes	3	1.9	0	37	0	0	0	100	0	0	0	100	0	0	0	100	0.0
13	320	16	20	5	4	1	yes	yes	8	4.1	0	17	0	0	0	100	0	0	0	100	0	0	0	100	0.0
14	324	9	36	16	4	6	yes	yes	0	1.0	0	93	0	0	0	100	0	0	0	100	0	0	0	100	0.1
15	336	8	42	21	4	9	no	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
16	338	13	26	8	4	2	yes	yes	6	1.1	4	0	0	0	0	100	0	0	0	100	0	0	0	100	1.1
17	338	13	26	12	6	5	no	yes	10	1.6	6	0	2.9	1.8	0	27	2.9	1.8	0	27	0	0	0	100	4.5
18	360	10	36	18	5	8	no	yes	3	1.9	0	33	0	0	0	100	0	0	0	100	0	0	0	100	0.0
19	361	19	19	9	9	4	no	yes	46	3.4	35	0	0.4	2.0	0	97	0.4	2.0	0	97	0	0	0	100	0.3
20	363	11	33	15	5	6	no	yes	4.1	1.4	0	7	0	0	0	100	0	0	0	100	0	0	0	100	0.1
21	364	14	26	13	7	6	no	no	13	2.3	6	0	5.8	0.9	4	0	5.8	0.9	4	0	0	0	0	100	34.5
22	384	16	24	9	6	3	no	yes	20	2.2	15	0	4.7	4.4	0	40	4.7	4.4	0	40	0	0	0	100	2.2
23	396	12	33	11	4	3	yes	yes	5	1.2	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0.1
24	441	21	21	5	5	1	yes	yes	23	7.2	0	3	0	0	0	100	0	0	0	100	0	0	0	100	0.2
25	448	8	56	28	4	12	no	yes	0	0	0	100	0	0	0	100	0	0	0	100	0	0	0	100	0.1
26	450	10	45	18	4	6	no	yes	2	2.0	0	50	0	0	0	100	0	0	0	100	0	0	0	100	0.2
27	450	15	30	14	7	6	no	no	17	2.4	11	0	8.1	1.8	4	0	8.1	1.8	4	0	0	0	0	100	103
28	480	16	30	15	8	7	no	no	22.7	2.5	16	0	11.7	2	9	0	11.7	1.51	9	0	3.2	1.6	0	20	623
29	484	11	44	20	5	8	no	yes	3.9	1.7	0	13	0	0	0	100	0	0	0	100	0	0	0	100	0.2
30	486	9	54	24	4	9	no	yes	0.7	1.5	0	83	0	0	0	100	0	0	0	100	0	0	0	100	0.1
31	507	13	39	12	4	3	no	yes	6.4	1.9	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0.2
32	507	13	39	15	5	5	no	yes	8.6	1.7	5	0	0.3	1.0	0	93	0.3	1	0	93	0	0	0	100	4.6
33	512	16	32	12	6	4	no	no	21	2.6	15	0	9.4	1.9	4	0	9.4	1.9	4	0	0	0	0	100	155
34	525	15	35	14	6	5	no	no	16	2.7	10	0	6.7	1.4	4	0	6.7	1.4	4	0	0	0	0	100	858
35	528	12	44	22	6	10	no	yes	6.1	1.7	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0.6
36	529	23	23	11	11	5	no	no	84.4	4.1	72	0	49.5	18.6	0	10	49.5	18.6	0	10	0	0	0	100	2.7
37	540	10	54	27	5	12	no	yes	3.5	1.4	0	13	0	0	0	100	0	0	0	100	0	0	0	100	0.1
38	560	8	70	35	4	15	no	yes	0.1	0.7	0	97	0	0	0	100	0	0	0	100	0	0	0	100	0.1
39	578	17	34	16	8	7	no	no	28.2	2.0	23	0	15.5	1.8	12	0	15.5	1.8	12	0	8.3	1.2	4	0	1280
40	600	10	60	24	4	8	no	yes	2.7	1.9	0	33	0	0	0	100	0	0	0	100	0	0	0	100	0.1
41	605	11	55	20	4	6	no	yes	4.1	1.6	0	10	0	0	0	100	0	0	0	100	0	0	0	100	0.3
42	605	11	55	25	5	10	no	yes	4.5	1.7	0	7	0	0	0	100	0	0	0	100	0	0	0	100	0.3
43	612	18	34	17	9	8	no	no	34.9	3.2	28	0	20.5	1.9	16	0	20.5	1.9	16	0	12	1	11	0	228

Continued...

Tabla 2: Comparison results of 6 algorithms: NN (*Neural Network*), CLS (*Constrained Local Search*), HC (*Hill Climbing*), TS (*Tabu Search*), GA (*Genetic Algorithms*), SA (*Simulated Annealing*). For the NN and CLS algorithms, the exact solution is recorded after 3 and 10 runs. For the HC, TS, and GA algorithms, 30 runs were performed in each case, and the percentage of times the exact solution was obtained is recorded, as well as the average, standard deviation, and minimum of the objective function. For our algorithm, we performed 5 runs in each instance and recorded the percentage of times the exact solution was obtained, the average, standard deviation, and minimum of the objective function, as well as the average time used by the algorithm. Source: Prepared by the authors.

continuation…

		BIBD					NN	CLS	HC				TS				GA				SA				
Id	*vb*	*v*	*b*	*r*	*k*	*λ*			μ	σ	m	%	μ	σ	m	%	μ	σ	%		μ	σ	m	%	sec
44	625	25	25	9	9	3	no	no	100.4	3.1	95	0	66.7	9.5	22	0	66.7	9.5	22	0	0	0	0	100	106
45	630	15	42	14	5	4	no	yes	15.0	2.1	11	0	4.3	1.4	0	7	4.3	1.4	0	7	0	0	0	100	6.8
46	630	21	30	10	7	3	no	no	50.6	3.2	45	0	32.4	1.7	29	0	32.4	1.7	29	0	22	2	14	0	183
47	640	16	40	10	4	2	no	yes	13.1	2.4	8	0	2.4	2	0	40	2.4	2	0	40	0	0	0	100	3.1
48	640	16	40	15	6	5	no	no	20.5	2.6	11	0	8.7	1.7	4	0	8.7	1.7	4	0	0	0	0	100	725
49	648	9	72	32	4	12	no	yes	2.0	2	0	50	0	0	0	100	0	0	0	100	0	0	0	100	0.1
50	675	15	45	21	7	9	no	no	17.6	2.1	13	0	6.0	1.4	4	0	6.03	1.4	4	0	0	0	0	100	1326
51	676	13	52	16	4	4	no	yes	7.6	1.6	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0.7
52	676	13	52	24	6	10	no	yes	10.0	2.8	4	0	0.4	1.2	0	90	0.4	1.2	0	90	0	0	0	100	0.3
53	720	10	72	36	5	16	no	yes	3.2	2.2	0	30	0	0	0	100	0	0	0	100	0	0	0	100	0.2
54	722	19	38	18	9	8	no	no	42.0	3.8	33	0	24.5	2.1	20	0	24.5	2.1	20	0	13.8	1	12	0	180
55	726	11	66	30	5	12	no	yes	5.9	2.2	0	7	0	0	0	100	0	0	0	100	0.0	0	0	100	0.2
56	726	22	33	12	8	4	no	no	61.4	4.3	50	0	40.5	2.7	35	0	40.5	2.7	35	0	There is no solution				
57	780	15	52	26	7	12	no	no	44.6	1.7	41	0	40.1	0.4	40	0	40.1	0.4	40	0	There is no solution				
58	729	27	27	13	13	6	no	no	137.0	6.5	111	0	100.4	3.9	91	0	100.4	3.9	91	0	77	3.0	72	0	863.9
59	735	21	35	15	9	6	no	no	58.3	3.9	51	0	35.9	2.5	30	0	35.9	2.5	30	0	24	1.4	22	0	227.8
60	750	10	75	30	4	10	no	yes	3.4	1.8	0	20	0	0	0	100	0	0	0	100	0	0	0	100	0.1
61	750	25	30	6	5	1	no	yes	49.5	4.1	41	0	14.4	12	0	33	14.4	12	0	33	0	0	0	100	0.7
62	760	20	38	19	10	9	no	no	48.6	3.4	42	0	29.8	1.8	26	0	29.8	1.8	26	0	20	1	19	0	400
63	768	16	48	15	5	4	no	yes	18.5	2.9	13	0	5.2	1.4	0	3	5.2	1.4	0	3	0	0	0	100	1.4
64	768	16	48	18	6	6	no	no	20.6	3	13	0	8.1	1.7	4	0	8.1	1.7	4	0	0	0	0	100	2784
65	792	12	66	22	4	6	no	yes	7.0	2.5	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0
66	792	12	66	33	6	15	no	yes	7.4	2.6	0	3	0	0	0	100	0	0	0	100	0	0	0	100	1.5
67	810	9	90	40	4	15	no	yes	3.3	2.4	0	30	0	0	0	100	0	0	0	100	0	0	0	100	0.1
68	845	13	65	20	4	5	no	yes	9.7	2.4	4	0	0	0	0	100	0	0	0	100	0	0	0	100	0.4
69	847	11	77	35	5	14	no	yes	5.4	2.1	0	3	0.1	0.7	0	97	0.1	0.7	0	97	0	0	0	100	0.4
70	882	21	42	10	5	2	no	no	36.9	3.2	29	0	18.2	1.8	14	0	18.2	1.8	14	0	8.3	1.0	6	0	509
71	882	21	42	12	6	3	no	no	44.7	3.3	37	0	24.2	2.6	17	0	24.2	2.6	17	0	14	2	10	0	100
72	882	21	42	20	10	9	no	no	59.2	5.9	46	0	36.8	2.6	32	0	36.8	2.6	32	0	23	2	21	0	290
73	896	16	56	21	6	7	no	no	22.5	3.3	17	0	8.3	1.7	4	0	8.3	1.7	4	0	0	0	0	100	378
74	900	10	90	36	4	12	no	yes	5.7	2.6	0	10	0	0	0	100	0	0	0	100	0	0	0	100	0.2
75	900	15	60	28	7	12	no	no	19.1	2.8	13	0	5.6	2.3	0	7	5.6	2.3	0	7	0	0	0	100	128
76	918	18	51	17	6	5	no	no	30.6	4.6	21	0	14.1	2.5	9	0	14.1	2.5	9	0	4.6	1.4	0	20	5611
77	924	22	42	21	11	10	no	no	67.3	5.5	54	0	43.9	2.2	41	0	43.9	2.2	41	0	27	1	26	0	420
78	945	15	63	21	5	6	no	yes	16.4	3	11	0	3.2	2.1	0	27	3.2	2.1	0	27	0	0	0	100	22.4
79	960	16	60	15	4	3	no	yes	14.6	3.5	9	0	1.4	2.1	0	67	1.4	2.1	0	67	0	0	0	100	11.2
80	960	16	60	30	8	14	no	no	24.8	3.2	17	0	10.2	2	6	0	10.2	2	6	0	0	0	0	100	1427
81	961	31	31	6	6	1	no	yes	100.8	5	87	0	22.9	19	0	40	22.9	19	0	40	0	0	0	100	0.4
82	961	31	31	10	10	3	no	no	175.6	6.4	160	0	134.1	5.1	124	0	134.1	5.1	124	0	100	1	100	0	596
83	961	31	31	15	15	7	no	no	206.1	6.1	194	0	159.6	5.9	148	0	159.6	5.9	148	0	122	3.0	116	0	633
84	968	11	88	40	5	16	no	yes	7.7	2.6	0	3	0.5	1.3	0	87	0.5	1.3	0	87	0	0	0	100	0.4
85	968	22	44	14	7	4	no	no	57.7	4.6	44	0	34.8	2.6	31	0	34.8	2.6	31	0	21	0	20	0	874
86	1000	25	40	16	10	6	no	no	98.6	5.8	88	0	65.7	3.6	56	0	65.7	3.6	56	0	45	2	42	0	781

Tabla 3: Continuation of the list of examined BIBD instances. Source: Prepared by the authors.

Our algorithm outperformed the others in finding exact solutions in all cases examined. Our simulated annealing algorithm also identified that BIBD(15, 52, 26, 7, 12) does not exist because it does not satisfy the constraint (1). This was not detected by the other algorithms. Regarding the non-existence of BIBD(22, 33, 12, 8, 4), it was already stated in section 3 that this is the smallest BIBD that satisfies all constraints but has no solution.

Our algorithm further obtained an exact solution in the following 12 instances of BIBDs(v, b, r, k, λ), with parameters equal to (14, 26, 13, 7, 6), (15, 35, 14, 6, 5), (16, 30,15, 8, 7), (16, 32, 12, 6, 4), (15, 35, 14, 6, 5), (16, 48, 18, 6, 6), (23, 23, 11, 11, 5), (25, 25, 9, 9, 3), (15, 42, 14, 5, 4), (16, 52, 21, 6, 7), (16, 60, 30, 8, 14), and (18, 51, 17, 6, 5). The previous algorithms had not found exact solutions in these instances of BIBDs. Table 4 compares the different algorithms in terms of the number of instances they solve correctly.

NN	**CLS**	**HC**	**TS**	**GA**	**SA**
16	55	35	57	37	70
18.6 %	63.9 %	40.7 %	62.3 %	40.0 %	81.4 %

Tabla 4: Number and percentage of instances solved by each algorithm out of the 86 instances considered. Source: Prepared by the authors.

However, perhaps all but one of the 12 BIBDs found in this comparison table using our algorithm are likely to be isomorphic solutions to other already known designs for these parameter sets. We do not know at this time. In fact, if we denote by N_d the number of known pairwise isomorphic designs for a given BIBD(v, b, r, k, λ), then the number of known solutions is shown in Table 5 (see Colbourn y Dinitz (2006)). However, the particular solution found for BIBD(16,56,21,6,7) is really new, as we will see in the next section.

Id	BIBD	N_d
21	BIBD$(14, 26, 13, 7, 6)$	$N_d = 15111019$
27	BIBD$(15, 30, 14, 7, 6)$	$N_d = 109$
28	BIBD$(16, 30, 15, 8, 7)$	$N_d \geq 9 \times 10^7$
33	BIBD$(16, 32, 12, 6, 4)$	$N_d \geq 111$
34	BIBD$(15, 35, 14, 6, 5)$	$N_d \geq 117$
36	BIBD$(23, 23, 11, 11, 5)$	$N_d = 1106$
44	BIBD$(25, 25, 9, 9, 3)$	$N_d = 78$
50	BIBD$(15, 45, 21, 7, 9)$	$N_d \geq 10^8$
64	BIBD$(16, 48, 18, 6, 6)$	$N_d > 10^8$
73	BIBD$(16, 56, 21, 6, 7)$	$N_d > 1$
76	BIBD$(18, 51, 17, 6, 5)$	$N_d > 582$
80	BIBD$(16, 60, 30, 8, 14)$	$N_d > 9 \times 10^7$

Tabla 5: Number of designs N_d for the 12 BIBD(v, b, r, k, λ) for which we found a solution with our algorithm, according to the *Handbook of Combinatorial Designs*, by Colbourn y Dinitz (2006). Source: Prepared by the authors.

11. Conclusions and future work

Let us turn our attention to the only interesting BIBD we found, BIBD(16,56, 21,6,7), for which we find the following solution:

$$
\begin{array}{llll}
B_1 = \{2,3,7,10,14,15\} & B_2 = \{1,4,5,11,15,16\} & B_3 = \{1,2,3,5,6,8\} & B_4 = \{5,8,10,11,15,16\} \\
B_5 = \{4,8,11,12,14,16\} & B_6 = \{3,6,8,10,14,16\} & B_7 = \{2,4,10,11,12,14\} & B_8 = \{1,3,4,8,13,14\} \\
B_9 = \{1,9,12,13,14,15\} & B_{10} = \{1,6,7,10,11,13\} & B_{11} = \{3,4,10,11,13,16\} & B_{12} = \{2,7,8,10,12,13\} \\
B_{13} = \{2,8,9,14,15,16\} & B_{14} = \{2,7,11,12,14,16\} & B_{15} = \{5,7,8,9,11,15\} & B_{16} = \{2,3,4,12,15,16\} \\
B_{17} = \{3,5,7,10,11,12\} & B_{18} = \{2,5,6,8,11,12\} & B_{19} = \{2,3,5,10,12,15\} & B_{20} = \{1,3,7,10,12,14\} \\
B_{21} = \{1,2,6,9,10,15\} & B_{22} = \{3,4,8,9,12,13\} & B_{23} = \{1,4,5,6,12,14\} & B_{24} = \{6,7,8,9,12,15\} \\
B_{25} = \{4,5,8,9,10,14\} & B_{26} = \{1,5,9,12,13,16\} & B_{27} = \{1,3,7,9,11,16\} & B_{28} = \{1,2,7,9,11,16\} \\
B_{29} = \{3,6,9,13,14,16\} & B_{30} = \{1,3,8,11,13,15\} & B_{31} = \{2,4,7,8,13,16\} & B_{32} = \{9,10,11,13,14,15\} \\
B_{33} = \{1,3,5,11,14,15\} & B_{34} = \{1,2,4,5,7,8\} & B_{35} = \{1,6,8,12,15,16\} & B_{36} = \{3,5,8,9,10,16\} \\
B_{37} = \{1,2,8,11,13,14\} & B_{38} = \{4,6,8,9,10,11\} & B_{39} = \{2,3,4,6,9,11\} & B_{40} = \{2,6,10,13,15,16\} \\
B_{41} = \{3,7,8,12,13,15\} & B_{42} = \{2,4,5,13,14,15\} & B_{43} = \{1,4,9,10,12,15\} & B_{44} = \{4,5,6,7,13,15\} \\
B_{45} = \{5,7,9,12,14,16\} & B_{46} = \{2,3,5,6,9,14\} & B_{47} = \{4,6,7,11,14,15\} & B_{48} = \{2,6,9,11,12,13\} \\
B_{49} = \{4,5,7,9,10,13\} & B_{50} = \{1,6,7,8,10,14\} & B_{51} = \{3,4,6,7,15,16\} & B_{52} = \{1,4,6,10,12,16\} \\
B_{53} = \{5,6,7,13,14,16\} & B_{54} = \{3,5,6,11,12,13\} & B_{55} = \{1,2,5,10,13,16\} & B_{56} = \{1,2,3,4,7,9\}.
\end{array}
$$

It is known that the number of N_d Non-pairwise-isomorphic solutions for this BIBD(16,56,21,6,7) is greater than 1 (see Colbourn y Dinitz (2006)). However, only one solution is actually known. The only known solution of this BIBD(16,56,21,6,7) is obtained by juxtaposing one solution of BIBD(16,16,6,6,2) with another solution of BIBD(16,40,15,6,5) using Fischer's juxtaposition rule explained in Section 3 of this article. While it is known that there are only 3 isomorphic distinct solutions for the first BIBD(16,16,6,6,2) and at least 25 isomorphic distinct solutions for the second BIBD(16,40,15,6,5), all juxtapositions of these BIBDs produce solutions for BIBD(16,56,21,6,7) that are pairwise isomorphic. That is, so far only one solution is known for BIBD(16,56,21,6,7), although by other theoretical means it is known that this BIBD has other solutions that are not isomorphic to the only one found so far (see Colbourn y Dinitz (2006)).

We use the The GAP Group (2025) software to compare the solution of BIBD(16,56,21, 6,7) found by our algorithm with the only known solution of BIBD(16,56,21,6,7). We find that they are non-isomorphic solutions, so it is the second solution of BIBD(16,56,21,6,7). This is our only new result.

Although our algorithm produced good results on the 86 instances of BIBDs analyzed, we believe it can be improved by considering an objective function to be minimized of the type

$$f(M) = \sum_{i=1}^{v} |\alpha_i(M) - r| + \sum_{j=1}^{b} |\beta_j(M) - k| + \sum_{1 \le i < i' \le v} |\varphi_{ii'}(M) - \lambda|,$$

where $\alpha_i(M)$ is the number of 1's in the i-th row of the binary matrix M, $\beta_j(M)$ is the number of 1's in the j-th column of M, and $\varphi_{ii'}(M)$ is the inner product of the i-th and i'-th rows. The type of moves to be made will be to randomly choose an entry m_{ij} of M and change its value from 0 to 1 or from 1 to 0, following the *Metropolis rule*. We hope that this algorithm will be a bit more flexible than the previous one, and will be able to solve those instances of the 86 BIBDs list where the current algorithm failed.

We will also conduct research on *Supplementary Difference Sets* (SDS), devices that allow us to build some instances of BIBDs of larger sizes (see Morales (2000)).

Author Contributions: Conceptualization: E.P.; E.S.; Formal analysis: E.P.; E.S.; Investigation: E.P.; E.S.; Methodology: E.P.; E.S.; Project administration: E.P.; E.S.; Resources: E.P.; E.S.; Software: E.P.; E.S.; Validation: E.P.; E.S.; Visualization: E.P.; E.S.; Writing – original draft: E.P.; E.S.; Writing – review and editing: E.P.; E.S.

Data availability: The data used in this article are available from the authors upon request by contacting eduardo.piza@ucr.ac.cr or esteban.seguraugalde@ucr.ac.cr.

Declaration of Non-Use of Artificial Intelligence The authors declare that they did not use artificial intelligence tools in the preparation of this article.

Referencias

Aarts, E. H. L., & Korst, J. H. M. (1990). *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing*. John Wiley & Sons.

Anderson, I., & Honkala, I. (2012). A short course in combinatorial designs [Spring 1997, revised 2012]. https://apachepersonal.miun.se/~tomnil/designs/designsShortC.pdf

Bofill, P., Guimerà, R., & Torras, C. (2003). Comparison of simulated annealing and mean field annealing as applied to the generation of block designs. *Neural Networks*, *16*(10), 1421-1428. https://doi.org/10.1016/j.neunet.2003.07.003

Bruck, R. H., & Ryser, H. J. (1949). The nonexistence of certain finite projective planes. *Canadian Journal of Mathematics*, *1*(1), 88-93. https://doi.org/10.4153/CJM-1949-009-2

Chowla, S., & Ryser, H. J. (1950). Combinatorial problems. *Canadian Journal of Mathematics*, *2*, 93-99. https://doi.org/10.4153/CJM-1950-009-8

Colbourn, C. J., & Dinitz, J. H. (2006). *Handbook of combinatorial designs* (2nd ed.). Chapman & Hall/CRC.

Corneil, D. G., & Mathon, R. A. (1978). Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations. *Annals of Discrete Mathematics*, *2*, 1-32. https://doi.org/10.1016/S0167-5060(08)70319-4

Fisher, R. A. (1940). An examination of the different possible solutions of a problem in incomplete blocks. *Annals of Eugenics*, *10*, 52-75. https://doi.org/10.1111/j.1469-1809.1940.tb02237.x

Gibbons, P. B., & Ostergård, P. R. J. (1996). Computational methods in design theory. En C. J. Colbourn & J. H. Dinitz (Eds.), *The CRC handbook of combinatorial designs* (pp. 730-740). CRC Press.

Godsil, C. D. (2019). Combinatorial design theory.

Gross, J. L. (2007). *Combinatorial methods with computer applications*. Chapman & Hall/CRC.

Hirschfeld, J. W. P. (1998). *Projective geometries over finite fields* (2nd ed.). Oxford University Press.

Kalmanovich, D. (2005). *Finite projective planes* [Mathematics thesis]. Ben-Gurion University of the Negev.

Lam, C. W. H. (1991). The search for a finite projective plane of order 10. *The American Mathematical Monthly*, *98*(4), 305-318. https://doi.org/10.1080/00029890.1991.12000759

Morales, L. B. (2000). Constructing difference families through an optimization approach: Six new BIBDs. *Journal of Combinatorial Designs*, *8*(4), 261-273. https://doi.org/10.1002/1520-6610(2000)8:4⟨261::AID-JCD4⟩3.0.CO;2-U

Piza, E. (2011). Búsqueda de matrices de Hadamard a través de secuencias de Turyn. *Revista de Matemática: Teoría y Aplicaciones*, *18*(2), 193-214. https://doi.org/10.15517/rmta.v18i2.2094

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge University Press.

Prestwich, S. D. (2003). A local search algorithm for balanced incomplete block designs. En F. Rossi (Ed.), *Principles and practice of constraint programming–CP 2003* (pp. 53-64, Vol. 2833). Springer. https://doi.org/10.1007/3-540-36607-5_10

Rodríguez, D., Cotta, C., & Fernández, A. J. (2009). Finding balanced incomplete block designs with metaheuristics. En C. Cotta & P. Cowling (Eds.), *Evolutionary computation in combinatorial optimization* (pp. 156-167, Vol. 5482). Springer. https://doi.org/10.1007/978-3-642-01009-5_14

Rosa, A. (1987). Combinatorial design theory. *Annals of Discrete Mathematics*, *41*, 1-469.

Stinson, D. R. (2003). *Combinatorial designs: Constructions and analysis*. Springer.

The GAP Group. (2025). GAP: Groups, algorithms, and programming, version 4.15.1 [Computer software]. https://www.gap-system.org/