Finding Balanced Incomplete Block Designs with Metaheuristics

David Rodríguez Rueda[1], Carlos Cotta[2], and Antonio J. Fernández[2]

[1] Universidad Nacional Experimental del Táchira (UNET), Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela, drodri@unet.edu.ve

[2] Universidad de Málaga, ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain, {ccottap,afdez}@lcc.uma.es

Abstract. This paper deals with the generation of balanced incomplete block designs (BIBD), a hard constrained combinatorial problem with multiple applications. This problem is here formulated as a combinatorial optimization problem (COP) whose solutions are binary matrices. Two different neighborhood structures are defined, based on bit-flipping and position-swapping. These are used within three metaheuristics, i.e., hill climbing, tabu search, and genetic algorithms. An extensive empirical evaluation is done using 86 different instances of the problem. The results indicate the superiority of the swap-based neighborhood, and the impressive performance of tabu search. This latter approach is capable of outperforming two techniques that had reported the best results in the literature (namely, a neural network with simulated annealing and a constraint local search algorithm).

1 Introduction

The generation of block designs is a well-known combinatorial problem, which is very hard to solve [1]. The problem has a number of variants, among which a popular one is the so-called Balanced Incomplete Block Designs (BIBDs). Basically, a BIBD is defined as an arrangement of v distinct objects into b blocks such that each block contains exactly k distinct objects, each object occurs in exactly r different blocks, and every two distinct objects occur together in exactly λ blocks (for $k, r, \lambda > 0$). The construction of BIBDs was initially attacked in the area of experiment design [2, 3]; however, nowadays BIBD can be applied to a variety of fields such as cryptography [4] and coding theory [5], among others.

BIBD generation is a NP-hard problem [6] that provides an excellent benchmark since it is scalable and has a wide variety of problem instances, ranging from easy instances to very difficult ones. The scalability of the problem as well as its difficulty make it an adequate setting to test the behavior of different techniques/algorithms. As it will be discussed in Sect. 2.2, complete methods (including exhaustive search) have been applied to the problem although this remains intractable even for designs of relatively small size [7]. As a proof of the difficulty of the problem, there currently exist a number of open instances that have not been solved yet (of course, it might be the case that there is no solution for them; then again, insolvability could not be established by complete methods). The fact that, in the general case, the algorithmic generation of block designs is an NP-hard problem [6] makes complete methods be inherently limited by the size of the problem instances. The application of metaheuristics thus seems to be more appropriate to attack larger problem instances. This paper provides some steps in this direction and demonstrates empirically that these approaches (particularly, local search techniques) are effective methods in the design of balanced incomplete blocks. More specifically, the paper describes two local searchers –i.e., a steepest descent hill climbing (HC) algorithm and a tabu search (TS)– and a genetic algorithm (GA), each of them with two variants. A wide range of problem instances have been tackled by these metaheuristics, and the results have been compared with two techniques found in the literature that reported the best results [8, 9]. In the following we will show that a particular TS algorithm outperforms the remaining approaches and even can find solutions to instances that the other methods could not solve.

2 Background

This section provides a brief overview of the problem, presents its classical formulation, and discusses how it has been tackled in the literature.

2.1 Formulation

A standard way of representing a BIBD is in terms of its incidence matrix $M \equiv \{m_{ij}\}_{v\times b}$, which is a $v \times b$ binary matrix with exactly r ones per row, k ones per column, a scalar product of λ between any pair of distinct rows, and where $m_{ij} \in \{0,1\}$ is equal to 1 if the ith object is contained in the jth block, and 0 otherwise; in this context, m_{ij} represents the incidence of object i in block j of M. A BIBD is then specified by five parameters $\langle v, b, r, k, \lambda\rangle$, i.e., a $\langle v, b, r, k, \lambda\rangle$-BIBD consists of a set of v points that is divided into b subsets in such a way that each point in v is contained in r different subsets and any couple of points in v is contained in $\lambda < b$ subsets with $k < v$ points in each subset.

The five parameters defining a $\langle v, b, r, k, \lambda\rangle$−BIBD are related and satisfy the following two relations: $bk = vr$ and $\lambda(v-1) = r(k-1)$. In fact, the corresponding instance can be defined by just three parameters $\langle v, k, \lambda\rangle$ since b and r are given in terms of the other parameters:

$$b = \frac{v(v-1)\lambda}{k(k-1)} \qquad\qquad r = \frac{(v-1)\lambda}{k-1} \tag{1}$$

Clearly, these relations restrict the set of admissible parameters for a BIBD; however, the parameter admissibility is a necessary condition but it is not sufficient to guarantee the existence of a BIBD [10, 11]. According to the Fisher's inequality theorem [12], $b > v$ in any block design; the case $b = v$ represents an special design called *symmetric design*. A direct consequence of a symmetric

0	0	0	1	0	1	1	1	0	0	0	1	1	1
1	1	0	1	1	0	1	0	0	1	0	0	0	1
0	1	1	1	1	1	0	0	1	0	0	0	1	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0
1	0	1	0	1	1	0	0	0	0	1	1	0	1
0	1	0	0	0	0	0	0	1	1	1	1	1	1
1	1	1	0	0	0	1	1	1	0	0	1	0	0
1	0	1	1	0	0	0	1	0	1	1	0	1	0

0	1	0	1	0	1	0
1	0	0	1	0	0	1
1	1	1	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	1
1	0	0	0	1	1	0
0	0	1	1	1	0	0

Fig. 1. (Left) a $\langle 8, 14, 7, 4, 3\rangle$−BIBD; (Right) a $\langle 7, 7, 3, 3, 1\rangle$−symmetric BIBD.

design is that $r = k$. This kind of blocks are usually used with a maximum order of $v = b = 7$, although this is not a strict requirement. Figure 1 shows configurations of the incidence matrix M representing possible solutions to a $\langle 8, 14, 7, 4, 3\rangle$−BIBD and a symmetric $\langle 7, 7, 3, 3, 1\rangle$−BIBD, respectively.

2.2 Related work

The BIBD problem has been tackled by a number of different techniques in the literature, with different success. Traditionally, the problem was dealt via deterministic, constructive and/or complete methods. For instance John *et al.* [13, 14] used mathematical programming methods to look for an optimal incomplete block design. Also, Zergaw [15] considered the error correlation, and presented a sequential algorithm for constructing optimal block designs. Following this line of work, Tjur [16] incorporated interchange mechanisms via the addition of experimental units (blocks) one by one. Flener *et al.* [17] proposed a matricial model based on ECLIPSE to solve the problem of block generation. Also, constraint programming techniques have been used; this way, Puget [18] formulated the problem as a constraint satisfaction problem (CSP) where each instance was represented by a classical binary matrix of size $v \times b$. Puget proposed to combine methods for *symmetry breaking via dominance detection* and *symmetry breaking using stabilizers* in order to solve the problem. Also, [19] explored two strategies (namely, a heuristic for variable selection and a domain pruning procedure) for exploiting the symmetry of the problem. The underlying idea in this work was to use symmetries to guide the search for a solution. The objective of this work was not solving specific instances but being effective in reducing search effort. Be as it may, although all these methods can be used to design BIBDs, their applicability is limited by the size of the problem instances. A survey of known results can be found in [1].

Stochastic methods were also applied to the problem. For example, the generation of BIBDs is formulated in [8] as a COP tackled with a neural network. Several optimization strategies were considered as relaxation strategies for comparative purposes. A simulated annealing algorithm endowed with this neural network (NN-SA) was shown to offer better performance than an analogous hybridization with mean field annealing. These results were further improved by Prestwich [20, 9], that considered different schemes for adding symmetry breaking constraints inside a constrained local search (CLS).

In general, most of the proposals to generate BIBDs are focused in unsolved problems and consider a small number of instances, and only a small number of papers provide an extensive experimentation on a large set of instances. Among these papers, the most interesting ones are [8, 9, 19]. To the best of our knowledge, [9] provides the best results published in many instances of the problem and therefore, represents the state-of-the-art in the generation of BIBDs. For these reasons the NN-SA and CLS proposals will be later considered in the experimental section of this paper for comparative purposes with the methods described in this paper.

Let us finally mention that there exist other variants of the BIBD problem, e.g., partially BIBDs, randomized block designs, pairwise balanced designs, regular graph designs, and maximally balanced maximally uniform designs, among others [21–24]. Although in some cases metaheuristic approaches have been used on some of them [25–27], to the best of our knowledge there exists no previous literature on this line of attack for the BIBD problem we consider in this work (save the SA approach mentioned before).

3 Solving the $\langle v, b, r, k, \lambda \rangle$-BIBD problem

The BIBD problem exhibits a clear combinatorial structure, and can be readily transformed in an optimization task. We have approached this challenging resolution via two local search techniques (HC and TS) and a population-based technique (GA), which will be described below. To this end, let us firstly define the objective function, and possible neighborhood structures.

3.1 Objective function

The generation of BIBDs is a CSP posed here as a COP. This is done by relaxing the problem (allowing the violation of constraints) and defining an objective function that accounts for the number and degree of violation of them. More precisely, for the general case of the instance $\langle v, b, r, k, \lambda \rangle$, the following objective function is defined:

$$f^{\langle v,b,r,k,\lambda \rangle}(M) = \sum_{i=1}^{v} \phi_{ir}(M) + \sum_{j=1}^{b} \phi'_{jk}(M) + \sum_{i=1}^{v-1} \sum_{j=i+1}^{v} \phi''_{ij\lambda}(M) \tag{2}$$

where

$$\phi_{ir}(M) = \left| r - \sum_{j=1}^{b} m_{ij} \right| ; \ \phi'_{jk}(M) = \left| k - \sum_{i=1}^{v} m_{ij} \right| ; \ \phi''_{ij\lambda}(M) = \left| \lambda - \sum_{k=1}^{b} m_{ik} m_{jk} \right| \tag{3}$$

Observe that, for a given incidence matrix M, the value returned by the objective function sums up all discrepancies with respect to the expected values of the row constraints, column constraints and scalar product constraints. Obviously, the aim is to minimize the value of the objective function. If the instance is satisfiable, a global optimum is a configuration M^* such that $f^{\langle v,b,r,k,\lambda \rangle}(M^*) = 0$.

3.2 Neighborhood Structures

Two neighborhood structures are considered. The first one arises naturally from the binary representation of solutions as the incidence matrix M. This neighborhood is based on the Hamming distance, and will be denoted as bit-flip. Let $H(M_1, M_2)$ be the Hamming distance between two incidence matrices M_1 and M_2; the bit-flip neighborhood is defined as $\mathcal{N}_{bit-flip}(M) = \{M' \mid H(M, M') = 1\}$. Clearly the size of this neighborhood is $|\mathcal{N}_{bit-flip}(M)| = vb$, and the evaluation of any $M' \in \mathcal{N}_{bit-flip}(M)$ requires the incremental re-computation (with respect to the evaluation of M) of $v + 1$ constraints (i.e., 1 row constraint + 1 column constraint + $v - 1$ scalar products). Observe that evaluating a solution from scratch requires to compute exactly $v + b + v(v - 1)/2$ constraints, and thus the complete exploration of the neighborhood can be assimilated to $n_{eq} = \frac{vb(v+1)}{v+b+v(v-1)/2}$ full evaluations. This consideration will be useful in order to provide a fair basis for comparing local search and population-based techniques later on, i.e., by taking constraint checks as a measure of computational effort. While this measure can admit several nuances, it is more informative than the number of solutions generated, and more hardware-independent than, e.g., running time.

A second neighborhood structure –which we denote as swap– can be considered as well. The underlying idea here is to take an object from one block, and move it to a different one. This can be formulated in binary terms as permuting a 0 and a 1 within the same row. Notice that by doing so, if a configuration holds the row constraint for a specific row, then all its neighbors will also hold it. The swap neighborhood is defined as $\mathcal{N}_{swap}(M) = \{M' \mid \exists! i, j, k : m_{ij} = m'_{ik} = 0, m'_{ij} = m_{ik} = 1\}$. Clearly, the size of this neighborhood is $|\mathcal{N}_{swap}(M)| = vr(b - r)$, from which the number of evaluations to explore the complete neighborhood can be directly inferred. Note that row constraints do not have to be re-evaluated as the number of 1's per row remains constant. In any case, the impact of this consideration is minimal since the computing effort is dominated by the quadratic term in the denominator. Let us note as a final consideration that a symmetrical version of this latter neighborhood could be defined, substituting an object by another different one within a block. Notice however that the objective function is not symmetrical in this sense, and the cost of exploring this neighborhood is higher (and it exhibits other difficulties when deployed on a GA, as $\mathcal{N}_{swap}$ will be in Sect. 3.4). For this reason, it has not been considered in this work.

3.3 Local search techniques

Two different versions of a hill climbing (HC) approach and a tabu search (TS) algorithm were defined on the basis of the two neighborhood structures. These are denoted as HC$_{bf}$, HC$_{sw}$, TS$_{bf}$ and TS$_{sw}$ respectively. Besides the obvious differences in algorithmic aspects and neighborhood computation, there is an additional consideration regarding the choice of neighborhood: since the swap neighborhood does not alter the number of 1's per row, if the initial solution does not fulfill all row constraints no feasible solution will be ever found. Hence, it is mandatory to enforce these constraints when generating the starting point for a swap-based local search algorithm. Their bit-flip-based counterparts do not require this, and can take a fully random solution as initial point for the search. Nevertheless, the effect that such a guided initialization can have on these latter algorithms has been empirically studied as well in Sect. 4.

The HC algorithms follow a steepest-descent procedure: the neighborhood of the current solution is completely explored, and the best solution is chosen unless this is worse than the current one; if this is the case, the current solution is a local optimum, and the process is re-started from a different point (randomly chosen) until the computational budget allocated is exhausted. Regarding the TS algorithms, they also conduct a full-exploration of the current neighborhood, moving to the best non-tabu neighbor even if it is worse than the current solution. In the case of TS_{bf}, a move is tabu if it modifies a specific bit m_{ij} stored in the tabu list. Similarly, in the case of TS_{sw}, a move is tabu if it attempts to reverse a previous swap $m_{ij} \leftrightarrow m_{ik}$ stored in the tabu list. To prevent cycling, the tabu tenure –i.e., the number of iterations tabu move stays in the list– is chosen randomly in the range $[\beta/2, 3\beta/2]$, where $\beta = vb$ in TS_{bf} and $\beta = vbr$ in TS_{sw}. The tabu status of a move can be overridden if the aspiration criteria is fulfilled, namely, finding a solution better than the current best solution found so far. After a number of n_ι evaluations (a parameter that we will set as a function of the total number of evaluations) with no improvement, the search is intensified, by returning to the best solution found so far.

3.4 Genetic algorithm

Two versions of a steady state GA have been considered. Both of them use binary tournament selection and replacement of the worst individual in the population. They differ in the reproductive stage though. The first one, which we denote as GA_{bf}, is related to the bit-flip neighborhood, since it uses uniform crossover and bit-flip mutation. The second one is denoted as GA_{sw}, and is more related to the swap neighborhood. To be precise, this latter algorithm performs uniform crossover at row level (that is, it randomly selects entire rows from either of the parents), and uses swap mutation. Obviously, this implies that the unitation of each row is never changed, and therefore the initialization of the population has to be done with solutions fulfilling all row constraints, as it was the case with HC_{sw} and TS_{sw}. Again, this guided initialization can be optionally done in GA_{bf}, although it is not mandatory.

To keep diversity in the population, both GA variants ban duplicated solutions, i.e., if an offspring is a copy of an existing solution it is discarded. Furthermore, a re-starting mechanism is introduced to re-activate the search whenever stagnation takes place. This is done by keeping a fraction $f_\%$ of the top individuals in the current population, and refreshing the rest of the population with random individuals. This procedure is triggered after a number of n_ι evaluations with no improvement of the current best solution.

Table 1. BIBD instances considered in this work, and their solvability status with respect to the simulated annealing/neural network hybrid algorithm (NN-SA) in [8], and the constrained local search algorithm (CLS) in [9].

ID	v	b	r	k	λ	vb	NN-SA	CLS	ID	v	b	r	k	λ	vb	NN-SA	CLS
1	8	14	7	4	3	112	yes	yes	44	25	25	9	9	3	625	no	no
2	11	11	5	5	2	121	yes	yes	45	15	42	14	5	4	630	no	yes
3	10	15	6	4	2	150	yes	yes	46	21	30	10	7	3	630	no	no
4	9	18	8	4	3	162	yes	yes	47	16	40	10	4	2	640	no	yes
5	13	13	4	4	1	169	yes	yes	48	16	40	15	6	5	640	no	no
6	10	18	9	5	4	180	yes	yes	49	9	72	32	4	12	648	no	yes
7	8	28	14	4	6	224	yes	yes	50	15	45	21	7	9	675	no	no
8	15	15	7	7	3	225	yes	yes	51	13	52	16	4	4	676	no	yes
9	11	22	10	5	4	242	yes	yes	52	13	52	24	6	10	676	no	yes
10	16	16	6	6	2	256	yes	yes	53	10	72	36	5	16	720	no	yes
11	12	22	11	6	5	264	no	yes	54	19	38	18	9	8	722	no	no
12	10	30	12	4	4	300	yes	yes	55	11	66	30	5	12	726	no	yes
13	16	20	5	4	1	320	yes	yes	56	22	33	12	8	4	726	no	no
14	9	36	16	4	6	324	yes	yes	57	15	52	26	7	12	780	no	no
15	8	42	21	4	9	336	no	yes	58	27	27	13	13	6	729	no	no
16	13	26	8	4	2	338	yes	yes	59	21	35	15	9	6	735	no	no
17	13	26	12	6	5	338	no	yes	60	10	75	30	4	10	750	no	yes
18	10	36	18	5	8	360	no	yes	61	25	30	6	5	1	750	no	yes
19	19	19	9	9	4	361	no	yes	62	20	38	19	10	9	760	no	no
20	11	33	15	5	6	363	no	yes	63	16	48	15	5	4	768	no	yes
21	14	26	13	7	6	364	no	no	64	16	48	18	6	6	768	no	no
22	16	24	9	6	3	384	no	yes	65	12	66	22	4	6	792	no	yes
23	12	33	11	4	3	396	yes	yes	66	12	66	33	6	15	792	no	yes
24	21	21	5	5	1	441	yes	yes	67	9	90	40	4	15	810	no	yes
25	8	56	28	4	12	448	no	yes	68	13	65	20	4	5	845	no	yes
26	10	45	18	4	6	450	no	yes	69	11	77	35	5	14	847	no	yes
27	15	30	14	7	6	450	no	no	70	21	42	10	5	2	882	no	no
28	16	30	15	8	7	480	no	no	71	21	42	12	6	3	882	no	no
29	11	44	20	5	8	484	no	yes	72	21	42	20	10	9	882	no	no
30	9	54	24	4	9	486	no	yes	73	16	56	21	6	7	896	no	no
31	13	39	12	4	3	507	no	yes	74	10	90	36	4	12	900	no	yes
32	13	39	15	5	5	507	no	yes	75	15	60	28	7	12	900	no	no
33	16	32	12	6	4	512	no	no	76	18	51	17	6	5	918	no	no
34	15	35	14	6	5	525	no	no	77	22	42	21	11	10	924	no	no
35	12	44	22	6	10	528	no	yes	78	15	63	21	5	6	945	no	yes
36	23	23	11	11	5	529	no	no	79	16	60	15	4	3	960	no	yes
37	10	54	27	5	12	540	no	yes	80	16	60	30	8	14	960	no	no
38	8	70	35	4	15	560	no	yes	81	31	31	6	6	1	961	no	yes
39	17	34	16	8	7	578	no	no	82	31	31	10	10	3	961	no	no
40	10	60	24	4	8	600	no	yes	83	31	31	15	15	7	961	no	no
41	11	55	20	4	6	605	no	yes	84	11	88	40	5	16	968	no	yes
42	11	55	25	5	10	605	no	yes	85	22	44	14	7	4	968	no	no
43	18	34	17	9	8	612	no	no	86	25	40	16	10	6	1000	no	no

4 Experimental results

The experiments have been done on 86 instances taken from [8, 9] where $vb \leqslant 1000$ and $k \neq 3$. This corresponds to the hardest instances reported therein, since the cases where $k = 3$ were easily solvable. Table 1 shows the particular instances considered, along with an identification label, and their solvability status regarding the NN-SA [8] and CLS [9] algorithm. Although the data reported in [9] is limited to the best result out of 3 runs of CLS per instance, it must be noted that the number of instances solved by the latter algorithm is more than 3 times that of NN-SA.

All algorithms have been run 30 times per problem instance. To make the comparison with CLS as fair as possible, all runs of local search techniques are limited to explore $n_\nu = 2 \cdot 10^6$ neighbors. This number correspond to the maximum number of backtrack steps (fixing one entry of the incidence matrix) performed by CLS in [9]. The GAs consider the equivalent number of full evaluations in each case (see Sect. 3.2). The number of evaluations without improvement to trigger intensification in TS or re-starting in GA is $n_\iota = n_\nu/10$. Other parameters of the GA are *population size*= 100, crossover and mutation probabilities

Table 2. Results of HC algorithms (30 runs per instance). $\overline{x}$, σ, B and S denote, respectively, the fitness average value, the standard deviation, the best obtained result, and the number of times that a problem instance solution is obtained.

	HC_{bf}			HC_{sw}				HC_{bf}			HC_{sw}		
ID	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S	ID	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S
1	3.50 ± 1.12	0	2	0.00 ± 0.00	0	30	44	139.67 ± 3.80	130	0	100.40 ± 3.08	95	0
2	5.13 ± 4.62	0	13	0.00 ± 0.00	0	30	45	35.33 ± 3.17	29	0	15.00 ± 2.14	11	0
3	5.83 ± 1.24	4	0	0.00 ± 0.00	0	30	46	84.60 ± 3.53	78	0	50.63 ± 3.22	45	0
4	5.73 ± 1.44	0	1	0.00 ± 0.00	0	30	47	29.63 ± 2.74	23	0	13.10 ± 2.41	8	0
5	3.13 ± 4.15	0	19	0.00 ± 0.00	0	30	48	47.57 ± 2.63	43	0	20.47 ± 2.60	11	0
6	11.27 ± 1.63	7	0	1.47 ± 1.93	0	19	49	25.93 ± 3.80	13	0	2.00 ± 2.00	0	15
7	7.80 ± 1.19	5	0	0.00 ± 0.00	0	30	50	54.90 ± 4.43	41	0	17.60 ± 2.09	13	0
8	36.60 ± 2.68	30	0	4.33 ± 5.17	0	17	51	24.67 ± 2.44	16	0	7.63 ± 1.62	4	0
9	14.67 ± 1.70	10	0	3.97 ± 0.84	0	1	52	42.20 ± 3.00	36	0	10.03 ± 2.76	4	0
10	37.23 ± 1.67	33	0	4.47 ± 5.67	0	18	53	38.87 ± 4.54	29	0	3.17 ± 2.18	0	9
11	22.27 ± 2.14	16	0	6.13 ± 1.26	4	0	54	96.27 ± 4.14	88	0	42.00 ± 3.75	33	0
12	11.30 ± 1.35	9	0	2.53 ± 1.93	0	11	55	31.93 ± 3.45	21	0	5.87 ± 2.17	0	2
13	20.67 ± 2.07	16	0	8.40 ± 4.12	0	5	56	104.97 ± 5.44	94	0	61.43 ± 4.26	50	0
14	11.20 ± 1.66	7	0	0.27 ± 1.00	0	28	57	72.53 ± 4.81	59	0	44.57 ± 1.65	41	0
15	12.07 ± 2.03	8	0	0.00 ± 0.00	0	30	58	214.20 ± 4.93	200	0	137.03 ± 6.52	111	0
16	16.17 ± 2.08	12	0	6.13 ± 1.06	4	0	59	110.00 ± 4.82	102	0	58.27 ± 3.85	51	0
17	28.53 ± 2.20	22	0	9.63 ± 1.58	6	0	60	25.50 ± 3.51	17	0	3.43 ± 1.84	0	6
18	19.67 ± 1.96	14	0	2.67 ± 1.89	0	10	61	73.53 ± 4.81	63	0	49.53 ± 4.11	41	0
19	78.00 ± 2.49	73	0	45.50 ± 3.38	35	0	62	116.73 ± 5.07	102	0	48.60 ± 3.42	42	0
20	19.17 ± 2.18	15	0	4.10 ± 1.35	0	2	63	41.77 ± 3.29	35	0	18.47 ± 2.92	13	0
21	38.77 ± 3.09	33	0	13.37 ± 2.26	6	0	64	50.37 ± 4.20	40	0	20.63 ± 3.01	13	0
22	40.47 ± 2.05	35	0	19.80 ± 2.17	15	0	65	24.13 ± 2.46	20	0	6.97 ± 2.51	4	0
23	16.30 ± 1.62	13	0	5.00 ± 1.18	4	0	66	51.70 ± 4.18	43	0	7.40 ± 2.56	0	1
24	48.87 ± 4.57	34	0	23.30 ± 7.20	0	1	67	33.90 ± 6.15	15	0	3.33 ± 2.36	0	9
25	19.20 ± 3.52	12	0	0.00 ± 0.00	0	30	68	26.67 ± 3.62	17	0	9.73 ± 2.35	4	0
26	15.37 ± 1.80	9	0	2.00 ± 2.00	0	15	69	38.47 ± 3.39	31	0	5.43 ± 2.06	0	1
27	46.27 ± 3.00	41	0	17.00 ± 2.42	11	0	70	64.90 ± 3.83	56	0	36.90 ± 3.16	29	0
28	59.83 ± 3.88	52	0	22.70 ± 2.52	16	0	71	76.73 ± 3.92	69	0	44.70 ± 3.28	37	0
29	24.00 ± 2.52	19	0	3.87 ± 1.73	0	4	72	125.93 ± 6.71	106	0	59.17 ± 5.88	46	0
30	17.63 ± 2.51	13	0	0.67 ± 1.49	0	25	73	53.83 ± 3.85	43	0	22.50 ± 3.28	17	0
31	20.63 ± 2.37	16	0	6.37 ± 1.94	4	0	74	28.00 ± 6.10	14	0	5.70 ± 2.58	0	3
32	26.97 ± 2.47	20	0	8.63 ± 1.74	5	0	75	66.43 ± 4.98	53	0	19.10 ± 2.75	13	0
33	43.53 ± 2.28	37	0	20.67 ± 2.56	15	0	76	60.73 ± 4.05	48	0	30.60 ± 4.57	21	0
34	40.87 ± 2.60	35	0	16.43 ± 2.68	10	0	77	153.00 ± 6.62	131	0	67.27 ± 5.53	54	0
35	36.20 ± 3.91	28	0	6.13 ± 1.67	4	0	78	43.30 ± 3.57	35	0	16.37 ± 3.02	11	0
36	135.73 ± 3.86	128	0	84.43 ± 4.10	72	0	79	39.20 ± 4.46	27	0	14.60 ± 3.53	9	0
37	27.83 ± 3.61	20	0	3.53 ± 1.43	0	4	80	83.07 ± 6.50	70	0	24.83 ± 3.22	17	0
38	29.00 ± 4.93	16	0	0.13 ± 0.72	0	29	81	134.73 ± 5.88	122	0	100.77 ± 5.04	87	0
39	67.13 ± 4.35	57	0	28.17 ± 2.00	23	0	82	231.17 ± 5.85	219	0	175.60 ± 6.37	160	0
40	19.40 ± 3.02	11	0	2.67 ± 1.89	0	10	83	312.40 ± 6.15	300	0	206.10 ± 6.14	194	0
41	19.23 ± 2.43	14	0	4.10 ± 1.60	0	3	84	42.53 ± 5.08	34	0	7.70 ± 2.64	0	1
42	26.90 ± 3.22	20	0	4.53 ± 1.67	0	2	85	102.07 ± 4.68	94	0	57.73 ± 4.63	44	0
43	85.57 ± 3.60	79	0	34.93 ± 3.22	28	0	86	167.60 ± 6.52	150	0	98.57 ± 5.78	88	0

$p_X = .9$ and $p_M = 1/\ell$ (where $\ell = vb$ is the size of individuals) respectively, and $f_{\%} = 10\%$.

Tables 2-4 show all results. Regarding the initialization procedure, bit-flip-based algorithms have been tested both with purely random initialization and guided initialization (i.e., enforcing row constraints). The guided initialization resulted in worse results for HC, indistinguishable results for TS, and better results for GA in most instances (in all cases with statistical significance at the standard 0.05 level according to a Wilcoxon ranksum test). This can be explained by the inferior exploration capabilities of HC_{bf} when starting from a solution satisfying row constraints (the search will be confined to a narrow path uphill). This consideration is unimportant for TS_{bf}, since it can easily make downhill moves to keep on exploring. The GA_{bf} benefits however from having a diverse population of higher quality than random. We have therefore opted for reporting the results of HC_{bf} and TS_{bf} with random initialization, and the results of GA_{bf} with guided initialization.

A summary of performance is provided in Table 5, showing the number of problem instances (out of 86) that were solved in at least one run by each

Table 3. Results of TS algorithms (30 runs per instance). $\overline{x}$, σ, B and S denote, respectively, the fitness average value, the standard deviation, the best obtained result, and the number of times that a problem instance solution is obtained.

	TS_{bf}			TS_{sw}				TS_{bf}			TS_{sw}		
ID	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S	ID	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S
1	3.30 ± 2.62	0	10	0.00 ± 0.00	0	30	44	108.80 ± 6.48	96	0	66.70 ± 9.48	22	0
2	6.93 ± 6.06	0	12	0.00 ± 0.00	0	30	45	17.53 ± 2.64	9	0	4.30 ± 1.44	0	2
3	6.17 ± 2.40	0	2	0.00 ± 0.00	0	30	46	58.37 ± 4.32	50	0	32.37 ± 1.74	29	0
4	4.30 ± 1.68	0	3	0.00 ± 0.00	0	30	47	16.87 ± 4.13	8	0	2.43 ± 1.99	0	12
5	3.67 ± 4.81	0	18	0.00 ± 0.00	0	30	48	24.23 ± 3.50	18	0	8.70 ± 1.72	4	0
6	7.33 ± 2.53	4	0	0.00 ± 0.00	0	30	49	1.50 ± 2.01	0	19	0.00 ± 0.00	0	30
7	1.67 ± 2.10	0	18	0.00 ± 0.00	0	30	50	23.43 ± 3.96	16	0	6.03 ± 1.35	4	0
8	28.60 ± 5.37	15	0	0.00 ± 0.00	0	30	51	8.70 ± 2.42	0	1	0.00 ± 0.00	0	30
9	9.27 ± 2.86	4	0	0.00 ± 0.00	0	30	52	14.37 ± 2.95	10	0	0.40 ± 1.20	0	27
10	27.13 ± 7.10	0	1	0.00 ± 0.00	0	30	53	6.20 ± 3.26	0	3	0.00 ± 0.00	0	30
11	13.50 ± 3.39	5	0	0.00 ± 0.00	0	30	54	50.77 ± 5.97	39	0	24.53 ± 2.05	20	0
12	3.93 ± 2.73	0	8	0.00 ± 0.00	0	30	55	6.83 ± 2.19	4	0	0.00 ± 0.00	0	30
13	16.60 ± 4.10	8	0	0.00 ± 0.00	0	30	56	68.33 ± 4.83	51	0	40.47 ± 2.68	35	0
14	3.13 ± 2.05	0	8	0.00 ± 0.00	0	30	57	43.17 ± 2.57	38	0	40.13 ± 0.43	40	0
15	1.60 ± 2.39	0	19	0.00 ± 0.00	0	30	58	160.80 ± 8.58	145	0	100.43 ± 3.85	91	0
16	10.60 ± 2.24	4	0	0.00 ± 0.00	0	30	59	66.53 ± 4.08	56	0	35.90 ± 2.53	30	0
17	16.33 ± 3.25	8	0	2.93 ± 1.77	0	8	60	2.47 ± 2.12	0	12	0.00 ± 0.00	0	30
18	7.27 ± 2.79	0	1	0.00 ± 0.00	0	30	61	54.60 ± 5.22	43	0	14.40 ± 11.64	0	10
19	59.27 ± 5.28	50	0	0.37 ± 1.97	0	29	62	65.37 ± 6.74	54	0	29.77 ± 1.80	26	0
20	8.37 ± 2.44	4	0	0.00 ± 0.00	0	30	63	20.73 ± 3.59	13	0	5.17 ± 1.44	0	1
21	22.70 ± 3.94	15	0	5.77 ± 0.88	4	0	64	23.53 ± 2.70	19	0	8.13 ± 1.69	4	0
22	27.43 ± 3.23	23	0	4.70 ± 4.41	0	12	65	5.50 ± 2.03	0	1	0.00 ± 0.00	0	30
23	8.33 ± 2.29	4	0	0.00 ± 0.00	0	30	66	12.57 ± 4.11	5	0	0.00 ± 0.00	0	30
24	33.33 ± 12.10	0	2	0.00 ± 0.00	0	30	67	2.23 ± 2.56	0	16	0.00 ± 0.00	0	30
25	1.43 ± 1.99	0	19	0.00 ± 0.00	0	30	68	7.37 ± 2.11	4	0	0.00 ± 0.00	0	30
26	3.77 ± 2.35	0	7	0.00 ± 0.00	0	30	69	7.27 ± 2.54	0	1	0.13 ± 0.72	0	29
27	26.10 ± 4.04	18	0	8.13 ± 1.82	4	0	70	40.53 ± 3.87	32	0	18.23 ± 1.84	14	0
28	34.70 ± 3.91	27	0	11.70 ± 1.51	9	0	71	47.37 ± 4.42	38	0	24.17 ± 2.60	17	0
29	8.57 ± 2.67	4	0	0.00 ± 0.00	0	30	72	70.63 ± 5.49	61	0	36.77 ± 2.60	32	0
30	2.67 ± 2.34	0	12	0.00 ± 0.00	0	30	73	21.90 ± 3.18	17	0	8.30 ± 1.73	4	0
31	8.47 ± 2.40	4	0	0.00 ± 0.00	0	30	74	2.20 ± 2.70	0	17	0.00 ± 0.00	0	30
32	12.93 ± 2.05	9	0	0.27 ± 1.00	0	28	75	24.90 ± 4.72	15	0	5.63 ± 2.33	0	2
33	25.63 ± 3.40	20	0	9.37 ± 1.87	4	0	76	32.00 ± 3.53	24	0	14.13 ± 2.53	9	0
34	22.67 ± 2.66	18	0	6.70 ± 1.39	4	0	77	84.23 ± 8.92	68	0	43.87 ± 2.20	41	0
35	12.70 ± 3.25	6	0	0.00 ± 0.00	0	30	78	16.47 ± 2.80	11	0	3.17 ± 2.07	0	8
36	100.07 ± 6.39	85	0	49.47 ± 18.59	0	3	79	15.00 ± 3.17	10	0	1.43 ± 2.06	0	20
37	6.13 ± 2.63	0	2	0.00 ± 0.00	0	30	80	32.00 ± 5.14	22	0	10.23 ± 2.01	6	0
38	2.33 ± 2.83	0	17	0.00 ± 0.00	0	30	81	109.23 ± 7.99	89	0	22.90 ± 18.70	0	12
39	36.27 ± 3.98	29	0	15.47 ± 1.78	12	0	82	184.40 ± 6.34	173	0	134.13 ± 5.08	124	0
40	3.30 ± 2.69	0	11	0.00 ± 0.00	0	30	83	230.37 ± 10.02	207	0	159.63 ± 5.92	148	0
41	5.47 ± 1.65	4	0	0.00 ± 0.00	0	30	84	7.60 ± 3.59	0	1	0.50 ± 1.28	0	26
42	7.47 ± 3.31	4	0	0.00 ± 0.00	0	30	85	61.90 ± 4.47	54	0	34.80 ± 2.56	31	0
43	48.80 ± 5.53	39	0	20.47 ± 1.87	16	0	86	107.70 ± 5.54	99	0	65.70 ± 3.64	56	0

of the algorithms, and the corresponding success percentage. As expected, TS variants outperform their HC counterparts. Note also that results of local search algorithms are considerably improved when considering the swap neighborhood. The difference is not so marked in the case of GAs, although GA_{sw} still manages to solve more instances than GA_{bf}. In global terms, TS_{sw} outperforms the rest of techniques, including NN-SA and CLS. In fact, TS_{sw} can solve every instance solved by CLS (i.e., the technique that had reported the best results on the problem), as well as instances $\langle 23, 23, 11, 11, 5\rangle$ and $\langle 15, 60, 28, 7, 12\rangle$.

A more fine-grained comparison of the algorithms considered is provided in Table 6. This table shows the percentage of instances in which a certain algorithm performs better (again, with statistical significance at the 0.05 level according to a Wilcoxon ranksum test) than another certain one (note that entries (i, j) and (j, i) in this table do not necessarily sum 100%, since there are instances on which there is no significant difference between the algorithms compared). As it can be seen, swap-based algorithms are consistently better than bit-flip-based algorithms (above 75% in almost all cases). Regarding the GAs, note GA_{sw} is better than GA_{bf} in about 78% of the runs, a larger difference than the number

Table 4. Results of GAs (30 runs per instance). $\overline{x}$, σ, B and S denote, respectively, the fitness average value, the standard deviation, the best obtained result, and the number of times that a problem instance solution is obtained.

	GA_{bf}			GA_{sw}				GA_{bf}			GA_{sw}		
ID	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S	ID	$\bar{x} \pm \sigma$	B	S	$\bar{x} \pm \sigma$	B	S
1	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	44	113.43 ± 5.44	104	0	86.80 ± 5.72	76	0
2	0.37 ± 1.97	0	29	0.00 ± 0.00	0	30	45	15.07 ± 3.17	8	0	11.83 ± 2.03	8	0
3	0.77 ± 1.75	0	25	0.00 ± 0.00	0	30	46	56.83 ± 5.41	47	0	47.23 ± 4.42	40	0
4	1.07 ± 1.77	0	22	0.27 ± 1.00	0	28	47	13.77 ± 3.29	7	0	14.00 ± 2.28	10	0
5	0.00 ± 0.00	0	30	0.00 ± 0.00	0	30	48	25.33 ± 3.65	19	0	21.00 ± 3.16	17	0
6	2.60 ± 2.17	0	12	0.13 ± 0.72	0	29	49	1.97 ± 2.36	0	17	4.00 ± 2.27	0	5
7	0.10 ± 0.54	0	29	0.00 ± 0.00	0	30	50	25.70 ± 4.13	17	0	18.63 ± 3.18	13	0
8	11.20 ± 8.82	0	10	4.20 ± 6.43	0	21	51	7.33 ± 2.51	4	0	8.60 ± 3.17	4	0
9	5.37 ± 1.76	0	1	3.60 ± 1.70	0	5	52	14.90 ± 3.34	9	0	12.03 ± 2.64	7	0
10	14.67 ± 7.11	0	4	8.47 ± 6.48	0	11	53	4.80 ± 2.87	0	6	6.00 ± 2.46	0	2
11	8.47 ± 2.36	0	1	4.87 ± 1.41	0	1	54	56.23 ± 6.01	44	0	39.83 ± 5.70	25	0
12	2.10 ± 2.13	0	15	1.07 ± 1.77	0	22	55	7.37 ± 3.02	0	1	6.90 ± 2.01	3	0
13	10.20 ± 4.46	0	4	4.40 ± 5.46	0	18	56	72.70 ± 7.34	61	0	60.27 ± 5.40	51	0
14	1.60 ± 1.96	0	18	0.63 ± 1.43	0	25	57	45.97 ± 2.95	40	0	43.73 ± 1.79	40	0
15	0.73 ± 1.48	0	24	0.00 ± 0.00	0	30	58	171.43 ± 8.58	154	0	131.93 ± 7.23	118	0
16	6.47 ± 1.71	4	0	5.60 ± 1.23	4	0	59	70.23 ± 6.72	58	0	56.30 ± 6.95	46	0
17	11.43 ± 2.26	7	0	7.17 ± 1.83	4	0	60	4.40 ± 2.39	0	5	5.43 ± 2.74	0	4
18	4.03 ± 2.06	0	5	1.40 ± 1.85	0	19	61	51.23 ± 5.17	41	0	46.57 ± 5.24	36	0
19	55.10 ± 3.94	47	0	31.27 ± 8.52	0	1	62	70.50 ± 8.35	50	0	49.63 ± 5.31	39	0
20	6.17 ± 2.05	4	0	2.87 ± 1.89	0	9	63	20.10 ± 3.62	13	0	19.13 ± 3.43	13	0
21	17.00 ± 2.71	11	0	9.97 ± 1.78	6	0	64	25.33 ± 4.41	16	0	20.67 ± 3.28	15	0
22	23.30 ± 3.63	12	0	15.57 ± 2.39	10	0	65	5.90 ± 2.33	0	1	8.23 ± 2.26	4	0
23	5.00 ± 2.03	0	3	2.87 ± 1.93	0	9	66	12.40 ± 3.24	7	0	10.60 ± 2.70	6	0
24	17.67 ± 12.38	0	8	7.47 ± 10.00	0	19	67	3.93 ± 2.28	0	6	5.20 ± 1.66	0	1
25	0.80 ± 1.62	0	24	0.00 ± 0.00	0	30	68	8.50 ± 3.29	4	0	10.53 ± 2.31	7	0
26	2.50 ± 2.39	0	14	1.13 ± 1.75	0	21	69	9.13 ± 2.73	4	0	9.20 ± 3.11	0	1
27	22.80 ± 3.33	17	0	13.80 ± 2.09	9	0	70	39.20 ± 4.83	32	0	36.23 ± 4.33	25	0
28	29.60 ± 3.59	23	0	17.43 ± 2.74	12	0	71	50.53 ± 5.04	36	0	41.63 ± 5.20	33	0
29	5.43 ± 2.80	0	4	3.40 ± 2.01	0	7	72	82.23 ± 7.28	69	0	58.20 ± 5.07	46	0
30	1.30 ± 1.86	0	20	0.27 ± 1.00	0	28	73	27.87 ± 4.57	17	0	23.03 ± 3.70	15	0
31	7.60 ± 2.44	4	0	4.77 ± 1.71	0	2	74	5.20 ± 2.54	0	1	7.17 ± 3.14	0	2
32	10.03 ± 2.77	4	0	6.70 ± 2.00	4	0	75	29.57 ± 4.51	22	0	20.27 ± 2.45	16	0
33	23.30 ± 2.93	19	0	15.93 ± 2.31	11	0	76	35.40 ± 4.42	25	0	29.30 ± 3.80	21	0
34	19.00 ± 3.11	13	0	12.63 ± 1.85	10	0	77	101.73 ± 10.25	76	0	65.90 ± 5.02	58	0
35	10.57 ± 2.51	5	0	4.43 ± 1.61	0	2	78	19.93 ± 4.30	11	0	18.33 ± 3.60	12	0
36	103.27 ± 5.50	94	0	73.53 ± 5.00	60	0	79	16.40 ± 3.57	9	0	17.00 ± 3.20	10	0
37	3.53 ± 2.59	0	9	2.00 ± 1.93	0	14	80	37.60 ± 6.40	26	0	26.33 ± 4.75	18	0
38	1.23 ± 2.25	0	22	0.37 ± 1.11	0	27	81	107.37 ± 10.77	80	0	97.93 ± 9.80	78	0
39	37.97 ± 4.28	29	0	22.27 ± 2.45	17	0	82	200.70 ± 9.84	185	0	174.17 ± 7.86	157	0
40	2.60 ± 2.33	0	13	1.27 ± 1.81	0	20	83	261.47 ± 7.67	244	0	200.63 ± 8.21	182	0
41	4.00 ± 2.13	0	5	2.33 ± 2.09	0	13	84	9.27 ± 3.76	0	2	9.73 ± 2.78	5	0
42	6.50 ± 2.26	0	1	3.00 ± 2.05	0	9	85	67.10 ± 6.23	55	0	58.53 ± 5.96	48	0
43	47.23 ± 5.68	38	0	28.93 ± 3.19	24	0	86	125.20 ± 11.65	107	0	94.90 ± 8.25	81	0

Table 5. Number and percentage of solved instances for each algorithm on the 86 instances considered.

NN-SA	CLS	HC_{bf}	HC_{sw}	TS_{bf}	TS_{sw}	GA_{bf}	GA_{sw}
16	55	4	35	27	**57**	35	37
(18.60%)	(63.95%)	(4.65%)	(40.70%)	(31.40%)	**(66.28%)**	(40.70%)	(43.02%)

of solved instances. Finally, TS_{sw} is the clear winner, beating the remaining algorithms in 78%-100% of instances.

5 Conclusions and Future Work

The application of metaheuristics to the design of balanced incomplete blocks has resulted in very encouraging and positive results. An empirical evaluation of three different techniques (i.e., a hill climbing method, a tabu search algorithm, and a genetic algorithm), with two variants each, has shown that highly competitive results can be achieved. Furthermore, a TS algorithm working on the

Table 6. Summary of statistical significance results. Each entry in the table indicates the percentage of instances in which the algorithm labelled in the row outperforms the algorithm labelled in the column, with a statistically significant difference according to a Wilcoxon ranksum test.

	HC_{bf}	HC_{sw}	TS_{bf}	TS_{sw}	GA_{bf}	GA_{sw}
HC_{bf}	–	0.00%	0.00%	0.00%	0.00%	0.00%
HC_{sw}	100%	–	76.64%	0.00%	61.63%	12.79%
TS_{bf}	95.35%	12.79%	–	0.00%	27.91%	11.63%
TS_{sw}	100%	97.67%	100.00%	–	86.05%	77.91%
GA_{bf}	100%	10.47%	43.02%	11.63%	–	5.81%
GA_{sw}	100%	47.67%	81.40%	17.44%	77.91%	–

swap neighborhood has been shown to be competitive to an ad-hoc constrained local search (CLS) method, the current incumbent for this problem.

In addition, our analysis also indicates the relevance of the neighborhood structure chosen. The swap neighborhood provides better navigational capabilities than the bit-flip neighborhood, regardless how initial solutions are chosen in the latter. However, this does not imply the bit-flip neighborhood is not appropriate for this problem. For example, we believe a hybrid approach that combine both neighborhoods –e.g., in a variable neighborhood search framework– would be of the foremost interest. Work is in progress in this line. This hybridization can be also done from the algorithmic point of view, i.e., a memetic combination of TS and GAs. The form of this combination is an issue of further work.

Acknowledgements

This work is partially supported by projects TIN2008-05941 (of MICIIN) and P06-TIC2250 (from Andalusian Regional Government).

References

1. Colbourn, C., Dinitz, J., eds.: The CRC handbook of combinatorial designs. Boca Raton: CRC Press (1996)
2. van Lint, J., Wilson, R.: A Course in Combinatorics. Cambridge University Press (1992)
3. Mead, R.: Design of Experiments: Statistical Principles for Practical Applications. Cambridge University Press (1993)
4. Buratti, M.: Some (17q, 17, 2) and (25q, 25, 3)BIBD constructions. Designs, Codes and Cryptography **16**(2) (1999) 117–120
5. Lan, L., Tai, Y.Y., Lin, S., Memari, B., Honary, B.: New constructions of quasi-cyclic LDPC codes based on special classes of BIDBs for the AWGN and binary erasure channels. IEEE Transactions on Communications **56**(1) (2008) 39–48
6. Corneil, D.G., Mathon, R.: Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations. Annals of Discrete Mathematics **2** (1978) 1–32

7. Gibbons, P., Östergård, P.: Computational methods in design theory. In: [1]. 730–740
8. Bofill, P., Guimerà, R., Torras, C.: Comparison of simulated annealing and mean field annealing as applied to the generation of block designs. Neural Networks **16**(10) (2003) 1421–1428
9. Prestwich, S.: A local search algorithm for balanced incomplete block designs. In Rossi, F., ed.: 9th International Conference on Principles and Practices of Constraint Programming (CP2003). LNCS, Springer (2003) 53–64
10. Cochran, W.G., Cox, G.M.: Experimental Design. John Wiley, New York (1957)
11. Fisher, R.A., Yates, F.: Statistical Tables for Biological, Agricultural and Medical Research. 3 edn. Oliver & Boy (1949)
12. Fisher, R.A.: An examination of the different possible solutions of a problem in incomplete blocks. Annals of Eugenics **10** (1940) 52–75
13. Whitaker, D., Triggs, C.M., John, J.A.: Construction of block designs using mathematical programming. J. Roy. Statist. Soc. B **52**(3) (1990) 497–503
14. John, J.A., Whitaker, D., Triggs, C.M.: Construction of cyclic designs using integer programming. Journal of statistical planning and inference **36**(2) (1993) 357–366
15. Zergaw, D.: A sequential method of constructing optimal block designs. Australian & New Zealand Journal of Statistics **31** (1989) 333–342
16. Tjur, T.: An algorithm for optimization of block designs. Journal of Statistical Planning and Inference **36** (1993) 277–282
17. Flener, P., Frisch, A.M., Hnich, B., Kzltan, Z., Miguel, I., Walsh, T.: Matrix modelling. In: CP-01 Workshop on Modelling and Problem Formulation. International Conference on the Principles and Practice of Constraint Programming. (2001)
18. Puget, J.F.: Symmetry breaking revisited. In Hentenryck, P.V., ed.: 8th International Conference on Principles and Practice of Constraint Programming (CP 2002). Volume 2470 of LNCS., Ithaca, NY, USA, Springer (2002) 446–461
19. Meseguer, P., Torras, C.: Exploiting symmetries within constraint satisfaction search. Artif. Intell. **129**(1-2) (2001) 133–163
20. Prestwich, S.: Negative effects of modeling techniques on search performance. Annals of Operations Research **18** (2003) 137–150
21. Street, D., Street, A.: Partially balanced incomplete block designs. In: [1]. 419–423
22. Mullin, C., Gronau, H.: PBDs and GDDs: The basics. In: [1]. 185–193
23. Wallis, W.: Regular graph designs. Journal of Statistical Planning and Inference **51** (1996) 272–281
24. Bofill, P., Torras, C.: MBMUDs: a combinatorial extension of BIBDs showing good optimality behaviour. Journal of Statistical Planning and Inference **124**(1) (2004) 185–204
25. Chuang, H.Y., Tsai, H.K., Kao, C.Y.: Optimal designs for microarray experiments. In: 7th International Symposium on Parallel Architectures, Algorithms, and Networks, Hong Kong, China, IEEE Computer Society (2004) 619–624
26. Morales, L.B.: Constructing difference families through an optimization approach: Six new BIBDs. Journal of Combinatorial Design **8**(4) (2000) 261–273
27. Angelis, L.: An evolutionary algorithm for A-optimal incomplete block designs. Journal of Statistical Computation and Simulation **73**(10) (2003) 753–771