Memetic collaborative approaches for finding balanced incomplete block designs ☆

David Rodríguez Rueda[a], Carlos Cotta[b], Antonio J. Fernández-Leiva[b]

[a] *Universidad Nacional Experimental del Táchira (UNET), Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Táchira, 5001, Venezuela*
[b] *Universidad de Málaga, ETSI Informática, Campus de Teatinos, 29071 Málaga, Spain*

Abstract

The balanced incomplete block design (BIBD) problem is a difficult combinatorial problem with a large number of symmetries, which add complexity to its resolution. In this paper, we propose a dual (integer) problem representation that serves as an alternative to the classical binary formulation of the problem. We attack this problem incrementally: firstly, we propose basic algorithms (i.e. local search techniques and genetic algorithms) intended to work separately on the two different search spaces (i.e. binary and integer); secondly, we propose two hybrid schemes: an integrative approach (i.e. a memetic algorithm) and a collaborative model in which the previous methods work in parallel, occasionally exchanging information. Three distinct two-dimensional structures are proposed as communication topology among the algorithms involved in the collaborative model, as well as a number of migration and acceptance criteria for sending and receiving data. An empirical analysis comparing a large number of instances of our schemes (with algorithms possibly working on different search spaces and with/without symmetry breaking methods) shows that some of these algorithms can be considered the state of the art of the metaheuristic methods applied to finding BIBDs. Moreover, our cooperative proposal is a general scheme from which distinct algorithmic variants can be instantiated to handle symmetrical optimisation problems. For this reason, we have also analysed its key parameters, thereby providing general guidelines for the design of efficient/robust cooperative algorithms devised from our proposal.

Keywords: Balanced Incomplete Block Design, Memetic Algorithms, Cooperative Models, Metaheuristics

1. Introduction

The generation of block designs is a well-known combinatorial problem of enormous difficulty [1]. The problem has a number of variants [2, 3, 4, 5, 6], among which a popular one is the so-called balanced incomplete block design (BIBD). Basically, a BIBD is defined as an arrangement of v different objects into b blocks such that each block contains exactly k different objects, each object occurs in exactly r different blocks, and every two different objects occur together in exactly λ blocks (for $k, r, \lambda > 0$). The construction of BIBDs was initially tackled in the area of experimental design [7, 8]; however, nowadays BIBDs are applied in a variety of fields such as cryptography [9], coding theory [10], food evaluation [11], load balance in distributed networks [12], and classification tasks [13], among others.

BIBD generation is an NP-hard problem [14] that provides an excellent benchmark for optimisation algorithms since it is scalable and has a wide variety of problem instances ranging from easy instances to very difficult ones. As discussed in Sect. 2.2, complete methods (including exhaustive search) have been applied to the problem although it remains intractable even for designs of a relatively small size [15]. In fact, as proof of the difficulty of the problem, there are currently a number of open instances that have not yet been solved (although, it may be that there is no solution for them; then again, non-solvability cannot be established by complete methods). The application of metaheuristics thus seems to be appropriate to tackle larger problem instances due to the limitations of complete methods. Indeed, some approaches in this area have already provided evidence of the potential of metaheuristic approaches applied to this problem, e.g. [16, 17, 18, 19].

One of the most interesting features of the BIBD is its highly-symmetrical nature. This introduces a number of considerations that have to be taken into account. Firstly, the existence of solutions that are equivalent with respect to the same representation space generally increases the size of the search space and, as a direct consequence, the difficulty of finding solutions (i.e. the problem solving complexity). In the last few decades, a number of methods have been applied to deal with symmetries [20, 21, 22, 23]. The primary method of dealing with them consists in applying some symmetry breaking technique. This method basically imposes new constraints to remove symmetries with the goal of reducing the problem's search space. Symmetry breaking can be applied in many diverse forms [24]. In connection with this, it is also well known that the encoding of solutions can drastically affect the search process, be-

☆This work is partially funded by Junta de Andalucía (project P10-TIC-6083, DNEMESIS – `http://dnemesis.lcc.uma.es/wordpress/`), Ministerio Español de Economía y Competitividad (projects TIN2014-56494-C4-1-P, UMA::EPHEMECH – `https://ephemech.wordpress.com/` and TIN2017-85727-C4-1-P, UMA::DeepBio – `http://deepbio.wordpress.com`), and Universidad de Málaga, Campus de Excelencia Internacional Andalucía Tech.

Email addresses: `drodri@unet.edu.ve` (David Rodríguez Rueda), `ccottap@lcc.uma.es` (Carlos Cotta), `afdez@lcc.uma.es` (Antonio J. Fernández-Leiva)

cause it influences the underlying landscape and its navigability. This paper proposes an alternative –and novel, to the best of our knowledge– representation scheme for BIBD solutions that we call the dual (or decimal) formulation (see Sect. 4.1), in response to the 'more natural' primal (or binary) model considered in the scientific literature, cf. Sect 2.1. A number of algorithms to tackle the BIBD problem are subsequently considered to take into account the large number of possible scenarios that arise from the combination of these two different encodings, as well as the symmetry-breaking constraints for the BIBD problem (see Sect. 4.2). Moreover, each scenario is tackled with a number of metaheuristic techniques, including local search and genetic algorithms. As a further step, this paper also proposes mechanisms for hybridising these algorithms. In particular, we consider both an integrative model (Sect. 3) and a collaborative scheme (Sect. 4.3). The latter, in particular, defines a network (i.e. a set) of algorithms that intensify the search in certain parts of the search space; the communication strategy among these algorithms is defined by a certain spatial structure. Three different topologies are considered for this purpose. We also study different policies to control communication among algorithms, i.e. which information should be submitted and when/how it should be handled by the metaheuristics in the network. The resulting techniques are exhaustively analysed from an empirical point of view in Sect. 5. The next section provides an overview of the problem's foundations as well as a brief look at related work.

This paper proposes a (novel) formulation for the representation of BIBDs and a number of metaheuristics (based on this formulation) to handle the problem. This paper also describes a large number of metaheuristic approaches to deal with the generation of BIBDs. Some of these (i.e. the cooperative methods) constitute state-of-the-art metaheuristic methods to handle the problem. Moreover, we provide a general scheme from which other (possibly cooperative) metaheuristics can be generated. Finally, we also propose a methodology to address, in a general way, symmetrical combinatorial problems so that our methods can be easily adjusted to deal with other symmetrical combinatorial problems.

2. Background

This section discusses how the BIBD problem has been tackled in the literature. The formal classical formulation of the problem is also provided.

2.1. BIBD: Formulation and primal (or binary) model

As mentioned in the introduction, a BIBD can be specified with five parameters $\langle v,b,r,k,\lambda\rangle$. Using this notation, a $\langle v,b,r,k,\lambda\rangle$-BIBD problem consists of dividing a set of v objects into b subsets of $k<v$ objects each, such that each object belongs to r different subsets and any pair of objects appear together in exactly $\lambda<b$ subsets. A standard way of representing the solution to such a problem –termed here as the *primal (or binary) model (B)*– is in terms of its incidence matrix $M\equiv\{m_{ij}\}_{v\times b}$, which is a $v\times b$ binary matrix where $m_{ij}\in\{0,1\}$ is equal to 1 if the ith object is contained in the jth block, and 0 otherwise; thus, it is easy to see that each row corresponds to an object and each column to a block, and that a matrix representing a feasible solution has exactly r ones per row, k ones per column, and the scalar product of any pair of different rows is λ. 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 (i.e. $b=v$) $\langle 7,7,3,3,1\rangle-$BIBD, respectively.

Note that the five parameters defining a $\langle v,b,r,k,\lambda\rangle-$BIBD are interrelated and satisfy the following two relations: $bk=vr$ and $\lambda(v-1)=r(k-1)$. This means that we could define an instance using just three parameters $\langle v,k,\lambda\rangle$ and compute b and r in terms of the former three. However, although these relations restrict the set of admissible parameters for a BIBD, such admissibility is a necessary yet insufficient condition to guarantee its existence [25, 26].

The BIBD problem is a constraint satisfaction problem (CSP) that can be readily transformed into a constraint optimisation problem (COP) by relaxing the problem (allowing the violation of constraints) and defining an objective function that accounts for the number and degree of their violations. More precisely, let $I=\langle v,b,r,k,\lambda\rangle$ and M represent, respectively, the instance values and the (binary) incidence matrix of size $v\times b$ for a BIBD problem. In addition, for the rest of the paper, let $\mathbb{N}_h^+=\{1,...,h\}$ (for any integer number $h\geq 1$). Therefore, the problem of finding a BIBD solution can be formulated as follows:

$$\min\ f^I(M)=\sum_{i=1}^{v}\phi_i(M,r)+\sum_{j=1}^{b}\phi'_j(M,k)+\sum_{i=1}^{v-1}\sum_{j=i+1}^{v}\phi''_{ij}(M,\lambda) \tag{1}$$

where

$$\phi_i(M,r)=\left|r-\sum_{j=1}^{b}m_{ij}\right|, \qquad \forall i\in[1,v] \tag{2}$$

$$\phi'_j(M,k)=\left|k-\sum_{i=1}^{v}m_{ij}\right|, \qquad \forall j\in[1,b] \tag{3}$$

$$\phi''_{ij}(M,\lambda)=\left|\lambda-\sum_{h=1}^{b}m_{ih}m_{jh}\right|, \quad \forall i,j\in[1,v]:i<j \tag{4}$$

We call this formulation *the primal model*, denoted as B because it is based on a binary representation of the candidates to be solved. Note that the required values of the row constraints, column constraints and scalar product constraints correspond with the number of ones per row (i.e. r), the number of ones per column (i.e. k), and the scalar product of any pair of different rows (i.e. λ). So, for each row i (resp. column j) in the incidence matrix, $\phi_i(M,r)$ in Eq. (2) (resp. $\phi'_j(M,k)$ Eq.(3)) computes the discrepancies between the required value r (resp. k) of ones for row i (resp. column j) and the existing number of ones in row i (resp. column j). Also note that for each pair of distinct rows i,j in the incidence matrix, $\phi''_{ij}(M,\lambda)$ in Eq. (4) calculates the discrepancies between the required value λ of the scalar product of the rows (i.e. coincidences of ones placed in

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

(b)

Figure 1: (a) A $\langle 8,14,7,4,3\rangle-$BIBD. (b) A $\langle 7,7,3,3,1\rangle-$symmetric BIBD.

the same positions in both rows) and the computed scalar product of the two rows (i.e. the existing number of ones in the same position in both rows i and j). As a consequence, for a given incidence matrix M, the value returned by the objective function sums up all discrepancies with respect to the required values of the row constraints (i.e. Eq.(2)), column constraints (i.e. Eq.(3)) and scalar product constraints (i.e. Eq.(4)).

Then, a solution to the BIBD problem is a configuration M^* such that $f^1(M^*) = 0$.

2.2. Related work

The BIBD problem has been tackled by a number of different techniques in the literature, with varying levels of success. Traditionally, the problem has been dealt with deterministic, constructive and/or complete methods. For instance, Whitaker *et al.* [27] used mathematical programming methods to look for an optimal incomplete block design. Zergaw [28] also considered the error correlation, and presented a sequential algorithm for constructing optimal block designs. Along the same lines, Tjur [29] incorporated interchange mechanisms with the addition of experimental units (blocks) one by one. Flener *et al.* [30] proposed a matrix model based on ECLIPSE to solve the problem of block generation.

One of the key points of interest in the problem is its symmetrical nature (i.e. rows and columns can be permuted and objects can be relabelled). In this sense, constraint programming is the most frequently used technique to deal with this issue. For instance, Puget [31] formulated the problem as a CSP where each instance was represented by a classical binary matrix of size $v \times b$, and proposed combining methods for *symmetry breaking via dominance detection* and *symmetry breaking using stabilisers* in order to solve the problem. In addition, Meseguer and Torras [24] explored two strategies (namely, a heuristic for variable selection and a domain pruning procedure) to exploit the symmetry of the problem. The underlying idea in this approach was to use symmetries to guide the search for a solution. The objective was not to solve specific instances but rather to be effective in reducing the search effort [1].

Although all these methods can be used to design BIBDs, their applicability is limited by the size of the problem instances. To address this, stochastic methods have also been applied to the problem. For instance, Bofill et al. [32] formulated the generation of BIBD as a combinatorial optimisation problem tackled with a neural network. A simulated annealing algorithm endowed with this neural network (NN-SA) was shown to offer better performance than an analogous hybridisation with mean field annealing. These results were further improved upon by Prestwich [16, 33], who considered different schemes for adding symmetry breaking constraints inside a constrained local search (CLS).

These results were improved by Rodríguez *et al.* [17] who used both local search methods (hill climbing and tabu search) and population-based techniques (genetic algorithms). Two different neighbourhood structures (defined over the primal encoding described in Sect. 2.1) were proposed, one based on bit-flipping and the other on position-swapping. It was shown that the swap-based neighbourhood (sw) was superior to the flip-based neighbourhood, and that tabu search based on position-swapping (TS_{sw}) offered the best performance, being capable not only of beating hill climbing methods –also based on the swap-based neighbourhood– and genetic algorithms (GA_{sw}) –based on both position-swapping and the bitwise uniform crossover operator (UX)– in more than 78% of the instances, but also of solving 57 instances from a selected set of 86 (66.28%), one more than CLS, the best method until that moment. Later, Rodríguez *et al.* [17] explored, in greater depth, the use of population-based methods [19], and more specifically, the use of memetic algorithms (MAs) in the form of a synergistic combination of a genetic algorithm with the use of a heuristic recombination operator and a TS-based local searcher. It was shown that MAs with a greedy recombination operator (Gd), termed MA_{Gd}, performed better than MAs based on the bitwise uniform crossover operator (UX) (as proposed in [17]) as well as its constituent parts TS_{sw} and GA_{sw} (as described in [17]). By increasing the number of evaluations, MA_{Gd} solved 63/86 instances (73.26%), TS_{sw} 59/86, and GA_{Gd} (i.e. a GA with the operator Gd) 48/86. MA_{Gd} can be considered the state-of-the-art (non-commercial) heuristic method.

There are other approaches which address the generation of BIBDs. In particular, there have been several constructive approaches such as the one described by Yokoya and Yamada [34]. These authors made use of the power of the commercial linear programming solver CPLEX and proposed a non-linear mixed-integer programming approach that was shown to be effective in solving BIBD instances. Later, Mandal [35] presented an improved linear integer programming approach that

Table 1: Number (#) and percentage (%) of problem instances solved by the basic and integrative metaheuristics (identified in first column) working alone on the set of 86 instances taken from [16, 32]. The third column indicates the reference in which the method was reported.

algorithm	# (%)	Ref.
NN-SA	16 (18.60%)	[32]
CLS	56 (65.12%)	[33, 16]
TS_{sw}	57 (66.28%)	[17]
GA_{sw}	37 (43.02%)	[17]
TS_{sw}	59 (68.60%)	[19]
GA_{Gd}	48 (55.81%)	[19]
MA_{Gd}	**63** (73.26%)	[19]
TABU-BIBD(20)	78 (90.70%)	[34]
Multi-step	**79** (91.86%)	[35]
Ts+Hc	74 (86.05%)	[36]

handled the problem in an easier way. Along the same lines, Rodríguez *et al.* [36] also proposed a constructive approach, based on local search with multi-start, that provided a greater capacity of exploration of new zones (major diversification) unlike, for example, other approaches that use major intensification. This method has shown good performance, although there were instances for which it was not able to reach optimal solutions. These methods, although efficient in finding solutions, all demand a high computational effort to generate BIBDs. Moreover, they follow a constructive approach that is very different from our metaheuristic proposals. Consequently, they are not considered here in our experimental study.

Table 1 summarises the performance (measured in number of problem instances solved) of all the metaheuristics methods mentioned in this section. We do not provide running times as this information was not reported for all the methods and, in addition, it strongly depends on many external factors.

3. Solving the BIBD with metaheuristics

With the aim of keeping this paper relatively self-contained, this section describes part of our previous work on the application of metaheuristics to the BIBD problem; in particular we outline the work described in [17, 19], where the primal problem representation (i.e. the binary encoding) was used.

In [17], two neighbourhood structures were considered: the first arose naturally from the binary representation of solutions using the incidence matrix M and was based on the Hamming distance (bit-flip (bf)); the second (denoted as swap (sw)) took an object from one block, and moved it to a different one, which can be formulated in binary terms as permuting a 0 and a 1 within the same row. Then, three different techniques based on these two neighbourhood variants, were proposed; more specifically, two hill climbing (HC) methods (i.e. steepest descent procedures termed HC_{bf} and HC_{sw}), two tabu search algorithms (TS_{bf} and TS_{sw}), and two steady-state genetic algorithms based on binary tournament selection and replacement of the worst individual in the population (termed GA_{bf} and GA_{sw}); GA_{bf} used uniform crossover and bit-flip mutation –easy to implement in the binary space, whereas GA_{sw} used uniform crossover at row level (that is, it randomly selects entire rows from either parent) and swap mutation (the interested reader is referred to [17] for more details). For the experiments, 86 instances taken from [16] were used as benchmarks. The best proposal was TS_{sw} which solved 57 instances from those 86, and performed better than CLS [16], the previous best solution method (see Table 1).

Subsequently, in [19] we proposed a memetic algorithm (MA) [37] to finding BIBDs. The resulting MA could be characterised as a steady-state genetic algorithm (GA) –which serves as the underlying population-based search mechanism– based on the *sw* scheme defined before (hence, intrinsically enforcing the row constraint), that incorporates two intensifying components, namely a specific heuristic recombination operator, and a local searcher, in order to guide the search towards promising regions in the search space. More precisely, two different multi-parent recombination operators, one based on uniform crossover and termed UX, and a specific greedy version of the uniform crossover termed Gd, were proposed. The Gd operator starts by creating a set with all available rows in the parents; then (if there are enough different rows; otherwise, standard UX is invoked), it randomly selects an initial row, and subsequently tests all available rows, picking the one which violates fewer scalar-product constraints. It was shown that the recombination operator played an important role in the discovery of new improved solutions. The local search (LS) method used inside the MA to intensify the search was TS_{sw} (mentioned above). More details can be found in [19].

The general scheme of the MA is depicted in Algorithm 1. The input of this algorithm consists of the problem parameters (i.e. v, b, r, k, and λ), as well as the algorithm parameters, i.e the genetic operators and their associated parameters (such as, e.g. application rates); the output of the algorithm is the best individual found during the search process. Parent selection was done randomly using a binary tournament for breeding and replacement of the worst individual in the population. To preserve the diversity in the population, no duplicate solution was accepted, and a re-starting mechanism re-activated the search whenever stagnation occurred. This was done by keeping a fraction $f\%$ of the top individuals in the current population, and refreshing the rest of the population with random individuals (line 19). This procedure was triggered after a number of evaluations without any improvement in the current best solution (n_l). The local search was restricted to explore n_v neighbours, and p_{LS}, p_X and p_M represent the probability of applying local search, the crossover and mutation rates, respectively.

It was shown that algorithms with the Gd operator solve more instances than their counterparts with UX, and MAs also outperform their GA counterparts. The version MA_{Gd} solved 63/86 instances, which even better than TS_{sw}, Even though a higher number of evaluations (i.e. 2×10^7) was considered in [19] than the one given in [17] (i.e, 2×10^6), MA_{Gd} can be considered the best metaheuristic solution reported so far in the literature.

Algorithm 1: Pseudo-code of the memetic algorithm.

```
1  begin
2      for i ← 1 to popsize do
3          pop[i] ← GENERATEMATRIX(v,b,r);
4          EVALUATE(pop[i]);
5      end
6      while numEvals < maxEvals do
7          if rand < p_X then
8              parent_1 ← TOURNAMENTSELECT(pop);
9              ...;
10             parent_m ← TOURNAMENTSELECT(pop);
11             offspring ←
                RECOMBINE(parent_1,...,parent_m);
12         else
13             offspring ← TOURNAMENTSELECT(pop);
14         end
15         offspring ← MUTATE(offspring, p_M);
16         if rand < p_LS then offspring ←
            LOCALSEARCH(offspring, n_v) ;
17         EVALUATE(offspring);
18         pop ← REPLACE(pop, offspring);
19         if stagnation(n_t) then pop ←
            RESTART(pop, f%) ;
20     end
21 end
```

4. A new battery of metaheuristics based on symmetry breaking, dual models and hybridisation

This section presents a number of new metaheuristic proposals to handle the BIBD problem. In Sect. 4.1, we first propose a novel dual problem representation for the BIBD, and a new problem formulation based on it. Next, Sect. 4.2 describes a symmetry-breaking method (that, to the best of our knowledge, is also novel for the problem under consideration in this paper) for both the primal representation (i.e. the classical binary one) and its dual encoding (i.e. the decimal representation). Finally, in Sect. 4.3, we suggest a cooperative scheme as an alternative to the integrative memetic algorithm (Algorithm 1). One of the primary particularities of this cooperative scheme is that it allows the cooperation of algorithms designed to work on different representation models (i.e. primal or dual).

4.1. A dual representation

It is well-known that the representation of candidate solutions can have dramatic effects on the problem solving process, especially in the universe of evolutionary algorithms [38]. For this reason, we consider the concept of *duality* as a way to obtain alternative representations to the natural (and primal) encoding of the solutions to the BIBD problem. Hence, the alternative model we propose is termed *the dual model* or *decimal formulation (D)*; Figure 2 shows an example of a dual representation for a given symmetric BIBD instance. Basically, the solution (or candidate) to the BIBD is now defined by a dual incidence matrix $M^d \equiv \{m^d_{ij}\}_{v\times r}$, which is a $v \times r$ integer matrix

	1	2	3	4	5	6	7
1	0	1	0	1	0	1	0
2	1	0	0	1	0	0	1
3	1	1	1	0	0	0	0
4	0	0	1	0	0	1	1
5	0	1	0	0	1	0	1
6	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0

	1	2	3
2	4	6
1	4	7
1	2	3
3	6	7
2	5	7
1	5	6
3	4	5

Figure 2: Primal/Binary and dual/decimal encodings of the $\langle 7,7,3,3,1\rangle-$symmetric BIBD shown in Figure 1 (b). The number of columns (7 and 3 for the primal and dual representations, respectively) and rows (7 in both cases) are identified for clarity.

where $m^d_{ij} \in \mathbb{N}^+_b$ contains a value from the range $[1,b]$ which identifies a block containing the object i. Note that there are r columns (i.e. $j \in [1,r]$), so that each object i is contained exactly in r blocks if a constraint that all values in a row have to be different is imposed. The dual formulation of the BIBD problem corresponds to a relaxed CSP problem with an objective function that involves the number and degree of violations of constraints defined only on the parameters k and λ as follows:

$$\min\ f^I_d(M^d) = \sum_{j=1}^{b} \psi_j(M^d,k) + \sum_{i=1}^{v-1}\sum_{j=i+1}^{v} \psi'_{ij}(M^d,\lambda) \qquad (5)$$

such that every object should be assigned to r distinct blocks, that is to say:

$$\forall j,h \in \mathbb{N}^+_r : j \neq h \Rightarrow m^d_{ij} \neq m^d_{ih}, \qquad \forall i \in [1,b] \qquad (6)$$

where

$$\psi_j(M^d,k) = \left| k - \sum_{i=1}^{v}\sum_{h=1}^{r} [m^d_{ih} = j] \right|, \qquad \forall j \in [1,b] \qquad (7)$$

$$\psi'_{ij}(M^d,\lambda) = \left| \lambda - \sum_{h=1}^{r}\sum_{l=1}^{r} [m^d_{ih} = m^d_{jl}] \right|, \qquad \forall i,j \in [1,v] : i < j \qquad (8)$$

In Eq. (7) and Eq. (8) we employ the Iverson brackets [] (i.e. [P]=1 if P is true, and 0 otherwise). Observe that, by adding an all-different-value constraint associated with each row – i.e. constraint (6)– each row i now contains the r (required) block assignments of object i, and thus the constraint requiring that each object be placed in r blocks is implicitly present in the new dual model. Therefore, the function to be minimised – i.e. (5) – only has two components. The first one, shown in Eq. (7), sums the discrepancies from the value k. Note that $\sum_{i=1}^{v}\sum_{h=1}^{r}[m^d_{ih} = j]$ quantifies how many times the jth block appears in the solution and each block $j \in [1,b]$ has to contain exactly k objects, something that happens when $\psi_j(M^d,k)$ equals 0. The second component of the objective function, shown in

Eq. (8), computes the discrepancies from the value λ. Observe that each object i has to coincide with any other object j in exactly λ blocks, which means that each two rows of M^d have to share λ blocks (as in the primal model). Hence, we measure the discrepancies between any two objects in (8) with respect to the required value λ. A solution to the BIBD problem is thus a configuration M^{d*} such that $f_d^I(M^{d*}) = 0$.

In this model the neighbourhood is similar to the *swap* version considered for the primal scheme as defined in [17], that is to say, a neighbour of a matrix M^d is any other incidence matrix M'^d obtained from M^d by replacing an element $\phi \in \mathbb{N}_b^+$ contained in a cell m_{ij}^d by any other label in $\mathbb{N}_b^+ \setminus \{\phi\}$, provided constraint (6) is still satisfied.

4.2. Symmetry breaking

According to [22], one way of reducing a problem's symmetries is to transform it into another problem with the same characteristics as the original but eliminating all or most symmetrical states. In the last few decades, a number of methods have been applied to deal with the problem of symmetry [20, 22, 23, 24]. Most of these methods are primarily aimed to reduce the search space of the problem. Other recent works have shown how solving combinatorial problems via mixed integer linear programming approaches can be sped up by adding symmetry breaking constraints to the original formulation. Another idea is to consider asymmetric representatives formulations (ARF) as alternatives to the natural symmetric formulation of the problem. They have been shown to be effective to deal with combinatorial optimisation problems such as p job grouping, binary clustering, node colouring, or blocking experimental designs [39, 40, 41, 42].

In this paper, we consider a symmetry-breaking approach, both for the primal model and for the dual model. This approach is called variable reduction in [42].

4.2.1. Primal (or binary) model

Consider the problem representation introduced in Sect. 2.1. In general, BIBD symmetries arise, first of all, because any two objects are interchangeable in the sense that any two rows can be permuted (i.e. the corresponding objects can have their labels swapped) in the incidence matrix and the resulting candidate will be the same solution. For primal encoding, in particular, this argument can be extended to blocks/columns, as any two columns can be permuted as well. To tackle these symmetries, we impose the following four constraints on the primal problem formulation:

- In Row 1: set $m_{1j} = 1$ for each $j \in \mathbb{N}_r^+$ and $m_{1j} = 0$ for $r < j \leq b$. In other words, place the first object (row 1) in the first r blocks (i.e. the first r columns) so that the row constraint is satisfied for object 1.

- In Row 2: set $m_{2j} = 1$ for $j \in \mathbb{N}_\lambda^+$, $m_{2(\lambda+j')} = 0$ and $m_{2(r+j')} = 1$ for $j' \in \mathbb{N}_{r-\lambda}^+$, and set the other cells in row 2 to 0. This guarantees that the scalar product constraint between rows 1 and 2 is satisfied.

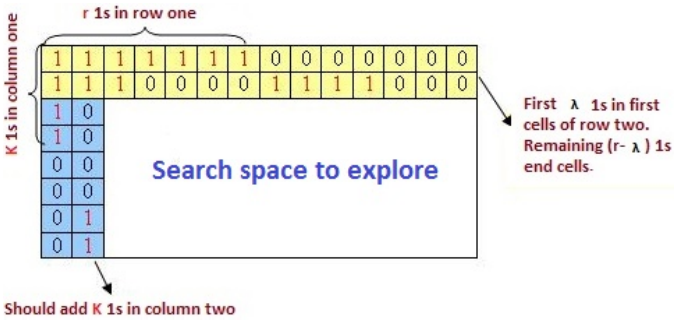

Figure 3: Symmetry breaking in an $\langle 8, 14, 7, 4, 3\rangle$-instance (binary encoding.)

- In Column 1: set $m_{i1} = 1$ for each $i \in \mathbb{N}_k^+$ and $m_{1i} = 0$ for $r < i \leq v$. In other words, place the first k objects in the first block (i.e. the first column) so that the column constraint is satisfied for the first block.

- In Column 2: set $m_{i2} = 1$ for $v - k - (m_{12} + m_{22}) < i \leq v$, and the other values in the column (except the first two rows, to 0). In other words, taking into account that $(m_{12} + m_{22})$ objects have already been placed in block 2, we place the last $k - (m_{12} + m_{22})$ objects in block 2, so that the column constraint is satisfied.

These four constraints can also be viewed as a preset process that fixes the values of the first two rows and the first two columns in the incidence binary matrix $M \equiv \{m_{ij}\}_{v\times b}$ of a particular $\langle v, b, r, k, \lambda\rangle$-instance. This produces a slight reduction of the problem symmetries. Note that the first two rows and the first two columns remain constant in each candidate solution so that they will never be permuted with any other row (resp. column). As a direct consequence, the search space is also reduced. Fixing these two rows/columns means that optimisation only has to be conducted in a binary matrix of size $(v-2)\times(b-2)$. Figure 3 shows an example of how to fix the rows and columns for an $\langle 8, 14, 7, 4, 3\rangle$-instance in the binary problem representation.

4.2.2. Dual (or decimal) model

Now, consider the dual problem representation introduced in Sect. 4.1. Symmetry breaking is achieved by fixing the two first rows of the incident matrix M^d as follows:

- In Row 1: set $m_{1j} = j$ for $j \in \mathbb{N}_r^+$. In other words, the first row contains numbers from 1 to r in sequence. This constraint assures that the first object (i.e. row 1) is placed in the first r blocks and breaks the symmetry of object placement.

- In Row 2: set $m_{2j} = j$ for $j \in \mathbb{N}_\lambda^+$, and set $m_{2j'} = r + j' - \lambda$ for $j' > \lambda$. In this way, the second object is placed in the first λ blocks (where the first object is also placed), and also in $r - \lambda$ blocks other than those in which the first object was placed. This constraint guarantees that the scalar product constraint is satisfied for objects 1 and 2.

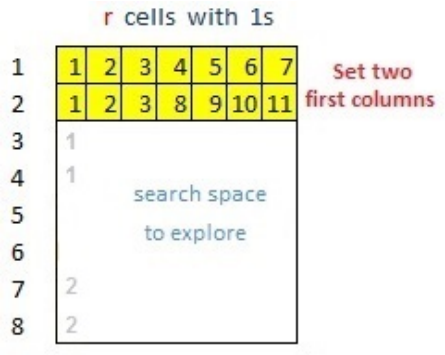

Figure 4: Symmetry breaking in the dual model of an $\langle 8,14,7,4,3\rangle$-instance. The first two rows correspond to the dual representation of the first two rows in the primal representation shown in Figure 3.

The idea is to fix the values of the first two rows in the incidence dual matrix M^d (of size $v \times r$) of a particular $\langle v,b,r,k,\lambda\rangle$-instance so that optimisation only has to be conducted in an integer matrix of size $(v-2)\times r$. Note that, unlike in the primal model, we have not fixed the columns in the dual formulation. The reason is that the first column would only be partially completed anyway, since, in the dual formulation, it is not possible to specify that an object (e.g. objects 5 and 6) is not placed in some specific block (e.g. blocks 1 and 2). This is possible in the primal model by using the value 0 in a cell of the matrix (see Figure 3). In addition, from an implementation point of view, partially completing the first columns would also hinder the natural encoding of individuals as a rectangular matrix. Figure 4 shows an example of how to fix the two first rows in the dual encoding of a problem instance $\langle 8,14,7,4,3\rangle$.

4.3. Cooperative model architecture

The memetic algorithm described in Sect. 3, and presented in Algorithm 1, can be viewed, according to the taxonomy for hybrid and cooperative algorithms given by Puchinger and Raidl [43], as an integrative hybrid algorithm in which a local search is subordinated to the execution of an external genetic algorithm (GA). In other words, local search is executed inside a GA. Puchinger and Raidl presented another interesting scheme, the *collaborative approach* in which several optimisation algorithms are executed in parallel (or sequentially) and exchange information with certain frequency. This kind of cooperation can be considered in itself a programming paradigm comprising two main elements [44]: (a) a set of autonomous programs (usually called agents), each implementing a particular solution method, and (b) a cooperative scheme that combines these autonomous elements into a simple and unified strategy for troubleshooting. In this collaborative approach, the idea is to apply a number of (possibly different) optimisation algorithms each of which explores a specific part of the search landscape through processes of intensification. Next, the agents synchronise from time to time to exchange information. A specific spatial structure (e.g. a ring in which each agent has a predecessor and a successor) identifies the communication topology, that is to say, the way in which this information is transmitted between

Algorithm 2: COOPERATIVE-MODEL$_n$

```
1  for i ∈ ℕ_n^+ do
       // Generation Adjusted to the problem
          model tackled by agent a_i
2      S_i ← GENERATEINITIALPOPULATION();
3  end
4  cycles ← 1;
5  while cycles ≤ Θ do
6      for i ∈ ℕ_n^+ do
           // Population update
7          S_i ← a_i(S_i);
8      end
9      for (i, j) ∈ T_R do
           // Select candidate to migrate via
              the migration policy
10         s_submitted ← SELECTCANDIDATEFROM(S_i) ;
           // Now, test candidate acceptance via
              the acceptance policy
11         if
            ACCEPTSUBMITTEDCANDIDATE(s_submitted, S_j)
            then
               // Selection of candidate to
                  replace
12             s_tobereplaced ←
                SELECTCANDIDATETOREPLACEIN(S_j);
               // adding migrated candidate
                  (translated to the problem
                  encoding of agent j)
13             S_j ← S_j ∪ {ENCODING_j(s_submitted)}\
                {s_tobereplaced} ;
14         end
15     end
16     cycles ← cycles + 1;
17 end
18 return arg min{FITNESS(BEST(S_i)) | i ∈ ℕ_n^+};
```

the agents (e.g. the information is transmitted from any given agent to its successor in the ring-based structure). The set of agents involved in this collaborative scheme can be regarded as a network of nodes, each containing a certain optimisation algorithm. These algorithms (i.e. the agents) operate in different parts of the same search space. The whole scheme constitutes an effective mechanism for escaping from local minima (by means of the information exchange among the agents). This approach has been proven to be efficient for a number of combinatorial problems [45, 46, 47, 48].

Now, unlike what it is often done in this kind of collaborative approach, we study the effect of considering different search spaces to be handled separately in the nodes of the network. More precisely, we consider a number of cooperative algorithms in which agents are loaded with one of the techniques previously proposed in Sect. 3 or their equivalent adapted to the dual representation (as shown in Sect. 4.1), and where the method loaded in any given agent might also have symmetry

breaking constraints (as explained in Sect. 4.2). This means that some agents possibly work on different encoding/search spaces and use distinct problem formulations. The algorithms depend on their interaction topology and the model used for encoding the candidates, and these are discussed below.

4.3.1. Formal definition

Let R be an architecture with n agents; each agent a_i ($1 \leq i \leq n$) in R consists of one of the metaheuristics described in preceding sections. Therefore, these agents can work on the primal or dual model, with or without symmetry breaking. The agents engage in periods of isolated exploration followed by synchronous communication. We denote by Θ the maximum number of exploration/communication cycles in a certain cooperative model. In addition, let S_i be the pool of solution candidates associated with agent a_i (i.e. if the agent is loaded with a local search (LS) method then $\#S_i = 1$, and if the agent is endowed with a population based method –e.g. an MA– then $\#S_i \geq 1$, where $\#S_i$ represents the cardinality of S_i), and let $\mathbf{T}_R \subseteq \mathbb{N}_n^+ \times \mathbb{N}_n^+$ be the communication topology over R (i.e. if $(i, j) \in \mathbf{T}_R$ then agent a_i can send information to agent a_j). The general architecture of the model is described in Algorithm 2. This algorithm's input consists of the problem parameters (i.e. v, b, r, k, and λ), the topology of the agent network (that defines the communication policy, as explained below), the n algorithms (i.e. agents or metaheuristics) running on each node of the cooperative system (i.e. the network), the candidate migration policy, and the criteria for accepting the candidates (see below for details about these two procedures). Each agent also has its own parameters (such as operator application rates, and type of encoding –primal/dual–). The algorithm's output is the best individual found during the search process (line 18; note that the FITNESS function is a well-known concept in evolutionary computing and basically returns a value that measures how close a candidate is to an optimal solution). First, all the agents are initialised with random solution(s) (lines 1-3). The initialisation of a pool S_i associated with agent i in the system is specific to the model (i.e. primal or dual) handled by the agent a_i. Next, the algorithm is executed for a maximum number Θ of iteration cycles (lines 5-15) where, in each cycle, the search technique contained within each agent is executed to update its associated pool of solutions (lines 6-8); note also that, if the agent contains an LS method, this basically means an improvement of its unique solution, but if the agent contains a population-based method, then a new pool of solutions is generated). Next, solutions are fed from one agent to another according to the topology considered (lines 9-14). This process means that, initially (line 10), the candidate to be transmitted from the pool of the source agent (i.e. node or metaheuristic i) is selected with respect to the *migration policy* given as input (see below). Next, agent a_j checks (line 11) whether the incoming solution from agent a_i (line 10) has to be accepted according to the *acceptance criteria* also provided as input (see below). Finally, if the submitted solution is accepted, it replaces an individual in the candidate pool of agent j (line 13); the candidate to replace is selected via previously defined heuristics (line 12). Note that many different criteria for candidate migration (from agent i to agent j) and candidate acceptance (in agent j) can be defined. Combining diverse policies generates different cooperative algorithms. We now describe a number of combinations that will be used in the experimental section.

4.3.2. Communication topologies

Three strategies for $\mathbf{T}_R$ (see line 9 in Algorithm 2) are considered here in this paper. These are based on the following interaction topologies:

- RING: $\mathbf{T}_R = \{(i, i(n) + 1) \mid i \in \mathbb{N}_n^+$ and $i(n)$ denotes i modulo $n\}$. Thus, there exists a circular list of agents in which each agent only sends (resp. receives) information to its successor (resp. from its predecessor).
- BROADCAST: $\mathbf{T}_R = \mathbb{N}_n^+ \times \mathbb{N}_n^+$, i.e. a *go with the winners*-like topology in which the best overall solution at each synchronisation point is transmitted to all agents. This means all agents execute the intensification over the same part of the search space at the beginning of each cycle.
- RANDOM: $\mathbf{T}_R$ is composed of n pairs (i, j) that are randomly sampled from $\mathbb{N}_n^+ \times \mathbb{N}_n^+$. This sampling is done each time communication takes place, and, hence, any two agents might eventually communicate at any step.

These communication topologies were already proposed in [47, 48] to handle a tool switching problem with some success; however, symmetry breaking, different encodings and different policies were not considered for migration and solution acceptance in that work. Now, we propose a wider cooperative scheme to handle symmetrical constrained optimisation problems. Moreover, we have also adapted some of the ideas proposed in [49] for memetic algorithms to our cooperative algorithms generated from the scheme in Algorithm 2. In particular, we consider a number of policies for the submission of candidates from agent i (i.e. the migration policy) as well as the acceptance of candidates submitted to agent j (i.e. the reception/acceptance policy). With respect to *candidate selection in the migration procedure* (i.e. SELECTCANDIDATEFROM(S_i) in line 10, of Algorithm 2), we propose three strategies:

- (RANDOM R): send a random solution of the pool from agent i,
- (DIVERSE D): send the candidate in S_i that maximises the diversity[1] in S_j, and
- (WORST W): send the worst candidate of the pool in agent i.

As for the reception and replacement policies (i.e. procedures ACCEPTSUBMITTEDCANDIDATE($s_{submitted}, S_j$) in line 11 and SELECTCANDIDATETOREPLACEIN(S_j) in line 12, respectively), three alternatives are also considered:

[1] To this end, individuals whose genotypic distance (in a Hamming sense) to individuals in the receiving population is maximal are selected.

- (RANDOM R): always accept the submitted candidate and replace one random individual in pool S_j,
- (DIVERSE D): accept a new individual if and only if, it improves the diversity of the pool in agent j and replace the worst, and
- (WORST W): always accept the candidate and replace the worst in pool S_j.

Also note that if a candidate solution taken from agent i is finally accepted in agent j, it first has to be translated – if necessary – to the encoding model used in agent j, as agents a_i and a_j may work on different search spaces (i.e. representation models); this is reflected in line 13 by the function called $\text{ENCODING}_j(s_{submitted})$.

5. Experiments

This section describes the experimental analysis conducted. Given the large number of algorithms considered (resulting from the combination of different metaheuristics, encodings, use of symmetry-breaking procedures, communication topology, etc.), we first describe the notation in detail, as well as the experimental setting in Sect. 5.1 and Sect. 5.2, respectively. Subsequently, we report the results obtained in Sect. 5.3 and Sect. 5.4, and analyse these in Sect. 5.5.

5.1. Notation

In this subsection we explain the notation used to describe the algorithmic models, providing some specific examples for the sake of clarity.

5.1.1. Non-cooperative algorithms

Each algorithm is identified by a sequence of identifiers separated by a dot. First, the basic metaheuristics (as described in Sect. 3) are hill climbing (Hc), tabu search (Ts), genetic algorithm (GA), and memetic algorithm (MA), all of which are based on the swap neighbourhood. Additionally, for the population-based methods (i.e. GA and MA), the recombination procedure is characterised by the particular operator used – here we focus on the use of the greedy crossover operator (Gd)– and by its arity, i.e. the number m of parents used (denoted as Am). Additionally, we use an asterisk ($*$) to indicate the use of symmetry-breaking methods, a B to indicate that individuals are encoded in a binary way (i.e. the primal model), and a D to indicate that these are encoded in the decimal representation (i.e. the dual formulation).

Examples of notation of non-cooperative algorithms: Hc.B (resp. Hc.B*) denotes a hill climbing method that was implemented for the primal (i.e. binary) model without (resp. with) symmetry breaking; Ts.D (resp. Ts.D*) denotes a tabu search implemented for the dual model without (resp. with) symmetry breaking; likewise, GA.B*.A2.Gd denotes a genetic algorithm with 2-parent greedy crossover (as explained in Sect. 3 – see [19] for details on this crossover operator) implemented for the primal encoding with symmetry breaking, GA.D.A4.Gd is a genetic algorithm with a 4-parent greedy crossover implemented for the dual formulation without symmetry breaking, MA.Hc.B.A2.Gd a memetic algorithm with a hill climbing method as local search and a 2-parent greedy recombination implemented for the primal formulation without symmetry breaking, and MA.Ts.D*.A2.Gd is a memetic algorithm with tabu search as local search and a 2-parent greedy crossover operator implemented for the dual model with symmetry breaking.

5.1.2. Cooperative methods

These algorithms are composed of some of the previous techniques combined according to given topology and migration policies. The notation $\mathbf{T}n(a_1,\ldots,a_n)$MR is used to characterise the method. Here:

- $\mathbf{T} \in \{$BROADCAST (Bc), RANDOM (Ra), RING (Ri)$\}$ denotes the topology of the model,
- n is the number of agents (i.e. algorithms) connected as described in Sect. 4.3,
- a_i is the optimisation method used by agent i (for $1 \leq i \leq n$), and
- M, R $\in \{$RANDOM (R), DIVERSE (D), WORST (W) $\}$ identify, respectively, the policies to migrate and accept candidates in the agents (see Sect. 4.3). In our experiments, we have considered the following six combinations for $migration-reception$ policies:
  1. RANDOM-RANDOM (RR), that is to say, RANDOM policy for both migration and reception.
  2. RANDOM-WORST (RW): RANDOM policy for migration and WORST strategy for reception.
  3. RANDOM-DIVERSE (RD): RANDOM policy for migration and DIVERSE strategy for reception.
  4. DIVERSE-RANDOM (DR): DIVERSE policy for migration and RANDOM strategy for reception.
  5. DIVERSE-WORST (DW): DIVERSE policy for migration and WORST strategy for reception, and
  6. DIVERSE-DIVERSE (DD): DIVERSE policy for migration and DIVERSE strategy for reception.

  Note that we do not include the combinations WD (i.e. WORST-DIVERSE), WR (i.e. WORST-RANDOM) and WW (i.e. WORST-WORST). The reason is that preliminary experiments showed that choosing the WORST policy for migration exhibited a poor performance compared to the other combinations.

Examples of notation of cooperative algorithms: Ri2(Ts.B, MA.Ts.D.A2.Gd)RW is a 2-agent $\{$RING topology$\}$-based cooperative algorithm that connects (a) a TS working on the binary representation, and (b) an MA that works on the dual representation, which uses a 2-parent greedy crossover, and that integrates TS as the underlying local search; in this case, the algorithm always sends a random candidate selected from the pool of the origin node (RANDOM policy for migration),

which will replace the worst individual in the destination node (WORST policy for the acceptance policy). Similarly, Ra3(Ts.B, MA.Ts.B.A2.Gd, MA.Ts.D.A4.Gd)RD denotes a 3-agent cooperative algorithm that connects, in a RANDOM topology, (a) tabu search and (b) two different MAs; the individuals to migrate are randomly chosen (i.e. a RANDOM policy for migration) whereas candidates are accepted only if they increase the diversity of the solution pool (i.e. DIVERSE acceptance criteria).

Note that in the cooperative algorithms the same optimisation method might be used by several agents (this is the case, for instance, in the algorithm Bc4(Ts.B,Ts.B,Ts.B,MA.Ts.D*.A4.Gd)RD in which 3 of the 4 agents contain the local search Ts.B.). The rationale for this is to try to increase the contribution of a certain method to the resulting cooperative hybrid, whose overall search profile is influenced by the particular mix of optimisation methods used. For clarity, in these cases, we use the notation $\mathbf{T}n(pa,qb)$MR to denote the n-agent cooperative algorithm

$$\mathbf{T}n(\underbrace{a,\ldots,a}_{p \text{ times}},\underbrace{b\ldots b}_{q \text{ times}})\text{MR}$$

in which agents a and b are employed p and q times respectively (and where p and q are arbitrary numbers that fulfill $n = p+q$); moreover, p (resp. q) is not written when $p = 1$ (resp. $q = 1$). So, for instance, Bc4(3Ts.B, MA.Ts.D*.A4.Gd)RD denotes the model Bc4(Ts.B,Ts.B,Ts.B,MA.Ts.D*.A4.Gd)RD (i.e. here $p = 3$ and $q = 1$). Also, Ra5(3Ts.B,2MA.Ts.B.A2.Gd)DW is a 5-agent algorithm where the local search Ts.B is embedded in 3 agents and the algorithm MA.Ts.B.A2.Gd is contained within the other two agents (i.e. here $p = 3$ and $q = 2$).

5.2. Experimental configuration

The experiments were conducted on the 86 instances taken from [16, 32] where $vb \leqslant 1000$ and $k \neq 3$. This corresponds to the hardest instances reported, since the cases where $k = 3$ were easily solvable. All algorithms have been run 30 times per problem instance and for a maximum number of evaluations equal to $E_{\max} = 2 \cdot 10^7$. All runs of local search techniques inside the memetic versions were limited to exploring $n_v = 2 \cdot 10^6$ neighbours. This number corresponds to the maximum number of backtrack steps (fixing one entry of the incidence matrix) performed by CLS in [16]. The GAs consider the equivalent number of full evaluations in each case. The number of evaluations without improvement to trigger intensification in a Local Search method or re-starting in a population-based method is $n_\iota = n_v/10$. Other parameters of the GA/MA are population size $popsize = 100$, crossover and mutation probabilities $p_X = .9$ and $p_M = 1/\ell$ (where $\ell = vb$ is the size of individuals) respectively, $f_{\%} = 10\%$ and binary tournament selection of parents to be recombined. We have also considered 2 and 4 parents for recombination (i.e. $m \in \{2,4\}$) and –particularly for the MA– p_{LS} was set to 0.005. In addition, the number of cycles Θ was set to 5 in the cooperative versions. These parameter values were chosen because some preliminary experiments indicated that they provided a good trade-off between the computational cost and the quality of solutions attained. Note however, that most of these values were the same as those used in [17, 19]. The versions of HC, TS and GA working on the primal model (i.e. in the binary search space) are those described in [17]. The MAs with Gd correspond to the versions described in [19].

Table 2: Number (and percentage) of the instances solved by the basic and integrative metaheuristics working alone on the set of 86 instances taken from [32, 16].

algorithm	# (%)	algorithm	# (%)
Hc.B	35 (40.70 %)	Ts.B	57 (66.28 %)
Hc.D	6 (6.98 %)	Ts.D	43 (50.00 %)
Hc.B*	25 (29.07 %)	Ts.B*	51 (59.30 %)
Hc.D*	3 (3.49 %)	Ts.D*	46 (53.49 %)
GA.B.A2.Gd	25 (29.07 %)	GA.B.A4.Gd	35 (40.70 %)
GA.D.A2.Gd	35 (40.70 %)	GA.D.A4.Gd	36 (41.86 %)
GA.B*.A2.Gd	38 (44.19 %)	GA.B*.A4.Gd	43 (50.00 %)
GA.D*.A2.Gd	28 (32.56 %)	GA.D*.A4.Gd	31 (36.05 %)
MA.Hc.B.A2.Gd	43 (50.00 %)	MA.Ts.B.A2.Gd	53 (61.63 %)
MA.Hc.B.A4.Gd	46 (53.49 %)	MA.Ts.B.A4.Gd	56 (65.12 %)
MA.Hc.D.B.A2.Gd	14 (16.28 %)	MA.Ts.D.B.A2.Gd	52 (60.47 %)
MA.Hc.D.B.A4.Gd	15 (17.44 %)	MA.Ts.D.B.A4.Gd	52 (60.47 %)
MA.Hc.B*.A2.Gd	43 (50.00 %)	MA.Ts.B*.A2.Gd	53 (61.63 %)
MA.Hc.B*.A4.Gd	46 (53.49 %)	**MA.Ts.B*.A4.Gd**	**59 (68.60 %)**
MA.Hc.D*.A2.Gd	10 (11.63 %)	MA.Ts.D*.A2.Gd	51 (59.30 %)
MA.Hc.D*.A4.Gd	9 (10.47 %)	MA.Ts.D*.A4.Gd	47 (54.65 %)

The dual versions of these algorithms all use the same parameters (population size, genetic operator rates, etc) as the corresponding primal versions. Versions with symmetry breaking follow the considerations described in Sect. 4.2.1 (for the primal model-based algorithms) and Sect. 4.2.2 (for the dual model-based algorithms). The combination of the two problem representation models and the possibility of breaking the symmetries gave rise to four different scenarios, namely, primal representation with and without symmetry breaking, and their equivalents in the dual model.

5.3. Basic and integrative approaches

Thirty two basic and integrative algorithms have been considered, i.e. 8 local search algorithms (resulting from the two LS methods considered in this paper –HC and TS– and the four aforementioned scenarios), 8 GAs (resulting from the four previous scenarios plus two different arities for greedy recombination), and 16 MAs (resulting from the integration of either HC or TS, for performing local improvement, in each of the previous GAs). The performance results obtained for all these metaheuristics are reported in Table 2, which shows the number of problem instances (out of 86) that were solved in at least one run by each of the algorithms, along with the corresponding success percentage.

In general, TS variants outperform both their HC counterparts and GA versions. More specifically, the TS algorithm working on the swap neighbourhood and the primal model (i.e. Ts.B) has proven to be very efficient (this confirms the results shown in [17]). We have also found that using symmetry breaking in the best memetic proposal (i.e. MA.Ts.B.A4.Gd) described in [19] produces an improvement:

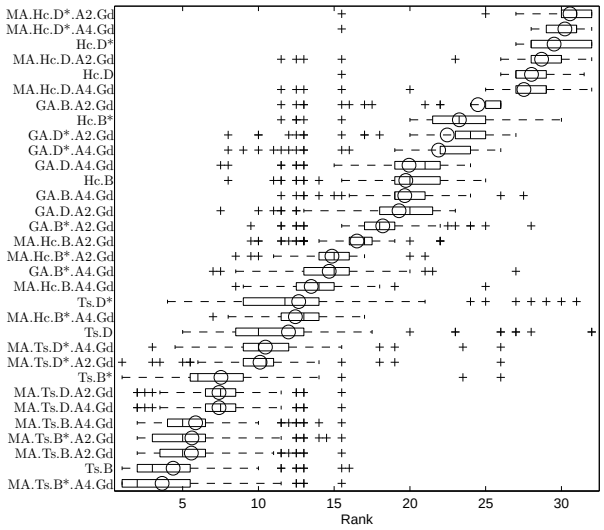

Figure 5: Rank distribution of basic and integrative metaheuristics on the 86 instances taken from [32, 16].

MA.Ts.B*.A4.Gd solves 59 instances while only 56 are solved by MA.Ts.B.A4.Gd.

5.3.1. A rank-based comparison

Due to the high number of algorithm variants, it is not easy to compare their performances by simply inspecting the numerical tables. Therefore, we have opted for a rank-based approach. More precisely, we have computed the rank r_j^i of each algorithm j on each instance i. For ranking purposes, we have used the number of solutions found for each instance (from the set of 30 runs) and employed the mean fitness to break ties. The best algorithm is ranked first and the worst is ranked 32nd. The distributions of these ranks are shown in Figure 5. At first glance, the integrative cooperative algorithms (i.e. the MA versions) perform better than non-cooperative counterparts. The results confirm that the MA with symmetry breaking (i.e. MA.Ts.B*.A4.Gd) outperforms the one without it and can now be considered the best metaheuristic for the BIBD problem. A more detailed statistical analysis indicates that there are significant differences ($\alpha = 0.05$) among the different algorithms according to the Friedman test [50] and the Iman-Davenport test [51]. For this reason, we carried out a post-hoc analysis using the Holm-Bonferroni test [52]. As shown in Table 3, MA.Ts.B*.A4.Gd is significantly better than the other algorithms, except Ts.B and the other three MAs using TS on the primal model (regardless of the use of symmetry breaking or recombination arity).

5.3.2. A factor-based comparison

Some interesting observations emerge when the data is factorised along particular dimensions. To begin with, let us consider the representation used. If we compare the algorithms operating on the primal representation with those operating on the dual representation, we observe a highly significant difference in favour of the primal representation (according to a Wilcoxon

Table 3: Results of the Holm-Bonferroni test on integrative approaches using MA.Ts.B*.A4.Gd as the control algorithm. Only the algorithms that show no significant statistical difference –at the standard level $\alpha = 0.05$– with respect to the control algorithm are shown (i.e. those for which p-value $\geq \alpha/i$).

i	algorithm	z-statistic	p-value	α/i
1	Ts.B	5.039e-001	3.071e-001	5.000e-002
2	MA.Ts.B.A2.Gd	1.345e+000	8.928e-002	2.500e-002
3	MA.Ts.B*.A2.Gd	1.374e+000	8.477e-002	1.667e-002
4	MA.Ts.B.A4.Gd	1.544e+000	6.125e-002	1.250e-002

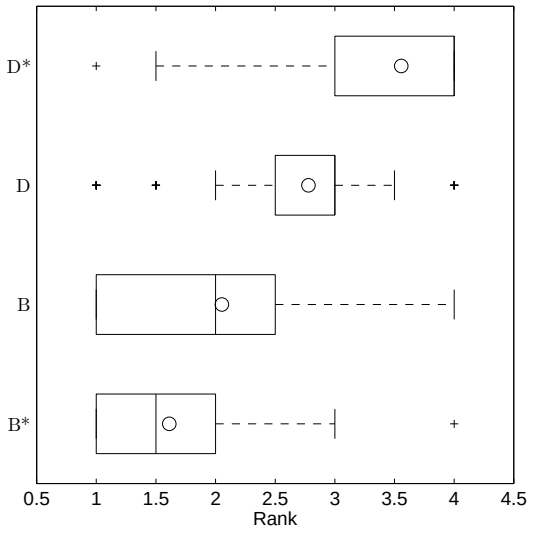

Figure 6: Rank distribution of the four groups: {primal, dual}×{with symmetry-breaking, without symmetry-breaking}. B identifies the Binary (i.e. Primal) model without symmetry breaking, B* the Binary Model with symmetry breaking, D the Decimal (i.e. Dual) model without symmetry breaking, and D* the Decimal model with symmetry breaking.

signed rank test [53], p-value ≈ 0). This result also holds for each group of techniques (LS, GA, MA) when analysed separately (p-values always below 0.0001). Now, if an analogous analysis with regard to the use of symmetry breaking (SB) is performed considering all algorithms, it turns out that not using SB is significantly better (again, p-value ≈ 0) as well as for LS and MAs.

A joint analysis of representation and use of symmetry breaking was also executed splitting the data into four groups: {primal, dual}×{with symmetry breaking, without symmetry breaking} (i.e. B, D, D and D*). As can be seen in Figure 6, using the primal model with symmetry breaking yields the best global rank, and significantly outperforms the other combinations. The Friedman test and the Iman-Davenport test confirm that there are significant differences (at the standard $\alpha = 0.05$) among the different groups. Results of the Holm-Bonferroni test, applied as a post-hoc analysis and using B* as control algorithm, indicates that the control algorithm shows significant statistical differences ($\alpha = 0.05$) with the other algorithms (i.e. in all the cases, p-value $< \alpha/i$).

Note, we have stated above that –globally speaking– the use of symmetry breaking (SB) in the basic metaheuristics work-

ing as standalone methods is not recommendable. However, we emphasise that we have utilised just one type of symmetry breaking, called variable reduction (VR) in [42]. Other types of symmetry breaking –such as, e.g. lexicographical ordering– may have more added value. In any case, we have detected that, when our SB proposal is used by some population-based methods, the performance improves. So, the best memetic algorithm (i.e. MA.Ts.B*.A4.Gd) as well as the two best GA versions (i.e. GA.B*.A2.Gd and GA.B*.A4.Gd) use symmetry breaking. Moreover, SB seems to work well when combined with the primal model but not with the dual model. This demonstrates that each representation and symmetry-breaking approach provides a different angle to the search process. This raises interesting prospects on their joint use in cooperative models, whose performance is analysed next.

5.4. Cooperative approaches

This section evaluates the performance of a number of different cooperative algorithms (instantiated from the scheme shown in Algorithm 2 described in Sect. 4.3). The idea is to harness the synergy between the metaheuristics when these work in cooperation. We have considered the three topologies proposed with a number n of agents between 2 and 5 (following [48]), and a number of cycles $\Theta = 5$ (we set this value based on preliminary experiments with values of $\Theta \in \{5, 10, 15\}$).

The cooperative models considered are variations of the template (see Sect. 5.1) $\mathbf{T}n(pa,qb)$MR, where $p+q=n$ and $a,b \in \mathscr{A}$ for a certain collection $\mathscr{A}$ that contains two types of agents. We have considered the following four collections:

- $\mathscr{A}_1 = \{$Ts.B,MA.Ts.B*.A4.Gd$\}$
- $\mathscr{A}_2 = \{$Ts.B,MA.Ts.B.A2.Gd$\}$
- $\mathscr{A}_3 = \{$Ts.B,MA.Ts.D.A4.Gd$\}$
- $\mathscr{A}_4 = \{$Ts.B,MA.Ts.D*.A2.Gd$\}$

The algorithms in these collections have been picked due to their good individual performances according to Table 3. Ts.B is the best basic technique whereas the other four algorithms in these collections represent the best integrative methods in the domains B*, B, D and D*. Moreover, each collection represents a form of combining algorithms: $\mathscr{A}_1$ represents the cooperation of a model (in this case, the Binary representation) with and without symmetry breaking (i.e. B-B*), $\mathscr{A}_2$ represents the cooperation of techniques working in the same computation domains with no symmetry breaking (i.e. B-B), $\mathscr{A}_3$ represents the cooperation of methods working in distinct computation domains with no symmetry breaking (i.e. B-D), and $\mathscr{A}_4$ the scheme in which methods working on distinct computation domains with distinct policies for symmetry breaking are cooperating (i.e. B-D*). Considering all possible combinations of topology, number of agents and migration/reception policies, a total of 288 algorithmic variants were created. The value 288 results from combining: (1) two different representations (i.e. primal, dual), (2) two distinct forms of managing the problem solving (i.e. with or without symmetry breaking), (3) three distinct communication topologies (i.e. RING, RAND or BROADCAST), (4) six options for migration/reception policies, and (5) 4 collections of algorithms to load the agents in the 4-agent scheme (i.e. collections $\mathscr{A}_1$,$\mathscr{A}_2$,$\mathscr{A}_3$ and $\mathscr{A}_4$).

Table 4: The 29 problem instances considered hard from the 86 instances in [32, 16]. The first column is the instance label assigned in [16], columns 2–6 present the instance parameters, and column 7 gives an indication of the size of the instance.

ID	v	b	r	k	λ	vb
21	14	26	13	7	6	364
27	15	30	14	7	6	450
28	16	30	15	8	7	480
33	16	32	12	6	4	512
34	15	35	14	6	5	525
39	17	34	16	8	7	578
43	18	34	17	9	8	612
44	25	25	9	9	3	625
46	21	30	10	7	3	630
48	16	40	15	6	5	640
50	15	45	21	7	9	675
54	19	38	18	9	8	722
56	22	33	12	8	4	726
57	14	52	26	7	12	780
58	27	27	13	13	6	729

ID	v	b	r	k	λ	vb
59	21	35	15	9	6	735
62	20	38	19	10	9	760
63	16	48	18	6	6	768
70	21	42	10	5	2	882
71	21	42	12	6	3	882
72	21	42	20	10	9	882
73	16	56	21	6	7	896
76	18	51	17	6	5	918
77	22	42	21	11	10	924
80	16	60	30	8	14	960
82	31	31	10	10	3	961
83	31	31	15	15	7	961
85	22	44	14	7	4	968
86	25	40	16	10	6	1000

Due to the large number of variants, we consider a reduced set of 29 instances to evaluate the performance of the algorithms. More specifically, we use the 29 instances that could not be solved by the (non-constructive) metaheuristics methods mentioned in Sect. 2.2[2]. These 29 hard problem instances are shown in Table 4.

5.4.1. Analysis of Design Factors

Given the large number of algorithms, it is useful to factorise the analysis along different dimensions corresponding to different design decisions regarding the number of agents involved, their topology, or the communication policy. Let us start by considering the six combinations (i.e. DD, DR, DW, RD, RR and RW) used. Comparing the results of these six policies on any single algorithmic variant. More precisely, for each of the 288/6=48 variants (resulting from different combinations of topology, number of agents and individual algorithms used), we ranked the six migration/reception (M/R) policies according to the number of optimal solutions found (using the mean fitness to break ties). The best M/R policy is RD (random selection of migrants, replacement for diversity). This is further confirmed by a Friedman and Iman-Davenport tests (at the standard level of $\alpha = 0.05$) which indicated that there are statistically significant differences among policies, and by a Holm-Bonferroni test that showed that RD is significantly better than the other policies.

Next, we consider an analysis along the topology axis. In this case, the three topologies are ranked on 288/3=96 algorithmic variants each. As indicated by the Friedman and Iman-

[2]Note that an MA with Gd proposed in [19] was able to solve 63 of the 86 instances but it required a substantially larger number of evaluations compared to the same algorithm that solved 57 instances as reported in [17].

Davenport tests ($\alpha = 0.05$) for all topologies, there is a statistically significant difference in this case. Indeed, the BROADCAST topology stands out from the others, as confirmed by the Holm-Bonferroni employing BROADCAST as control algorithm. The larger exchange of information among agents in this topology might be the cause.

We now turn our attention to the number of agents used in the cooperative model. We have considered $n \in [2,5]$ and these four values are ranked across 288/4=72 algorithmic variants each. Both the Friedman test and the Iman-Davenport test indicate that there are significant differences among the different values, so we conducted the Holm-Bonferroni test using $n = 2$ (the best ranked value) as the control algorithm. The result was that the difference is significant against the remaining values of n. The number of agents is therefore a factor that exerts a significant influence on the performance of the cooperative model. In our experiments, the algorithms that employ 2 agents perform better than those that employ more agents. In addition, the algorithms that employ 2 or 3 agents perform, in general, better than those based on 4-5 agents.

5.4.2. Analysis of Top Performing Models

We now focus on the most effective cooperative models. More specifically, we consider all cooperative models that were able to solve at least 9 (hard) problem instances (out of the 29 mentioned above) in at least one run. This set of algorithms is composed of the 41 variants shown in Table 5 together with the number of problem instances solved and the corresponding success percentage.

Note first that there is a predominance of the collection $\mathscr{A}_2$. The collaboration of techniques working in the same computation domain (in this case, Binary encoding) is present in 38 of the 41 algorithms. Good results are also provided by 3 other cooperative algorithms in which the collaboration of methods is based on the collection $\mathscr{A}_3$, that is to say, techniques working collaboratively on the primal and dual models. However, encouraging cooperation between methods that manage symmetry breaking and other techniques that do not consider symmetry issues does not seem to be helpful. Note however that this statement should not be generalised to any symmetry breaking. Specifically, for the case considered here, cooperation between agents working in B-B* or B-D* (i.e. those based on collections $\mathscr{A}_1$ or $\mathscr{A}_4$) is not advisable. Both the Friedman test and the Iman-Davenport test (with $\alpha = 0.05$) indicate that there are significant differences among the different collections of algorithms. Moreover, the Holm-Bonferroni test using $\mathscr{A}_2$ as control algorithm confirms that it is statistically significant from the other collections.

If we check the relative frequency of each particular design parameter among these 41 algorithmic models, we obtain the results shown in Table 6. The results are consistent with the statistical analysis from the previous section. Thus, we can see that the RD policy for migration/reception performs best. Regarding topology, BROADCAST is present in more than half of the models. Finally, a low number of agents (i.e. 2 or 3 agents) seems to offer a good performance more frequently.

Table 5: (Central column) Number (#) and percentage (%) of instances from Table 4 solved by those cooperative algorithms (identified in the first column) that were successful in at least 9 (hard) problem instances. Right column displays the collection of algorithms that collaborate in the cooperative search.

Algorithms	# (%)	Collection
Bc2(Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathscr{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)DR	9 (31.03 %)	$\mathscr{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)DW	12 (41.38 %)	$\mathscr{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)RD	11 (37.93 %)	$\mathscr{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)RR	12 (41.38 %)	$\mathscr{A}_2$
Bc2(Ts.B,MA.Ts.B.A2.Gd)RW	9 (31.03 %)	$\mathscr{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)DD	12 (41.38 %)	$\mathscr{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathscr{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)DW	11 (37.93 %)	$\mathscr{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)RD	10 (34.48 %)	$\mathscr{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)RR	12 (41.38 %)	$\mathscr{A}_2$
Ra2(Ts.B,MA.Ts.B.A2.Gd)RW	12 (41.38 %)	$\mathscr{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathscr{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)DR	**13 (44.83 %)**	$\mathscr{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)RD	10 (34.48 %)	$\mathscr{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)RR	10 (34.48 %)	$\mathscr{A}_2$
Ri2(Ts.B,MA.Ts.B.A2.Gd)RW	11 (37.93 %)	$\mathscr{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathscr{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)DR	9 (31.03 %)	$\mathscr{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)DW	9 (31.03 %)	$\mathscr{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)RD	11 (37.93 %)	$\mathscr{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathscr{A}_2$
Bc3(2Ts.B,MA.Ts.B.A2.Gd)RW	9 (31.03 %)	$\mathscr{A}_2$
Ra3(2Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathscr{A}_2$
Ra3(2Ts.B,MA.Ts.B.A2.Gd)RD	**13 (44.83 %)**	$\mathscr{A}_2$
Ra3(2Ts.B,MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathscr{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathscr{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathscr{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)RD	10 (34.48 %)	$\mathscr{A}_2$
Ri3(2Ts.B,MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathscr{A}_2$
Bc4(2Ts.B,2MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathscr{A}_2$
Bc4(2Ts.B,2MA.Ts.B.A2.Gd)RR	9 (31.03 %)	$\mathscr{A}_2$
Ra4(2Ts.B,2MA.Ts.B.A2.Gd)DW	9 (31.03 %)	$\mathscr{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)DD	9 (31.03 %)	$\mathscr{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)DR	10 (34.48 %)	$\mathscr{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)RD	9 (31.03 %)	$\mathscr{A}_2$
Bc5(3Ts.B,2MA.Ts.B.A2.Gd)RW	9 (31.03 %)	$\mathscr{A}_2$
Ra5(3Ts.B,2MA.Ts.B.A2.Gd)DW	9 (31.03 %)	$\mathscr{A}_2$
Bc2(Ts.B,MA.Ts.D.A4.Gd)RD	9 (31.03 %)	$\mathscr{A}_3$
Ra5(3Ts.B,2MA.Ts.D.A4.Gd)RD	9 (31.03 %)	$\mathscr{A}_3$
Ri5(3Ts.B,2MA.Ts.D.A4.Gd)RD	9 (31.03 %)	$\mathscr{A}_3$

Table 6: Relative frequency of each particular design parameter among the selected cooperative algorithms in Table 5. Left column indicates the combination MR of migration(M)/reception(R) policies where M, R $\in$ {RANDOM (R), DIVERSE (D), WORST (W) as explained in Sect. 5.1. Central column shows the communication topology where Bc = BROADCAST, Ra = RANDOM, and Ri = RING. Right column refers to the number of agents in the algorithm.

M/R Policy			Topology			Number of agents		
DD	7	(17.07 %)	Bc	19	(46.34 %)	$n = 2$	18	(43.90 %)
DR	7	(17.07 %)	Ra	12	(29.27 %)	$n = 3$	13	(31.71 %)
DW	5	(12.19 %)	Ri	10	(24.39 %)	$n = 4$	3	(7.32 %)
RD	10	(24.39 %)				$n = 5$	7	(17.07 %)
RR	7	(17.07 %)						
RW	5	(12.19 %)						

Figure 7 shows the rank distribution for the 41 models. The Friedman and Iman-Davenport tests indicate significant differences in their ranks ($\alpha = 0.05$). Subsequently, the Holm-Bonferroni test, using as the control algorithm Bc3(2Ts.B,MA.Ts.B.A2.Gd)RD, shows that there is no statistical difference between the first four cooperative algorithms, but there is a statistically significant difference with the others (see

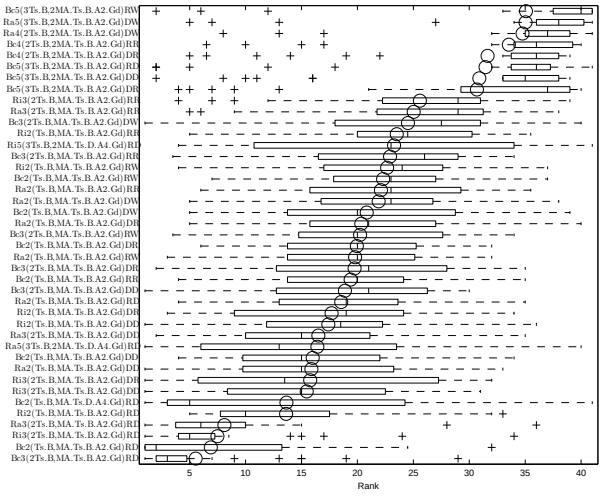

Figure 7: Rank distribution of selected cooperative algorithms from Table 5.

results in Table 7).

Table 7: Results of the Holm-Bonferroni test, for cooperative algorithms using Bc3(2Ts.B,MA.Ts.B.A2.Gd)RD – i.e. the algorithm with the best mean rank according to Figure 7– as the control algorithm. Only the algorithms that show no significant difference $-\alpha = 0.05-$ to the control algorithm are shown (i.e. those for which p-value $\geq \alpha/i$).

i	algorithm	z-statistic	p-value	α/i
1	Bc2(Ts.B,MA.Ts.B.A2.Gd)RD	4.330e-001	3.325e-001	5.000e-002
2	Ri3(2Ts.B,MA.Ts.B.A2.Gd)RD	6.193e-001	2.679e-001	2.500e-002
3	Ra3(2Ts.B,MA.Ts.B.A2.Gd)RD	8.166e-001	2.071e-001	1.667e-002

5.5. Discussion: Cooperation vs. Integration

The high-level distinction between integration and cooperation is well known in the literature and has been used to taxonomise the hybridisation of search algorithms (see e.g. [43]). This section discusses and compares the performance of our cooperative and integrative approaches. Note that we have first compared a collection of 32 integrative metaheuristics on the 86 classical instances of the problem. An interesting result is that the best metaheuristic approach described previously in the literature, namely a memetic algorithm, was improved by adding symmetry breaking. This new algorithm solved 59 out of 86 instances (i.e. 68.60% effectiveness). Despite this improvement, there were 27 instances that could not be solved. These instances are part of a subset of 29 instances considered difficult to solve. To tackle these harder instances, we proposed a number of cooperative algorithms. Two of these could solve 13 of the hard instances, and, overall, 14 different instances were solved by some of the cooperative methods, namely, instances 21, 27, 28, 33, 34, 39, 44, 48, 50, 57, 58, 64, 73, and 76 in Table 4 (the precise set of problems solved by each method is shown in [54]). In summary, this means that we could solve 71 out of 86 instances (i.e. 82.56% effectiveness) by either method (i.e. 57 instances from the original set of 86 plus 14 hard ones).

Table 8: Running time comparison between MA=$Ma.Ts.B*.A4.Gd$ and Coop=$Ra3(2Ts.B,Ma.Ts.B.A2.Gd)RD$. The two problem instances $I_1 = \langle 14,26,13,7,6\rangle$ and $I_2 = \langle 25,25,9,9,3\rangle$ that were solved by both methods are selected as benchmarks for comparison. T_{max}/T_{min} show the maximum/minimum time (in seconds) consumed by the best execution of each technique to find a solution. Last column displays the improvement of the cooperative method with respect to the MA.

	Instances					
	I_1		I_2		Coop/MA(%)	
Time(s)	MA	Coop	MA	Coop	I_1	I_2
T_{min}	23.00	0.02	20.08	0.03	1150	669
T_{max}	5754.78	24.67	9317.03	34.91	233	266

It is important to underline that a large number of cooperative algorithms were successful for many problem instances where simpler metaheuristics did not succeed. Note that this result, which shows the advantages of the cooperative techniques with respect to their underlying components (i.e. the algorithms that compose the collaboration) working alone, is consistent with other studies conducted for problems other than the BIBD problem. For instance, the results are similar to those obtained in [47] in the context of the tool switching problem, and our findings are consistent with the conclusions of this paper. Needless to say, not all integrative methods (or cooperative methods) perform in the same way. Indeed, the crux of the whole matter with regards to effectiveness is often (1) the balance between exploration and exploitation and (2) the search overhead in which composite methods incur. These two issues are influenced by different factors and in this sense, one of the contributions of the work presented here, with respect to the conclusions regarding integration and cooperation of algorithms, is to calibrate them in this particular domain. Another point of novelty is to shine a spotlight on two more dimensions (in addition to the methods used in integrative hybrid and the cooperative schemes defined with them), namely symmetry breaking and problem primal/dual formulations, which to the best of our knowledge has not been explored before in this context.

Additionally, we highlight the importance of the policies for migrating and accepting solutions from the agents in the metaheuristic network for the performance of the cooperative algorithms. In our experiments, using the WORST policy for migration of candidates deteriorated the performance of the algorithm. In general, the combination RD has a positive effect on cooperation as the first six algorithms from the top-ten cooperative algorithms are based on it. Also note that we have executed a large number of experiments, and considered other combinations and algorithms, many of which are not reported here to avoid clutter, given that their performance was poor.

Another very interesting result that can be extracted from our experiments is that cooperation performs better when all the connected algorithms are working in the same computation domain and using the same problem formulation.

With respect to running times, we have compared the best integrative approach (i.e. $Ma.Ts.B*.A4.Gd$, which could only solve 2 of the 29 hard problem instances shown in Table 4), with one of our best cooperative methods (i.e.

$Ra3(2Ts.B, Ma.Ts.B.A2.Gd)RD$, which solved 13 of the 29 hard instances). To be fair, we only considered the two instances that were solved by both methods. The results are shown in Table 8. The memetic algorithm is noticeably much slower than the cooperative one. This must be due to some kind of synergy between the agents.

6. Conclusions and future work

This paper has dealt with the generation BIBDs, a difficult combinatorial problem, using metaheuristic techniques. In previous work, we tackled this problem by means of local search algorithms, genetic algorithms and hybrid techniques, always using a binary genotypical space and a classical (primal) formulation of the problem. In this paper, we have proposed a number of alternative approaches to deal with this problem. First, we have defined a (novel) dual problem formulation with a natural representation in the integer domain. All heuristics proposed in previous work have been adapted to this novel formulation. In addition, based on the highly symmetrical nature of the problem, we have considered basic symmetry-breaking procedures to reduce the search space of the problem in both encodings, i.e. primal (binary) and dual (integer). The advantage of using these procedures is highly dependent on other design decisions but the best approach has been shown to be a memetic algorithm using symmetry breaking on the primal representation, outperforming the previous best known metaheuristics for this problem.

Despite these good results, integrative metaheuristics were somewhat limited for harder problem instances in the test suite. We have therefore also proposed a scheme to instantiate cooperative models that combine the integrative algorithms. This scheme is based on a spatial topology and policies for exchanging solutions, and allows a large number of different instantiations. We have considered three different topologies (to define the flow of information among algorithms), a varying number of connecting agents in these topologies, three different migration procedures (for selecting solutions to be sent), and three different acceptance criteria (for handling incoming solutions). All these factors affect the performance of the algorithms. For a better understanding of our cooperative algorithms, we have also conducted an analysis of the influence of some of these factors. The conclusions extracted can help design better cooperative algorithms. In addition, some of our cooperative proposals can be considered at present, as state-of-the-art metaheuristic methods for handling BIBDs.

Moreover, the conclusions extracted from our use of primal/dual encodings, symmetry breaking and their combination in a cooperative model may guide the design of other cooperative algorithms to handle other combinatorial problems with symmetries. In this sense, the work presented here may also be viewed as a methodology to address combinatorial problems with symmetries in a general way, that is to say, firstly designing basic metaheuristics, secondly applying hybrid methods constructed from them (i.e. memetic algorithms), then considering alternative formulation/encodings of the problem by adjusting the basic and integrative methods to these, and finally connecting all the previous techniques using cooperative schemes.

The work described here has focused on solving BIBDs. However, we believe that a large part of our work can be generalised to solve any symmetrical combinatorial problem. For this reason, our cooperative scheme is general and so does not depend on specific algorithms, but rather on the design factors that have been analysed here. In this sense, the metaheuristics (especially the cooperative versions) described should not be applied directly to handle open problem instances in design theory. To cope with open instances, our proposals should be tailored to the problem at hand. This basically means that we should craft our metaheuristics to suit the particular characteristics of the open instance with the aim of exploiting the available structure as much as possible. This requires not only making decisions about design factors (e.g. nature of agents, topology of the cooperative system, and migration/reception policies), but also incorporating specific knowledge about the problem instance.

In future research, we plan to deepen the study of the intensification/diversification balance of the algorithm, aiming to improve its performance for the hardest problem instances. An intermediate goal would be to endow the MAs with self-adaptive capabilities [55] to enhance their search capabilities. This topic can also be explored in cooperative models with many defining parameters, which could be adjusted while running in response to the state of the search. In addition, asymmetric representations have been shown to be effective in solving other combinatorial problems (see [41, 42]) and might be used in the formulation of the dual model as well.

References

[1] C. Colbourn, J. Dinitz (Eds.), CRC Handbook of Combinatorial Designs, Discrete Mathematics and Its Applications, CRC Press, 2010.

[2] K. Hinkelmann, O. Kempthorne, Partially Balanced Incomplete Block Designs, John Wiley & Sons, Inc., pp. 119–157.

[3] B. Ariel, D. P. Farrington, Randomized block designs, in: G. Bruinsma, D. Weisburd (Eds.), Encyclopedia of Criminology and Criminal Justice, Springer New York, 2014, pp. 4273–4283.

[4] M. Buratti, Pairwise balanced designs from finite fields, Discrete Mathematics 208–209 (1999) 103–117.

[5] C.-S. Cheng, Regular graph designs, in: Encyclopedia of Statistical Sciences, John Wiley & Sons, Ltd, 11 edition, 2014.

[6] K.-T. Fang, D. Lin, Uniform design in computer and physical experiments, in: H. Tsubaki, S. Yamada, K. Nishina (Eds.), The Grammar of Technology Development, Springer Japan, 2008, pp. 105–125.

[7] J. van Lint, R. Wilson, A Course in Combinatorics, Cambridge University Press, 2001.

[8] R. Mead, The Design of Experiments: Statistical Principles for Practical Applications, Cambridge University Press, 1990.

[9] M. Buratti, Some (17q, 17, 2) and (25q, 25, 3)BIBD constructions, Designs, Codes and Cryptography 16 (1999) 117–120.

[10] L. Lan, Y. Y. Tai, S. Lin, B. Memari, B. Honary, 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 (2008) 39–48.

[11] R. d. C. dos Santos Navarro, V. P. R. Minim, A. N. da Silva, A. A. Simiqueli, S. M. Della Lucia, L. A. Minim, et al., Balanced incomplete block design: an alternative for data collection in the optimized descriptive profile, Food Research International 64 (2014) 289–297.

[12] M. Basu, S. Bagchi, D. K. Ghosh, Design of an efficient load balancing algorithm using the symmetric balanced incomplete block design, Information Sciences 278 (2014) 221–230.

[13] A. L. Madsen, F. Jensen, A. Salmerón, M. Karlsen, H. Langseth, T. D. Nielsen, A new method for vertical parallelisation of tan learning based on balanced incomplete block designs, in: L. C. van der Gaag, A. J. Feelders (Eds.), 7th European Workshop on Probabilistic Graphical Models, Springer International Publishing, Cham, 2014, pp. 302–317.

[14] D. G. Corneil, R. Mathon, Algorithmic techniques for the generation and analysis of strongly regular graphs and other combinatorial configurations, Annals of Discrete Mathematics 2 (1978) 1–32.

[15] P. Gibbons, P. Östergrd, Computational methods in design theory, in: C. Colbourn, J. Dinitz (Eds.), The CRC handbook of combinatorial designs, Boca Raton: CRC Press, 1996, pp. 730–740.

[16] S. Prestwich, A local search algorithm for balanced incomplete block designs, in: F. Rossi (Ed.), 9th International Conference on Principles and Practices of Constraint Programming (CP2003), volume 2833 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 53–64.

[17] D. Rodríguez, C. Cotta, A. Fernández-Leiva, Finding balanced incomplete block designs with metaheuristics, in: 9th European Conference Evolutionary Computation in Combinatorial Optimization – EvoCOP 2009, volume 5482 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2009, pp. 156–167.

[18] M. R. Faghihi, S. Tat, On ϕ p-optimality of incomplete block designs: An algorithm, Communications in Statistics - Simulation and Computation 45 (2016) 758–769.

[19] D. Rodríguez, C. Cotta, A. Fernández-Leiva, A memetic algorithm for designing balanced incomplete blocks, International Journal of Combinatorial Optimization Problems and Informatics (IJCOPI) 2 (2011) 14–22.

[20] B. Benhamou, Study of symmetry in constraint satisfaction problems, in: 2nd Workshop on Principles and Practice of Constraint Programming, PPCP 94, DTIC Document, pp. 246–254.

[21] R. Backofen, S. Will, Excluding symmetries in constraint-based search, Constraints 7 (2002) 333–349.

[22] T. Fahle, S. Schamberger, M. Sellmann, Symmetry breaking, in: W. T. (Ed.), 7th International Conference on the Principles and Practice of Constraint Programming - CP 2001, volume 2239 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001, pp. 93–107.

[23] I. P. Gent, B. Smith, Symmetry breaking in constraint programming, in: W. Horn (Ed.), 14th European Conference on Artificial Intelligence – ECAI 2000, IOS Press, 1999, pp. 599–603.

[24] P. Meseguer, C. Torras, Exploiting symmetries within constraint satisfaction search, Artif. Intell. 129 (2001) 133–163.

[25] W. G. Cochran, G. M. Cox, Experimental Design, John Wiley, New York, 1957.

[26] R. A. Fisher, F. Yates, Statistical Tables for Biological, Agricultural and Medical Research, Oliver & Boy, 3 edition, 1949.

[27] D. Whitaker, C. M. Triggs, J. A. John, Construction of block designs using mathematical programming, Journal of the Royal Statistical Society. Series B (Methodological) 52 (1990) 497–503.

[28] G. Zergaw, A sequential method of constructing optimal block designs, Australian & New Zealand Journal of Statistics 31 (1989) 333–342.

[29] T. Tjur, An algorithm for optimization of block designs, Journal of Statistical Planning and Inference 36 (1993) 277–282.

[30] P. Flener, A. M. Frisch, B. Hnich, Z. Kzltan, I. Miguel, T. Walsh, Matrix modelling, in: B. Smith, K. Brown, P. Prosser, I. P. Gent (Eds.), CP-01 Workshop on Modelling and Problem Formulation. International Conference on the Principles and Practice of Constraint Programming, Paphos, Cyprus, pp. 1–7.

[31] J.-F. Puget, Symmetry breaking revisited, in: P. V. Hentenryck (Ed.), 8th International Conference on the Principles and Practice of Constraint Programming (CP 2002), volume 2470 of *Lecture Notes in Computer Science*, Springer, Ithaca, NY, USA, 2002, pp. 446–461.

[32] P. Bofill, R. Guimerà, C. Torras, Comparison of simulated annealing and mean field annealing as applied to the generation of block designs, Neural Networks 16 (2003) 1421–1428.

[33] S. Prestwich, Negative effects of modeling techniques on search performance, Annals of Operations Research 18 (2003) 137–150.

[34] D. Yokoya, T. Yamada, A mathematical programming approach to the construction of bibds, International Journal of Computer Mathematics 88 (2011) 1067–1082.

[35] B. N. Mandal, Linear integer programming approach to construction of balanced incomplete block designs, Communications in Statistics - Simulation and Computation 44 (2015) 1405–1411.

[36] D. Rodríguez, E. Darghan, J. Monroy, A multi-agent proposal in the resolution of instances of BIBD, Revista Colombiana de Estadística 39 (2016) 267–280.

[37] F. Neri, C. Cotta, P. Moscato, Handbook of Memetic Algorithms, volume 379 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, 2012.

[38] F. Rothlauf, Representations for genetic and evolutionary algorithms, Springer, 2 edition, 2006.

[39] M. Campêlo, V. A. Campos, R. C. Corrêa, On the asymmetric representatives formulation for the vertex coloring problem, Discrete Applied Mathematics 156 (2008) 1097 – 1111. GRACO 2005.

[40] R. Jans, J. Desrosiers, Binary clustering problems: Symmetric, asymmetric and decomposition formulations, GERAD Technical Report G-2010-44 (2010) 1 – 15.

[41] R. Jans, J. Desrosiers, Efficient symmetry breaking formulations for the job grouping problem, Computers & Operations Research 40 (2013) 1132 – 1142.

[42] N. Vo-Thanh, R. Jans, E. D. Schoen, P. Goos, Symmetry breaking in mixed integer linear programming formulations for blocking two-level orthogonal experimental designs, Computers & Operations Research 97 (2018) 96 – 110.

[43] J. Puchinger, G. R. Raidl, Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification, in: J. Mira, J. Álvarez (Eds.), Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, volume 3562 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2005, pp. 113–124.

[44] T. G. Crainic, M. Toulouse, Explicit and emergent cooperation schemes for search algorithms, in: V. Maniezzo, R. Battiti, J.-P. Watson (Eds.), Learning and Intelligent Optimization – LION 2007 II, Springer, Berlin Heidelberg, 2008, pp. 95–109.

[45] C. Cruz, D. Pelta, Soft computing and cooperative strategies for optimization, Applied Soft Computing 9 (2009) 30–38.

[46] A. Masegosa, F. Mascia, D. Pelta, M. Brunato, Cooperative strategies and reactive search: A hybrid model proposal, in: T. Stützle (Ed.), Learning and Intelligent Optimization, volume 5851 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2009, pp. 206–220.

[47] J. Amaya, C. Cotta, A. Fernández-Leiva, A memetic cooperative optimization schema and its application to the tool switching problem, in: R. Schaefer, et al. (Eds.), Parallel Problem Solving from Nature - PPSN XI, volume 6238 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2011, pp. 445–454.

[48] J. Amaya, C. Cotta, A. Fernández-Leiva, Memetic cooperative models for the tool switching problem, Memetic Computing 3 (2011) 199–216.

[49] R. Nogueras, C. Cotta, An analysis of migration strategies in island-based multimemetic algorithms, in: T. Bartz-Beielstein, J. Branke, B. Filipić, J. Smith (Eds.), Parallel Problem Solving from Nature – PPSN XIII, volume 8672 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2014, pp. 731–740.

[50] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (1937) 675–701.

[51] R. Iman, J. Davenport, Approximations of the critical region of the Friedman statistic, Communications in Statistics 9 (1980) 571–595.

[52] S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[53] E. Lehmann, H. D'Abrera, Nonparametrics: statistical methods based on ranks, Prentice-Hall, Englewood Cliffs, NJ, 1998.

[54] D. Rodríguez, C. Cotta, A. Fernández-Leiva, Data for the search of 2-designs: solutions and metaheuristics code, Data in Brief (2017). Submitted.

[55] J. Smith, Self-adaptative and coevolving memetic algorithms, in: F. Neri, C. Cotta, P. Moscato (Eds.), Handbook of Memetic Algorithms, volume 379 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, 2012, pp. 167–188.