Computers & Operations Research 55 (2015) 52–64

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A grouping genetic algorithm with controlled gene transmission for the bin packing problem

Marcela Quiroz-Castellanos [a,*], Laura Cruz-Reyes [a], Jose Torres-Jimenez [b], Claudia Gómez S. [a], Héctor J. Fraire Huacuja [a], Adriana C.F. Alvim [c]

[a] Tecnológico Nacional de México, Instituto Tecnológico de Ciudad Madero, División de Estudios de Posgrado e Investigación/Depto. de Sistemas y Computación, Juventino Rosas y Jesús Urueta, C.P. 89440, Cd. Madero, Tamaulipas, Mexico
[b] CINVESTAV-Tamaulipas, Information Technology Laboratory, Km. 5.5 Carretera Victoria-Soto La Marina, 87130 Victoria Tamps., Mexico
[c] Universidade Federal do Estado do Rio de Janeiro, Av. Pasteur 458 – CCET – Urca 22290-240, Rio de Janeiro, Brazil

ARTICLE INFO

Available online 28 October 2014

Keywords:
One-dimensional bin packing
Genetic algorithms
Grouping problems
Heuristics
Gene transmission

ABSTRACT

In this study, the one-dimensional Bin Packing Problem (BPP) is approached. The BPP is a classical optimization problem that is known for its applicability and complexity. We propose a method that is referred to as the *Grouping Genetic Algorithm with Controlled Gene Transmission* (GGA-CGT) for Bin Packing. The proposed algorithm promotes the transmission of the best genes in the chromosomes without losing the balance between the selective pressure and population diversity. The transmission of the best genes is accomplished by means of a new set of grouping genetic operators, while the evolution is balanced with a new reproduction technique that controls the exploration of the search space and prevents premature convergence of the algorithm. The results obtained from an extensive computational study confirm that (1) promoting the transmission of the best genes improves the performance of each grouping genetic operator; (2) adding intelligence to the packing and rearrangement heuristics enhances the performance of a GGA; (3) controlling selective pressure and population diversity tends to lead to higher effectiveness; and (4) GGA-CGT is comparable to the best state-of-the-art algorithms, outperforming the published results for the class of instances Hard28, which appears to have the greatest degree of difficulty for BPP algorithms.

1. Introduction

The off-line one-dimensional Bin Packing Problem (BPP) is defined as follows. Given an unlimited number of bins with a fixed capacity $c > 0$ and a set of n items, each with a specific weight $0 < w_i \leq c$, BPP comprises packing all of the items into the minimum number of bins without violating the capacity of any bin. The size m of the solution is defined by a partition of the set of items into m subsets, each of which corresponds to the content of one bin. BPP is a well-known grouping optimization problem that is considered to be intractable because it belongs to the NP-hard class; as a result, it demands a large amount of resources for its solutions [1]. The importance of BPP comes from its extensive number of industrial and logistic applications; furthermore, BPP frequently occurs as a sub-problem in several practical problems in different areas [2–4].

In the present work, a *Grouping Genetic Algorithm with Controlled Gene Transmission* (GGA-CGT) for Bin Packing is presented. GGA-CGT incorporates efficient heuristics to drive the transmission of the best genes of the chromosomes, to favor the generation and evolution of high-quality solutions. We propose (a) a new efficient packing heuristic that improves the packing process and promotes the generation of a high-quality initial population; (b) the use of intelligent grouping operators, which promote the transmission of the best genes and the exploitation of the search space; (c) a new simple rearrangement procedure that improves solutions and promotes the exploration of the search space; and (d) the use of a controlled reproduction technique to ensure the balance between selective pressure and population diversity. The proposed algorithm and its main components are experimentally contrasted with related studies from the literature to confirm that the gene transmission strategy is a promising approach. Experimental results show that GGA-CGT obtains high-quality solutions in a short amount of time for all of the standard instances, outperforming the effectiveness of the best algorithms

* Corresponding author. Tel.: +52 833 3 57 48 20x124; fax: +52 833 2 15 85 44.
E-mail addresses: qc.marcela@gmail.com (M. Quiroz-Castellanos), lauracruzreyes@itcm.edu.mx (L. Cruz-Reyes), jtj@cinvestav.mx (J. Torres-Jimenez), cggs71@hotmail.com (C. Gómez S.), automatas2002@yahoo.com.mx (H.J.F. Huacuja), adriana@uniriotec.br (A.C.F. Alvim).

http://dx.doi.org/10.1016/j.cor.2014.10.010
0305-0548/

that are state-of-the-art in the most difficult instances, the Hard28 class [5,6].

In Section 2, we describe the most prominent algorithms that were designed for the BPP solution. In Section 3, we introduce our new grouping genetic algorithm, which is referred to as GGA-CGT. Next, in Section 4, we present the components of GGA-CGT. In Section 5, we propose a new reproduction technique for controlling the evolutionary process. To show the effectiveness of each of the proposed strategies, preliminary computational experiments are scattered throughout the paper. The main experiments are shown in Section 6; this section presents the study of the overall performance of GGA-CGT and the impact of some of its features. Last, in Section 7, we present the conclusions and discuss further research directions.

2. Related studies

Over the past twenty years, throughout the search for the best possible solutions for BPP, elaborate procedures that incorporate various techniques have been designed. The performance of the algorithms has been evaluated with different types of published problems that were considered to be very hard. The most relevant results have been obtained by means of hybrid and heuristic algorithms [6–8].

Martello and Toth [9] proposed a branch-and-bound procedure (MTP) and a dominance criterion [10] to reduce the search space. The criterion establishes that if it is possible to extract a set of small items from a bin and substitute them for a single free item of the same weight, then it might be easier to improve the solution. This fact is true because it is easier to pack the set of small items into the remaining bins than to find a place for the larger item. Falkenauer [7] proposed a hybrid grouping genetic algorithm (HGGA) that uses a special encoding scheme for groups of items and a local optimization heuristic inspired by the dominance criterion of Martello and Toth. Scholl et al. [11] developed a hybrid technique (BISON) that combines a tabu search-based heuristic with a branch-and-bound method, using a "dual" strategy that comprises minimizing the fullness of the bins given a fixed number of bins. The proposed procedure includes a new branching scheme, which is based on some lower bounds of the optimal solution. Schwerin and Wäscher [12] proposed a new lower bound for BPP that is based on the cutting stock problem, and they integrated it into the MTP procedure of Martello and Toth, obtaining high-quality results with their new procedure (MTPCS).

Fleszar and Hindi [13] developed a new high-quality algorithm (Perturbation-MBS') that incorporates a modified version of the MBS heuristic from Gupta and Ho [14], a procedure that is based on a variable neighborhood search and lower bounds, to successively build a new solution by perturbing the current solution. Levine and Ducatelle [15] proposed a hybrid procedure that implements the ant colony optimization metaheuristic (HACO-BP), which includes a strategy for a local search that relies on the dominance criterion from Martello and Toth [10]. Bhatia and Basu [16] introduced a multi-chromosomal grouping genetic algorithm (MGGA) with grouping genetic operators and a rearrangement procedure based on the better-fit heuristic. Alvim et al. [8] presented a hybrid improvement heuristic (HI_BP) in which the dual strategy used by Scholl et al. [11] is reintroduced, in addition to the search space reduction techniques of Martello and Toth [10] and the use of lower bounding strategies. This procedure uses a load redistribution method that is based on dominance, differencing, and unbalancing and the inclusion of an improvement process that utilizes a tabu search, obtaining the best-reported results until that moment.

Singh and Gupta [17] proposed a combined approach (C_BP) that uses two heuristics: a hybrid steady-state grouping genetic algorithm and an improved version of the Perturbation-MBS' heuristic from Fleszar and Hindi [13]. Although the obtained performance is comparable to the HI_BP strategy from Alvim et al. [8], the procedure C_BP presents a less robust behavior in terms of the number of times that it finds the optimal solutions. Stawowy [18] proposed the use of a non-specialized evolutionary strategy (ES) that includes the following: a modified permutation with a separators encoding scheme, a separators removal technique for problem size reduction and intelligent mutations. The results that were obtained by this procedure are comparable to other results from more complex algorithms, such as HGGA and HI_BP [7,8]. Rohlfshagen and Bullinaria [19] developed a new genetic algorithm that was inspired by aspects of molecular genetics (ESGA). The authors compared their results with other state-of-the-art strategies, such as HI_BP and BISON [8,11], and obtained promising results.

Loh et al. [20] developed a weight annealing procedure (WA), making use of the concept of weight annealing to expand and speed up the neighborhood search by creating distortions in different parts of the search space. The proposed algorithm is simple and easy to implement, and the authors reported a high-quality performance, outperforming the solutions obtained by HI_BP [8]. Gómez-Meneses and Randall [21] presented a new evolutionary approach that uses the principle of eliminating the weakest component of a population and replacing it by a random component: hybrid extremal optimization (HEO). This procedure incorporates a local search that is inspired on the basis of the proposed Falkenauer's technique to improve the quality of the packing [7]. Lewis [22] proposed an intuitive hill-climbing method (HC) that uses a simple improvement scheme based on the dominance criterion to increase the fullness of the bins; the approach obtains good solutions, outperforming some algorithms [7,15] without surpassing the best state-of-the-art algorithms.

The most recent work in this area, to our knowledge, was presented by Fleszar and Charalambous [6]. The authors proposed a modification to the Perturbation-MBS' method [13] that uses a new sufficient average weight (SAW) principle to control the average weight of the items that are packed in each bin (Perturbation-SAWMBS). This heuristic outperforms the best algorithms from the state-of-the-art HI_BP, C_BP and WA [8,17,20]. In their paper, Fleszar and Charalambous [6] presented corrections to the results that were reported by Loh et.al [20] for the WA heuristic.

The study of the literature that is related to the BPP solution revealed that, until now, the best algorithms are the following: HGGA [7], HI_BP [8] and Perturbation-SAWMBS [6]. In Section 6, we compared the results that were obtained by our algorithm with the results that were obtained by these approaches; this comparison showed the impact of our strategies, which promote the best characteristics of the BPP solutions.

3. GGA-CGT genetic algorithm

A genetic algorithm (GA) is an evolutionary strategy that emulates the natural evolutionary process, seeking good-quality solutions to optimization problems. A GA modifies and re-combines pieces of existing solutions, which are encoded into chromosomes, by means of techniques that are inspired by biological evolution, such as the generation of an initial population, the selection of individuals with desirable characteristics, crossover, and mutation. Typically, the search for better solutions is guided by the results of evaluating a fitness function for each individual in the population, which is some ad-hoc function or the objective function. Based on this evaluation, individuals who have a higher fitness are given more opportunity to breed.

```
procedure GGA-CGT
1   Generate an initial population P with FF-ñ;
2   while generation < max_gen and Size(best_solution) > L2 do
3      Select n_c individuals to cross by means of Controlled_Selection;
4      Apply Gene_Level_Crossover+FFD to the n_c selected individuals;
5      Apply Controlled_Replacement to introduce progeny;
6      Select n_m individuals and clone elite solutions by means of Controlled_Selection;
7      Apply Adaptive_Mutation+RP to the best n_m individuals;
8      Apply Controlled_Replacement to introduce clones;
9      Updated the global_best_solution;
10  end;
end GGA-CGT.
```

Fig. 1. The proposed grouping genetic algorithm GGA-CGT for the bin packing problem.

Fig. 1 presents GGA-CGT, the algorithm that is proposed in this study (see Section 4 for a description of each genetic operator). This process begins generating an initial population P of individuals with our FF-$\tilde{n}$ packing heuristic (Line 1). Then, for a maximum of *max_gen* generations, selected individuals are recombined and mutated in two phases. In the first phase (Lines 3–5), n_c individuals are selected with Controlled_Selection, and the Gene_Level_Crossover+FFD operator is applied to them; then, Controlled_Replacement is used to introduce the progeny. In the second phase (Lines 6–8), according to a predefined *life_span*, elite individuals are cloned, and the best n_m individuals are selected to be the subject of genetic alterations by means of the Adaptive_Mutation+RP operator; next, clones are introduced to the population by means of Controlled_Replacement. At the end of each *generation*, the global best solution is updated (Line 9). The final result of the algorithm is obtaining the most capable individual over all of the evolutionary process. The algorithm iterates the maximum number of generations, *max_gen*, and it stops when it finds a solution for which the size matches the L_2 lower bound of Martello and Toth [9].

4. GA components

In this section, the intelligent grouping heuristics that are used for the generation and evolution of the BPP solutions (GA components) are described. As a principal contribution, we propose the following: a new intelligent packing heuristic that simplifies and improves the items' packing, a new rearrangement procedure that enables the exploitation and exploration of the search space and a new set of GA operators that preserve the best genes in the chromosomes.

4.1. Genetic encoding

The encoding of an optimization problem solution into a chromosome is a key issue in obtaining good GA performance. Generally, three encoding schemes have been proposed for bin packing problems: bin-based representation, object-based representation and group-based representation [23]. In the bin-based representation, a chromosome has a fixed length that is equal to the number of items, and each gene represents an item and indicates the bin where the item is packed. In the object-based representation, a chromosome represents a permutation of items, and a decoder is applied to retrieve the corresponding solution. In the group-based representation, the chromosomes could vary in length depending on how many bins are used in every solution; each gene in the chromosome represents a group of items, encoding the groups on a one-gene-for-one-bin basis. Falkenauer and Delchambre [24] noted that the first two encodings (bin-based and object-based) could run into difficulties because they are highly redundant and are not adapted to the BPP cost function; the authors proved that the group-based encoding scheme avoids these problems. Given the above reasons, we have chosen to represent our solutions with the group-based encoding scheme. In this way, our grouping genetic operators will work with the bins and the information about which items belong to which bins.

4.2. Fitness function

The fitness evaluation is made by means of the cost function that was introduced by Falkenauer and Delchambre [24]. The cost function evaluates the average of the squaring of the 'bin efficiency', which measures the exploitation of a bin's capacity. Given this function, it is better to have some nearly full bins and some nearly empty bins rather than several equally filled bins. Thus, in the evolutionary process, nearly empty bins can more easily be eliminated, and it is more likely that the small items that are packed in them could be moved to existing non-full bins (which would yield an improved solution). Additionally, the nearly empty bins will more easily accommodate additional items, which could otherwise be too large to fit into either of the half-filled bins. The objective of the GGA-CGT algorithm is to maximize the fitness values of the individuals in the population. Eq. 1 defines the cost function for the BPP, where m is the number of bins that are used in the solution, S_i is the sum of the item weights in bin i, and c is the capacity of the bins.

$$F_{BPP} = \frac{\sum_{i=1}^{m} (S_i/c)^2}{m} \tag{1}$$

4.3. Initial population

The initial population in most BPP evolutionary approaches is generally generated in a random manner by running a BPP heuristic on random permutations of the items [7,18,19]. Although we also generate random solutions, we first contribute to simplifying and improving the items' packing using a heuristic that is based on a dominance pattern that, given the set N of n items, determines the subset $\tilde{N}$ of $\tilde{n}$ items that have weights that are larger than fifty percent of the bin capacity. The $\tilde{n}$ value represents the total number of large items that could not be combined with others of the same subset. We propose the FF-$\tilde{n}$ heuristic, which takes advantage of this knowledge to reduce the number of steps in the packing process and to obtain solutions that have well-filled bins:

4.3.1. First fit with ñ pre-allocated-items (FF-ñ)

First, the $\tilde{n}$ large items of $\tilde{N}$ are packed in separate bins; then, the items of $N \backslash \tilde{N}$ are placed using the standard FF packing heuristic on a random permutation of this subset. Each of the items of $N \backslash \tilde{N}$ is placed in the first bin that has sufficient available capacity to accommodate the item; if none of the bins can store it, the item is placed in a new empty bin.

Experimental results showed a significant improvement in the solutions' quality and packing efficiency by generating the initial population using our FF-$\tilde{n}$ heuristic rather than the FF heuristic. For example, Fig. 2 presents the average number of bins that were

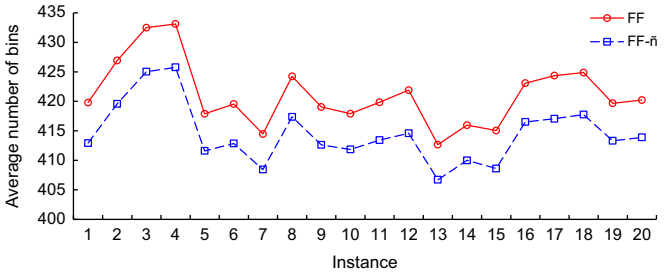

Fig. 2. Experimental results obtained by FF and FF-$\tilde{n}$ packing heuristics applied to the u1000 class (number of bins).

obtained by generating 100 random solutions with FF and FF-$\tilde{n}$ packing heuristics for the 20 instances of the uniform class u1000, where the average proportion of large items is $\tilde{n}/n = 0.3$. From this figure, it can be observed that by packing the $\tilde{n}$ large items first, the average quality of the random solutions is improved between 5 and 8 bins for each instance. For the 1615 standard instances, when FF-$\tilde{n}$ was applied ($\tilde{n} > 0$), we observed an average improvement of 3.67 bins for the solutions in the initial population using our heuristic rather than FF; for large instances ($n \geq 500$) when $\tilde{n}/n > 0.5$, the average improvement was 10.94 bins. These results hence prove that adding intelligence to the packing heuristics improves the solutions' quality and packing efficiency. In Section 6.3, we analyze the performance of our GGA-CGT over instances with different proportions of large items, which shows the impact of generating the initial population with FF-$\tilde{n}$ instead of FF.

Generally, the $N \backslash \tilde{N}$ items can be placed using any BPP packing heuristic. In this case, FF was chosen after comparing it with BF and WF because FF is the simplest among them, and we did not observe significant differences in the average quality of the population between their experimental results. Johnson [25] provides more information about the standard heuristics FF, BF and WF.

4.4. Grouping crossover operator

Generally, a crossover operator takes two parent solutions (chromosomes) and recombines them to produce a child solution. Given the two chromosomes, crossing them implies determining which genes must be inherited from parents to children. Those genes that survive the evolutionary process are characterized as being dominant over the others. In many types of problems, patterns of genes that are "close together" are meaningful, and preserving a section of the genes (a segment of the chromosome) from a good solution tends to lead to another good solution. However, when we use a grouping encoding scheme, the patterns of genes that are "close together" are much less meaningful, and preserving those patterns is not an appropriate goal for a recombination operator [26].

A high-quality crossover operator must be able to propagate to the offspring those patterns of features that contribute to increasing the fitness of the parents. In the case of a BPP solution, it is easy to realize that the better each bin is used, the fewer bins are needed to pack the items. Thus, in an optimal solution, it is expected that most of the bins are almost filled; therefore, the well-filled bins, which we refer to as the fullest-bin pattern, are the features of the chromosome that contribute to the fitness of the individuals.

4.4.1. Gene-level crossover

Accounting for the fullest-bin pattern, we propose a new BPP crossover that enables the parent chromosomes to contribute at the gene level rather than at the segment level, giving a higher probability of being preserved to the best genes (the fullest bins). Fig. 3 includes an example of this operator. Given two parent solutions p_1 and p_2, our crossover operator is used twice, to generate two children: p_1 with p_2 and p_2 with p_1. This operator considers the bins in descending order of their filling, to increase the likelihood that the best bins will be inherited by the children. Beginning at the best bin, the individual bins of both parents are compared in parallel,

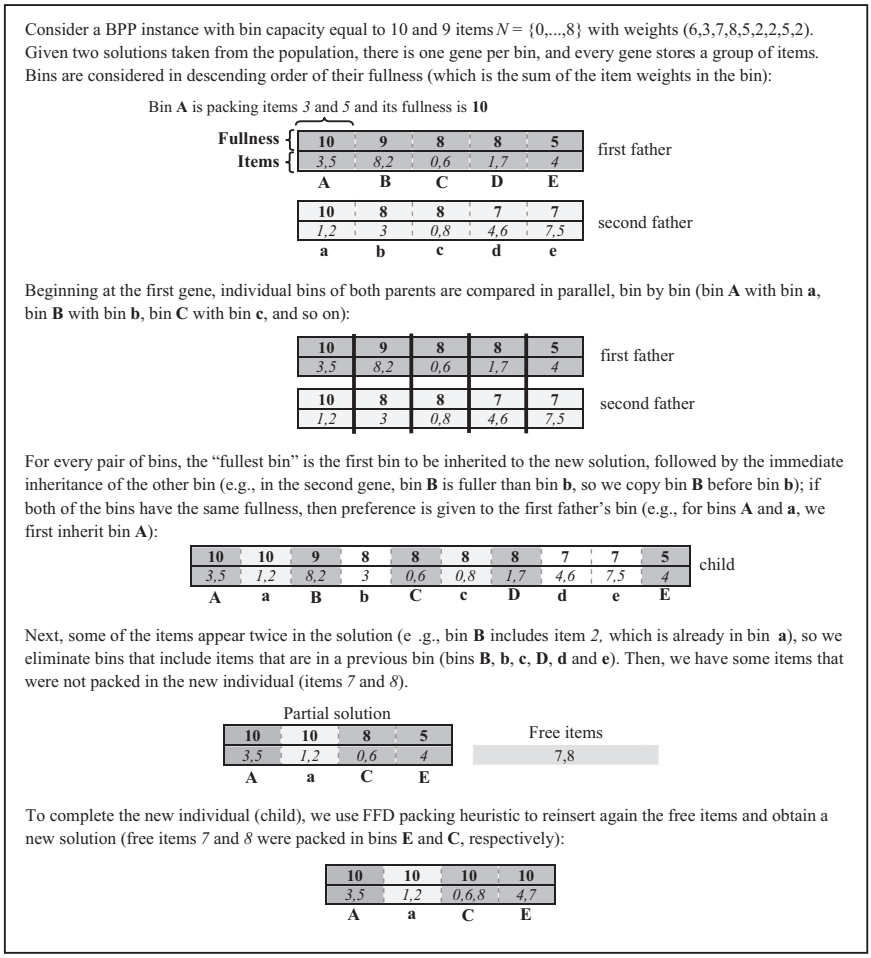

Fig. 3. An example of the gene-level crossover operator that preserves the fullest-bin pattern.

bin by bin. For each pair of bins, the fullest-bin is the first bin to be inherited by the new solution, followed by immediately inheriting the other bin. If both of the bins have the same fullness, then preference is given to the first father's bin. If one solution includes more bins than the other, then after all of the pairs of bins are compared and inherited, the remaining bins of the longer solution are inherited directly from this solution. Bins with duplicated items are eliminated from the new solution, and free items are reinserted with the FFD packing heuristic.

To show the contribution of our new crossover operator, we conducted a comparative experiment with three crossover alternatives: (1) segment-level crossover [24]; (2) the proposed gene-level crossover without considering the bins in descending order of their filling; and (3) the proposed crossover with ordered bins. All of the algorithms were run under the same conditions. We used a population of 100 individuals who was generated with FF-$\tilde{n}$. In each experiment, for a maximum of 100 generations, the 100 individuals (taken at random) were replaced by their progeny, which were produced by a crossover operator; FFD was used to reinsert the free items. We experiment with nine sets of benchmark instances that are recognized by the scientific community [27–30]. Table 1 shows the results that were obtained in the experiment.

For every class of instances, Table 1 first shows the number of test cases (Inst.), followed by the number of optimal solutions found by each crossover alternative (opt.). Last, the table shows the total *relative difference* (dif.), which is defined as $(y-x)/x$, where x is the optimal or best-known solution value, and y is the number of bins obtained by an algorithm. The last row of the table shows the *effectiveness*, which is defined as o/t, where o is the number of optimal solutions obtained by an algorithm, and t is the total number of instances used in the experiment. From the table, it can be observed that we observed an effectiveness improvement of 11% using a gene-level crossover rather than the segment-level crossover and an improvement of 19% by giving a higher probability to the best bins to be inherited, which is implicit in the ordering. These results prove that our strategy for preserving the fullest-bin pattern (that promotes the transmission of the best genes) tends to lead to a higher performance.

4.5. Grouping mutation operator

The grouping mutation operator has the function of introducing random changes (at a low rate) into the population by producing a small modification in some of the chromosomes. In traditional GAs, a mutation is applied by flipping each gene of the chromosome with a small probability. The mutation operator for BPP, which is encoded by the group-based scheme, has been defined differently [24]: given a chromosome, a few bins are eliminated from the solution; the items that composed those eliminated bins are thus missing from the solution, and they must be reinserted with some rearrangement heuristic. Usually, the selection of the bins to eliminate is made at random [7,19]; however, it has been noted that it is important to eliminate at least three bins and always eliminate the emptiest bin, to increase the probability of improving the solution quality [17,24]; we combine these two approaches.

Mutation operators that were proposed in the literature are highly disruptive and lose important genetic information by giving the same elimination probability to good and bad genes; in this way, some of the fullest bins are eliminated from the solution, losing patterns that are used to contribute to the solution's quality [7,31]. Next, let us consider the case in which all of the chromosomes, regardless of their size, have the same mutation probability; for example, when $p_m=0.1$, for a solution of 20 bins, we could expect to eliminate approximately 2 bins, which might be too few to obtain an improvement in the quality of the solution. However, for a solution of 1000 bins, we could expect to eliminate approximately 100 bins, which loses a large amount of genetic information and represents a substantial job for the rearrangement heuristic.

4.5.1. Adaptive mutation operator

We propose a new mutation operator, which works at the gene level (bin), promoting the transmission of the best genes in the chromosome. To introduce new genetic material without losing the best individual patterns (the well-filled bins), this new BPP mutation operator considers the bins in descending order of their filling, eliminating the least full bins of the solution and reinserting their items with a rearrangement heuristic (see Section 4.6). In contrast to the approach followed in the literature, we propose an adaptive function to calculate the number of bins to be eliminated from the solution. For each individual to mutate, the number of bins to eliminate n_b is a random variable that is calculated in relation to the size of the solution and the number of incomplete bins (that are not filled completely). Given a BPP solution with m bins, of which ι bins are incomplete ($\iota > 1$), the number of bins to eliminate is given by Eq. (2), where ε is the elimination proportion defined by Eq. 3, p_ε is the elimination probability defined by Eq. (4), and k is a parameter that defines the rate of change of ε and p_ε with respect to ι ($k > 0$).

$$n_b = \lceil \iota \cdot \varepsilon \cdot p_\varepsilon \rceil \tag{2}$$

$$\varepsilon = \frac{2-(\iota/m)}{\iota^{1/k}} \tag{3}$$

$$p_\varepsilon = 1 - Uniform\left(0, \frac{1}{\iota^{1/k}}\right) \tag{4}$$

Let us concentrate on Eqs. (3) and (4); the elimination proportion ε is inversely proportional to the number of incomplete bins (ι) and the percentage of incomplete bins (ι/m). In this way, the percentage of bins to eliminate (n_b/m) is higher when the solutions are smaller and decreases with larger solutions. Alternatively, the elimination probability p_ε is directly proportional to the number of incomplete bins, allowing, in most cases, more variations in ε when the solutions are smaller. Fig. 4 includes an example of this operator.

Note that our mutation operator introduces new genetic material without losing the fullest-bin pattern, but this arrangement does not mean that our operator is a local search mechanism. By not allowing a

Table 1
Experimental results for the segment-level crossover proposed by Falkenauer, and our proposed operators: gene-level crossover without and with ordered bins. The last operator gives a higher preserving probability to fullest-bins.

Class	Inst.	Segment-level crossover		Gene-level crossover			
		opt.	dif.	Without ordered bins		With ordered bins	
				opt.	dif.	opt.	dif.
Uniform	80	2	1.33	16	0.78	63	0.09
Triplets	80	0	7.90	0	6.64	0	2.00
Data Set 1	720	568	3.11	618	1.40	695	0.24
Data Set 2	480	284	7.63	322	5.83	439	0.65
Data Set 3	10	0	0.48	0	0.48	7	0.05
Was 1	100	8	5.11	48	2.88	89	0.61
Was 2	100	47	2.42	78	1.01	95	0.23
Gau 1	17	3	0.83	5	0.70	12	0.20
Hard28	28	5	0.33	5	0.33	5	0.33
Total	1615	917	29.18	1092	20.08	1405	4.44
Effectiveness		0.56		0.67		0.86	

reduction in the fullness of the best bins, we ensure the preservation of the fullest-bin pattern; however, we allow swaps between the items of the best bins and the free items to promote the creation of new genetic material. On the other hand, after the first phase of our rearrangement heuristic, the free items are reinserted by using the FF packing heuristic; thus, the fitness of the mutated solution could be worse than its original fitness.

4.6. Rearrangement heuristics for grouping operators

In the solution process, the genetic operators generate and modify individuals. During this process, some of the bins are eliminated, and free items exist that must be reinserted into the solution (see Figs. 3 and 4); to accomplish this reinsertion, some authors have used simple packing heuristics such as FFD or BFD [24,31], and others have used local optimizations [7,32]. It has been demonstrated that the success of a grouping genetic algorithm (GGA) is heavily dependent on the replacement heuristic and that adding intelligence to these heuristics enhances the performance of a GGA [26]. The example in Fig. 4 provides evidence that the use of a rearrangement heuristic that is capable of making swaps between packed and free items would have a positive impact on the effectiveness of the proposed mutation operator. We propose a new simple rearrangement procedure that works with pairs of free items.

4.6.1. Rearrangement by Pairs (RP)

This heuristic is composed of two stages: first, each bin is checked in an attempt to improve its package by making swaps between pairs of packed and free items; second, the final free items are introduced to the solution using the FF packing heuristic. We summarize in Fig. 5 the procedure RP, and we denote by S'=Swap(S, F, (p, s), i) the solution derived from S by swapping items p and s of the current bin B_j with the free item i of set F. Feasible(S') returns TRUE if the swap does not exceed the capacity of B_j and FALSE otherwise. In the first stage (Lines 1–10), given a random permutation of the bins and a random permutation of the free items, each bin is considered until a replacement is possible. For every pair of items in the bins, the free items are also considered in pairs. For every two pairs of items, there are two alternatives for replacement: (a) if it is possible, then replace the pair of items in the bin with one of the two free items of equal or higher weight that does not exceed the bin capacity (Lines 7–8); (b) if the pair of free items can fill the bin as good as or better than the pair of items in the bin without overflowing the bin, then swap the pair of items in the bin with the pair of free items (Line 9). In the second stage (Line 11), the FF heuristic is applied to reinsert the free items and complete the solution. A random permutation of the free items is useful when the weight of the largest free item is less than fifty percent of the bin capacity; in the alternative case,

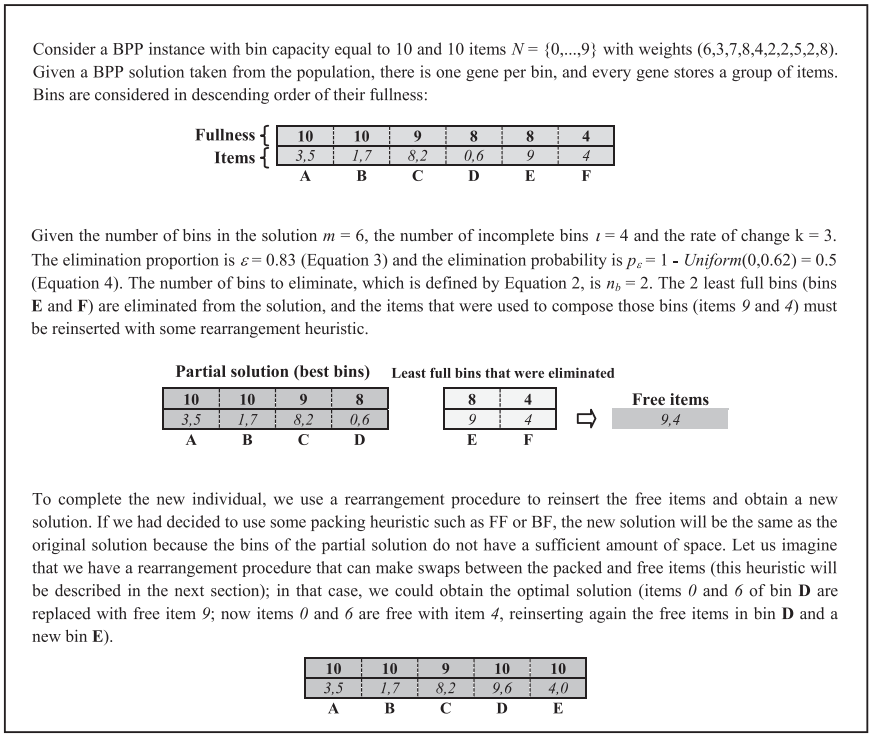

Fig. 4. An example of the adaptive mutation operator that preserves the fullest-bin pattern.

procedure RP(S, F)
1 Sort the bins of S in random order obtaining a sequence of b bins $S = (B_1, \ldots, B_b)$;
2 Sort the free items of F in random order;
3 $j \leftarrow 1$;
4 **while** $j \leq b$
5 **forall** $(p, s) \in B_j$ **do**
6 **forall** $(i, k) \in F$ **do**
7 **if** $w_i \geq w_p + w_s$ and Feasible($S' \leftarrow$ Swap(S, F, (p, s), i)) **then do** $S \leftarrow S'$; $j \leftarrow$ j + 1; **go to** step 4; **end**;
8 **if** $w_k \geq w_p + w_s$ and Feasible($S' \leftarrow$ Swap(S, F, (p, s), k)) **then do** $S \leftarrow S'$; $j \leftarrow$ j + 1; **go to** step 4; **end**;
9 **if** $w_i + w_k \geq w_p + w_s$ and Feasible(S' $\leftarrow$ Swap(S, F, (p, s),(i, k))) **then do** $S \leftarrow S'$; $j \leftarrow$ j + 1; **go to** step 4; **end**;
10 **end;**
11 Apply FF to reinsert the items in F to the solution S;
end RP.

Fig. 5. Rearrangement by Pairs (RP) procedure.

it is better to take the free items in descending order, to induce the fullest-bin pattern.

Note that our new rearrangement procedure has important consequences in the exploitation and exploration of the search space; it improves the quality of the genetic material, finding fuller bins, when a swap that increases the fullness of a bin is performed. It also allows for exploring new groups of items, when different or smaller item weights are made available for the packing heuristic; for this reason, we make swaps even when the items do not change the fullness of the bins.

To show the contribution of the adaptive mutation operator and the RP rearrangement heuristic, we conducted three experiments: (1) using the Falkenauer mutation with p_m=0.1 [24]; (2) using adaptive mutation; and (3) using adaptive mutation with the RP heuristic. All of the experiments were conducted under the same conditions. We used a population of 100 individuals who were generated with FF-$\tilde{n}$. In each experiment, for a maximum of 100 generations, these 100 individuals were mutated; in the first two experiments, FFD was used to reinsert the free items.

Table 2 shows the results that were obtained in each experiment. It can be observed that we observed an effectiveness improvement of 16% using our mutation rather than the Falkenauer mutation. These results confirm that our strategy by which to preserve the fullest-bin pattern, which promotes the transmission of the best genes, tends to be more effective and that the use of an elimination proportion as a random variable makes a major contribution to the mutations' success. Last, we can see that our rearrangement by pair (RP) heuristic improves the performance of our mutation operator by 22%, which demonstrates that adding intelligence to the rearrangement heuristic enhances the performance of a GGA.

5. Reproduction technique

The performance of a GA is affected not only by the genetic operators but also by the reproduction technique that is used to implement them. The reproduction technique defines how individuals must be chosen for reproduction and what individuals survive from one generation to the next. Not all of the algorithms produce an entirely new population at each generation. An elitist strategy involves copying the best individuals without changing them when they move from the current population to the new population. In a steady-state GA, n new individuals are introduced to the population each generation, and some of these new individuals could be selected to breed before moving on to the next generation. A well-planned reproduction technique will significantly impact the GA performance. We propose a new selection scheme and a new replacement strategy; they integrate our new reproduction technique, which we referred to as *Controlled Reproduction*.

Table 2
Experimental results obtained by the mutation proposed by Falkenauer and our mutation strategies.

Class	Inst.	Falkenauer mutation		Adaptive mutation		Adaptive mutation+RP	
		opt.	dif.	opt.	dif.	opt.	dif.
Uniform	80	3	1.06	32	0.33	78	0.01
Triplets	80	0	7.83	0	5.94	42	0.43
Data Set 1	720	537	3.10	623	1.00	694	0.25
Data Set 2	480	276	7.77	347	4.28	477	0.09
Data Set 3	10	0	0.42	0	0.21	8	0.03
Was 1	100	15	4.72	55	2.50	99	0.05
Was 2	100	61	1.79	87	0.60	99	0.04
Gau 1	17	2	0.92	5	0.70	12	0.20
Hard28	28	5	0.33	5	0.33	4	0.35
Total	1615	899	27.97	1154	15.93	1513	1.51
Effectiveness		0.55		0.71		0.93	

The tradeoff between exploration and exploitation in a GA is determined mainly by the mechanisms that are used to select individuals. Generally, a GA follows (a) a selection scheme to select individuals to which the genetic operators are applied; and (b) a replacement strategy by which to select the individuals in the population that are going to be replaced by the new individuals who are produced by genetic operators. Selective pressure gives individuals with a higher fitness a higher probability of being selected for reproduction, mutation, and survival; without it, the search process becomes random, and promising regions of the search space are not favored for exploitation. On the other hand, the population diversity allows for the exploration of new regions of the search space, helping to produce better solutions and avoid premature convergence [33,34]. We propose a simple reproduction technique, referred to as *Controlled Reproduction*, in which selective pressure and population diversity are balanced to increase the algorithm's performance.

5.1. Selection scheme

A selection scheme decides which individuals are allowed to pass on their genes to the next generation, either through cloning, crossover or mutation; it attempts to improve the average quality of the population by giving individuals of higher quality a higher probability of being part of the next generation. Generally, selection schemes from the literature could be split into three classes: proportional selection, tournament selection and ranking selection; the selection is according to the relative fitness, using the best of the random groups or the best individuals. In proportional selection, maintaining adequate selective pressure as the population becomes more homogeneous is difficult because there is less variation in the fitness. Alternatively, when tournament and ranking selection are configured to use a high selective pressure, a number of important solutions are discarded [35,36].

We propose a controlled selection scheme to make a clear differentiation between good and poor quality individuals, giving all of the individuals a chance to contribute to the next generation but forcing the survival of the best individuals. The strategy uses an elitist technique together with two-inverted rankings to select individuals, which ensures the diversity of the solutions. To maintain an adequate selective pressure, the elite technique controls the survival of an elite group B, which includes the best individuals of the population P, modifying elite individuals only when they exceed a predefined life span. In Section 6.3, we perform substantial experimentation on the impact of the size of the elite group B.

- *Controlled selection for crossover*. When applying the crossover operator, n_c parents are selected to generate n_c children. We generate the sets of parents G and R, with good and random individuals, respectively; these elements are going to be pairwise combined, crossing G_i with R_i ($0 \le i < n_c/2$). Set G includes $n_c/2$ individuals taken at random with a uniform probability from the best n_c individuals of P; set R includes $n_c/2$ individuals chosen at random from $P \backslash B$ with a uniform probability. Note that, as the best n_c individuals might have common members with the $P \backslash B$ set, the algorithm guards against the selection of the same individual at the same position, which eliminates the possibility of a crossover within the same individual. In contrast to other selection methods, the best individuals are forced to breed every generation, which promotes the survival of the best genes. However, we forbid the crossover between the best individuals (the elite group B) to promote genetic diversification.
- *Controlled selection for mutation*. Given a number of individuals to mutate n_m (with $n_m > |B|$), the mutation operator is applied

to the best n_m individuals of P in two phases: cloning of the elite group and mutation of the best. Following an elitist approach, the cloning is for individuals of set B whose age is less than a predefined *life_span* parameter; this criterion is intended to exploit the best solutions without losing good patterns. After the cloning phase, individuals to mutate are taken from P given the decreasing order of their fitness.

5.2. Replacement strategy

A replacement strategy defines which individuals in the current population are forced to perish to make room for new offspring. Generally, the replacement strategies can be split into three classes: age-based, fitness-based and random-based (deleting the oldest, worst or random individuals). It has been demonstrated that replacing the oldest and replacing random strategies decreases the selective pressure, losing some of the best solutions; while replacing the worst can lead to premature convergence [33,34].

Our controlled replacement strategy contributes to seeking an appropriate balance between the exploration and exploitation of the search space. To promote exploration, in contrast to other replacement strategies, new individuals are always allowed to enter the population. Moreover, we combine an elitist approach with a strategy for preserving the population diversity by replacing the worst individuals and duplicated-fitness individuals with new offspring:

- *Controlled replacement for crossover*. After applying controlled selection, we have a set of high-quality solutions G as the first parents and a set of random solutions R as the second parents. The crossover operator generates a set C of n_c children. The first $n_c/2$ children are introduced to P and replace the individuals in the set of random parents R. The other $n_c/2$ children are introduced in $P\backslash(B\cup R)$ with the following rules: (a) if there are individuals with duplicated fitness, replace them with new offspring; (b) if there are children that have not been inserted into P, then introduce them, replacing the worst solutions.
- *Controlled replacement for mutation*. When applying the mutation operator, some of the best individuals are cloned to preserve the best solutions. Every clone can be entered into the population in two ways: (a) if there are solutions with duplicated fitness, then the clone will replace one of these; and (b) if the first alternative is not possible, then the clone will replace the worst solution.

Note that if x individuals (solutions) have the same fitness value, our controlled replacement strategy attempts to replace $x-1$ individuals with new offspring. Observe that the fact that two individuals have the same fitness does not implicate that they are duplicate chromosomes; given that looking for duplicate chromosomes is too expensive, our proposal is an approximation to this scheme.

5.3. Comparing controlled reproduction with other state-of-the-art techniques

To show the contribution of our new reproduction technique, we conducted a comparative experiment in which three selection schemes and two replacement strategies were combined with the objective to explore the performance of different reproduction techniques:

- *Ranking selection* (RS), where individuals to be recombined are selected from the best 50 individuals with a uniform probability, and a mutation operator is applied to the best n_m individuals [36].
- *Proportional selection* (PS), where individuals to be recombined are selected using a roulette wheel selection, and a mutation operator is applied to the best n_m individuals [36].
- *Controlled selection* (CS), where a selection of individuals for crossover and mutation is made with our selection scheme, which promotes survival of the best genes with the use of elitism and population diversity with two inverted rankings.
- *Replace worst* (RW), where the worst n_c individuals are replaced by their progeny [33].
- *Controlled replacement* (CR), where the replacement is made with our replacement strategy, which combines an elitist approach with the replacement of duplicated fitness and the worst individuals with new offspring.

The procedures were combined in six reproduction techniques: RS+RW, RS+CR, PS+RW, PS+CR, CS+RW and CS+CR. It is important to note that when we combine RS or PS with CR, CR is not our completed replacement method, given that our strategy is linked to our controlled selection CS (with the elite group and the set of random parents R). Instead, we use an approximation by the following: (a) if there are individuals with duplicated fitness, replace them with new offspring; and (b) if there are children that have not been inserted into P, then introduce them by replacing the worst solutions. All of the algorithms were run with the same configuration, which is described in Section 6. The first two selection schemes (RS and PS) use only the first five parameters because they do not use an elite group. We experiment with the nine sets of benchmark instances. Fig. 6 shows the results that were obtained from this experiment. For every selection scheme (RS, PS and CS), Fig. 6 shows the total *relative difference* that is obtained when a given selection scheme is combined with one of the two replacement strategies (RW and CR).

From Fig. 6, it can be observed an improvement in the effectiveness when we used our replacement strategy CR rather than RW, by introducing population diversity with the replacement of individuals who have repeated fitness. We can also see the superiority of our controlled reproduction technique (CS+CR), which improves the effectiveness of RS+CR and PS+CR by 198% and 463%, respectively. These results confirm that our strategy by which to balance the selective pressure and population diversity tends to lead to a higher effectiveness and that the use of elitism makes a major contribution to the final performance.

6. Main computational experiments

The GGA-CGT algorithm was developed in the C++ language and was compiled using Borland C++ 5.02 in the Win32 mode. The experiments were performed on a computer with an Intel

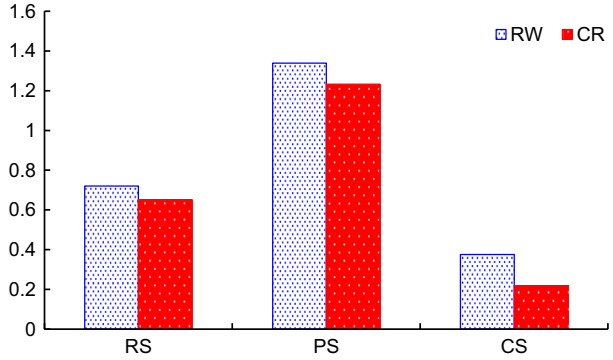

Fig. 6. Experimental results for three selection schemes (Ranking Selection RS, Proportional Selection PS, and Controlled Selection CS) and two replacement strategies (Replace Worst RW, and Controlled Replacement CR). The last bar represents the total relative difference in our controlled reproduction technique.

Table 3
Scale factors used to compare the performance of our algorithm against state-of-the-art algorithms (12 is the divisor, given that the processor that we used has this speed).

Algorithm	Processor	CPU speed	Scale factor
HGGA	R4000 Silicon Graphics workstation 50 MHz	0.097 (SPEC92 → PEC2006)	0.0972/12 = 0.008
HI_BP	Pentium IV 1.7 GHz	3.497(SPEC2000 → SPEC2006)	3.497/12 = 0.291
Pert.-SAWMBS	Intel core2 Quad CPU Q8200 2.33 GHz	18.0 (SPEC2006)	18/12 = 1.5
GGA-CGT	Intel Core2 Duo processor E6300 1.86 GHz	12 (SPEC2006)	12/12 = 1

Core 2 Duo processor E6300 1.86 GHz. Similar to most of the previous work on BPP [6,8], for each instance, a single execution of the algorithm GGA-CGT was run, with the initial seed for the random number generation set to 1. In all of the experiments, the following parameters were used for our GA implementation:

(a) Population size $|P| = 100$
(b) Maximal number of generations $max\_gen = 500$
(c) Number of individuals to be recombined $n_c = 20$
(d) Number of individuals to mutate $n_m = 83$
(e) The number of bins to eliminate n_b in the mutation operator for the non-cloned solutions is calculated using a rate of change of $k = 1.3$
(f) Number of individuals in the elite set $|B| = 10$
(g) The number of bins to eliminate n_b in the mutation operator when a solution was cloned is calculated using a rate of change of $k = 4$
(h) Maximal age for an individual to be cloned $life\_span = 10$

The first two parameter values were established considering the values used by other GAs that solved BPP; they use a population size of 100 individuals and a maximum number of generations of between 1000 and 10,000 [7,17,19]. The other parameter values were tuned by means of an experimental approach [37] that was based on covering arrays to obtain a minimal representative set of sample configurations to assess the impact of the parameters that control the performance of the algorithm. For the sake of brevity, we did not include the tuning process; however, in Section 6.3, the impact of some of the parameter values on the global performance of the algorithm is analyzed.

To assess the performance of the HGGA-PP algorithm, we experimented with different types of published problems, which are considered to be very hard. The trial benchmark had been elected by different authors to compare the effectiveness of their proposals with other algorithms that were state-of-the-art; it includes 1615 standard instances in which the number of items n varies within [50, 1000] and the intervals of the weights are within $(0, c]$. Such benchmark instances can be found on recognized websites [27–30].

6.1. Speed of different computers

A comparison between a new algorithm and several previously reported algorithms cannot always be accomplished directly, due to the variability of the different processors that are used in the computational tests. We address this problem by calculating *scale factors*, which are used to compare the performance of the algorithms as if they were running on the same computer. For determining the scale factors of each processor, we calculate the average CPU speed from the data that is reported on the SPEC standard benchmark. Because the processors that were used in the comparison are reported in different SPEC benchmarks, a preliminary treatment was required for scaling between benchmarks. Table 3 presents the scale factors for each of the processors when used by the reference algorithms, which is a CPU speed ratio that is calculated with respect to our computer (see CPU speed benchmarks available at http://www.spec.org/results).

For the experiments in this section, we adopted a Unified Computational Time (UCT) to express the conversion of the original CPU times taken from the literature into comparable units [38,39]. This convention is needed to do a fair comparison of the effectiveness of the different approaches evaluated in the experiments. The UCT of each algorithm is calculated by multiplying the time that is reported in the literature by its corresponding scale factor presented in Table 3. As explained above, the scale factors were obtained considering our computer (an Intel Core2 Duo processor E6300 1.86 GHz) as a reference machine. The comparative tables in this section show a column for each algorithm's computational time expressed in UCT, except for our algorithm GGA-CGT, where the CPU time and UCT are equivalent because its scale factor is by definition equal to 1 (see column 4 in Table 3).

6.2. Comparing GGA-CGT with an existing grouping genetic algorithm

In Section 4, we compared our new genetic operators with those proposed by Falkenauer, showing the superiority of our new methods: FF-$\tilde{n}$, gene-level crossover and adaptive mutation. In this section, we compare the GGA-CGT algorithm with Falkenauer's hybrid grouping genetic algorithm for BPP (HGGA), which includes a local optimization that is inspired by the dominance criterion [7]. Table 4 summarizes Falkenauer's results as well as those obtained by our GGA-CGT algorithm over the same benchmark instances (instances proposed by Falkenauer). Columns 1 and 2 list the sets of instances and the number of instances in every set. The next columns show, for each GA, the number of instances in which the algorithm obtained an optimal solution (Column opt.); the average running time as measured in seconds (Column time); and the maximum number of generations that was given to the algorithms (Column max gen).

Table 4 also shows the scaled average running time expressed in UCT (see Section 6.1). The scaled times in the column UCT (s) of HGGA are calculated by multiplying the times in the column time (s) by its corresponding factor presented in Table 3. Our algorithm GGA-CGT (the reference algorithm) does not have a column for UCT because this measure is equivalent to its original time. From Table 4, we can observe that with the original times, it appears that GGA-CGT is much more computationally efficient than HGGA, but when applying the conversion, the gain is less impressive (HGGA has a mean of 12 s and GGA-CGT 0.3 s). However, we can conclude that GGA-CGT is faster and more effective than HGGA by finding the optimal solutions of all of the instances in a small number of generations.

In spite of the simplicity of our GGA-CGT, the use of intelligent operators and controlled reproduction allow for us to obtain the

Table 4
Comparison between GGA-CGT and the grouping genetic algorithm HGGA of Falkenauer [7].

Class	Instances	HGGA				GGA-CGT		
		opt.	Time (s)	UCT (s)	max_gen	opt.	Time (s)	max_gen
u_120	20	18	381	3.048	2000	20	0.006	500
u_250	20	18	1337	10.696	2000	20	0.297	500
u_500	20	20	1015	8.120	5000	20	0.160	500
u_1000	20	20	7059	56.472	5000	20	0.583	500
t_60	20	18	47	0.376	1000	20	0.053	500
t_120	20	20	79	0.632	1000	20	0.147	500
t_249	20	20	728	5.824	2000	20	0.390	500
t_501	20	20	1663	13.304	2000	20	1.296	500
Total	160	154	1538.6	12.308		160	0.366	

optimal solution to all of the Falkenauer instances by using a minimal number of generations. GGA-CGT is capable of optimally solving all of these instances in less than 250 generations, but the parameter *max_gen* was configured by accounting for all of the benchmark instances.

Generally, the grouping genetic algorithm was introduced by Falkenauer [40]. Since the time of that introduction, this approach has been successfully applied to many problems in very different fields [41,42]. We have shown that a gene-level crossover is a better alternative for a grouping encoding scheme (Section 4.4); we also showed that the use of an adaptive elimination proportion improves the mutation performance (Section 4.5). Last, we showed that the performance of our grouping genetic algorithm is better than the performance of Falkenauer HGGA. Although the exact implementation details depend on the type of grouping problem, our genetic operators and our controlled reproduction technique are good alternatives to improving the grouping genetic algorithm's performance.

6.3. Comparing GGA-CGT with state-of-the-art procedures

To investigate the effectiveness of the GGA-CGT heuristic, we compared the results that were obtained by this algorithm with those that were obtained by the best approaches reported in the literature: HI_BP [8] and Perturbation-SAWMBS [6]. We also executed the code (in C) of HI_BP that was supplied by the authors on the Hard28 class. Table 5 shows, for each problem class, the number of problem instances and, for each approach, the number of instances in which the algorithm obtained an optimal solution (Column opt.). Table 5 also shows the average running time, measured in seconds (Column time) and in comparable units (column UCT); GGA-CGT does not require the UCT conversion because it is the reference algorithm (see Sections 6.1 and 6.2). In all of the cases, when an algorithm cannot find an optimal solution, the solution that was found has one more bin in relation to the optimal solution.

We note, in Table 5, the superiority of GGA-CGT while considering the class of Hard28 instances, which contain the most difficult instances for the known algorithms. For this class of test problems, it can be observed that GGA-CGT solves more instances, beating the effectiveness of the best algorithms in the state-of-the-art. It is important to remark that Perturbation-SAWMBS of Fleszar and Charalambous [6] is a fast heuristic approach that achieves almost identical results to GGA-CGT in all but the Hard28 instances set at a fraction of the time. Even with unified time, Perturbation-SAWMBS is the fastest algorithm; however, GGA-CGT significantly outperforms Perturbation-SAWMBS in the Hard28 set. For this group of difficult problems, Fleszar and Charalambous [6] note that even increasing the number of iterations of their heuristic (Perturbation-SAWMBS) from 2,000 to 100,000 does not make it possible to improve the found

Table 5
Results obtained by the best heuristic algorithms applied to BPP (time in seconds).

Class	Inst.	HI_BP			Pert.-SAWMBS			GGA-CGT	
		opt.	Time (s)	UCT (s)	opt.	Time (s)	UCT (s)	opt.	Time (s)
Uniform	80	80	0.03	0.00873	79	0.00	0.00	80	0.23
Triplets	80	80	0.98	0.28518	80	0.00	0.00	80	0.41
Data Set 1	720	720	0.19	0.05529	720	0.01	0.015	720	0.35
Data Set 2	480	480	0.01	0.00291	480	0.00	0.00	480	0.12
Data Set 3	10	10	4.60	1.3386	10	0.16	0.24	10	1.99
Was 1	100	100	0.02	0.00582	100	0.00	0.00	100	0.00
Was 2	100	100	0.02	0.00582	100	0.01	0.015	100	1.07
Gau 1	17	12	0.60	0.1746	16	0.04	0.06	16	0.27
Hard28	28	5	–		5	0.24	0.36	16*	2.40
Total	1615	1587	0.77	0.22407	1590	0.05	0.075	1602	0.35

* GGA-CGT outperforms the best algorithms on the Hard28 instances.

solutions, emphasizing the difficulty of these instances and recommending its use for future BPP heuristic studies. We obtained similar results when we increased the time of the search in the HI_BP procedure.

Similar to Alvim et al. [8] and Fleszar and Charalambous [6], we performed a robustness test by executing the GGA-CGT algorithm five times with different seeds of random numbers (8075 runs). The proposed algorithm found the optimal solutions in 7997 runs, missing the optimal solution in only 78 cases; in contrast, HI_BP [8] and Perturbation-SAWMBS [6] fail to obtain the optimal solutions in 144 and 128 cases, respectively (the numbers were adjusted while accounting for the Hard28 set). We can conclude that GGA-CGT shows high precision and robustness.

Experimental results indicate that our genetic strategy is a high-quality algorithm that produces good and robust results on a suite of test problems with very different sizes and structures, outperforming more complicated strategies in a short running time. As a consequence of the use of intelligent heuristics that preserve the best features of the solutions and a controlled reproduction technique, the proposed genetic algorithm is a simple and effective tool for different classes of bin packing instances.

6.4. Analyzing the performance of GGA-CGT

We performed an experimental analysis to study the global performance of the GGA-CGT algorithm and to observe the impact of some of the key strategies that are included in it. We studied the overall performance and the effectiveness of the algorithm in a long-term execution. We also analyzed the influence of the elite group B and the rate of change k used by the mutation operator. In all of the cases when the value for a parameter is not specified, the parameter will have the value defined in Section 6.

6.4.1. GGA-CGT overall performance

We can illustrate and analyze the average behavior of GGA-CGT by observing the evolution of the best global solutions that are produced in every generation during 20 independent executions of the algorithm over each instance. Fig. 7 shows the execution profiles of two Hard28 instances (hBPP561 and hBPP645) that could not be solved by the state-of-the-art algorithms. Each plot represents the worst, average and best fitness values of the global best solutions of the 20 executions (ordinate) in every generation of GGA-CGT (abscissa).

From Fig. 7, we can observe that, before one of the executions reaches the optimal solution (fitness ≈ 1), there is a relatively

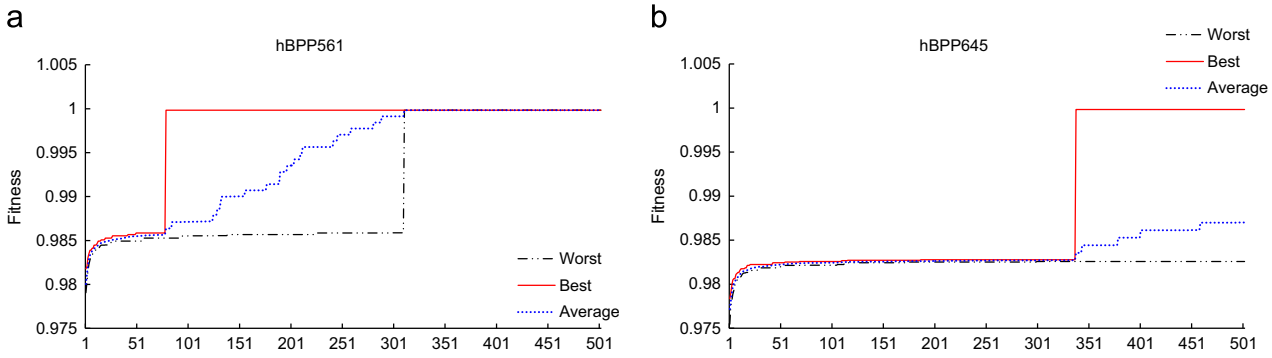

Fig. 7. Execution profiles of GGA-CGT over the two Hard28 instances: (a) hBPP561 and (b) hBPP645.

small gap between the worst and the best solutions that were found during the executions of GGA-CGT. The small variation in the GGA-CGT behavior in different executions is an indicator of the algorithm's precision and robustness. It is important to note that the largest fluctuation in the fitness is still small given the nature of the fitness function; a large leap means that the size in the best solution was improved in one bin, which in these cases means that the optimal solution was found after a period in which the genetic operators worked to improve the package of the current bins to reduce the size of the solutions. Similar behaviors were obtained using all of the benchmark instances.

As seen in Fig. 7, the hBPP561 instance was optimally solved in all of the executions in a short time, while for the hBPP645 instance, the optimum was found in some executions using a major number of generations. Observing the behavior of GGA-CGT over these similar instances opens up the question of what are the properties of these instances that impact their degree of difficulty.

6.4.2. GGA-CGT long-term execution

We have investigated how the values of *max_gen* influence the heuristic performance. Table 6 summarizes the results; for each problem class and each value of *max_gen*, it shows the probability of finding optimal solutions with GGA-CGT. The average probability of finding optimal solutions increases from 97.39 to 99.62, while the maximum number of generations increases from 50 to 10,000,000. These results prove the algorithm's precision and robustness because they show that GGA-CGT can reach the optimal solution of a huge variety of BPP instances. On the other hand, this study allowed for us to identify which instances show a higher degree of difficulty for the algorithm. It would be interesting to know the characteristics that make a difference in the degree of difficulty within each class of instances, to be able to define appropriate strategies that allow for us to improve the algorithm performance. Again, we observe the difficulty of the Hard28 instances, where the algorithm requires a larger effort to determine the optimal solutions.

6.4.3. Influence of the FF-$\tilde{n}$ heuristic

In Section 4.3, we showed the effect of exploiting knowledge about the problem to obtain a high-quality initial population. In this section, we show the impact of the FF-$\tilde{n}$ heuristic on the final performance of the GGA-CGT algorithm. When we execute our GGA-CGT with FF over the benchmark instances, its average performance is lower than with FF-$\tilde{n}$ (see Table 5); the algorithm finds the optimal solution in only 1598 instances. This experiment shows that the use of the FF-$\tilde{n}$ heuristic instead of FF allows us to obtain an advantage of 4 optimal solutions.

To evaluate all of the potential of our FF-$\tilde{n}$ heuristic, we have performed some experimental comparisons over large-scale instances with different proportions of large items. We generate 25 class instances, which are referred to as u-w1-wn, and they are available at https://sth-se.diino.com/lauracruzreyes/REPOSITORY/BPP-INSTANCES; each class includes 50 instances. For all of the 1250 instances, the number of items is $n=5000$, and the bin capacity is $c=1000$. For each class, given a lower weight w_1 and the upper weight w_n, the item weights were generated to be uniformly distributed in the interval $[w_1, w_n]$. Table 7 shows the performance of two versions of our GGA-CGT algorithm where the population was generated with FF-$\tilde{n}$ and FF, respectively. Each row indicates the lower weight w_1, and each column indicates the upper weight w_n, which were used to generate the instances of the corresponding class. For each class of instances $[w_1, w_n]$, the table shows the average proportion of large items $\tilde{n}/n$ as well as the average number of generations (Gen) that was used by the algorithm before it met its stopping criterion (see the algorithm in Fig. 1) for its two versions. The last rows of the table show the efficiency (Iter.), which is the average number of generations used by the algorithm, and the effectiveness (Eff.), which is the percentage of best solutions obtained by each version of the algorithm.

From Table 7, we can observe that the performance of our GGA-CGT improves when we switch from FF to FF-$\tilde{n}$ and that the impact of FF-$\tilde{n}$ is higher when the proportion of large items increases. In spite of its simplicity, the strategy of first packing the $\tilde{n}$ large items makes a major contribution to the algorithm's performance.

6.4.4. Influence of the elite group B

The elite group is a critical element in our controlled reproduction technique because it controls the selective pressure of the algorithm. To further examine the influence of this element on the global performance of our GGA-CGT implementation, we have performed some experimental comparisons using different sizes for this set. Fig. 8 shows the number of optimal solutions that are found when we explore the impact of the size of B over different configurations for n_c and n_m.

From Fig. 8, we can observe the impact of including an elite group in the reproduction technique of GGA-CGT; however, it is important for the size of the elite group to be small enough to contribute to evolution without an impact on the population diversity. These results confirm the importance of having a fair balance between the selective pressure and population diversity.

6.4.5. Influence of the rate of change k used by the mutation operator

In this experiment, we explore the effect of the proportion of bins to eliminate when applying our adaptive mutation operator. The mutation operator is applied to two classes of individuals: normal and cloned elite solutions. By studying the impact of the GGA-CGT parameters, we observed that the rate of change k used

Table 6
The probability of finding optimal solutions with the GGA-CGT heuristic.

max_gen	Percentage of optimal solutions for each problem class								
	U	*T*	Set 1	Set 2	Set 3	Was 1	Was 2	Gau 1	Hard28
50	97.50	90.00	99.72	100.00	90.00	100.00	100.00	88.23	3.57
100	97.50	96.25	99.72		100.00			88.23	28.57
500	100.00	100.00	100.00					94.11	57.14
5000								94.11	60.71
10000								94.11	64.28
100000								94.11	71.42
1000000								94.11	75.00
10000000								94.11	78.57

Table 7
The impact of the FF-$\tilde{n}$ and FF heuristics on the final performance of the GGA-CGT algorithm.

w_1	w_n														
	600			700			800			900			1000		
	$\tilde{n}/n$	Gen		$\tilde{n}/n$	Gen		$\tilde{n}/n$	Gen		$\tilde{n}/n$	Gen		$\tilde{n}/n$	Gen	
		FF-$\tilde{n}$	FF		FF-$\tilde{n}$	FF		FF-$\tilde{n}$	FF		FF-$\tilde{n}$	FF		FF-$\tilde{n}$	FF
1	0.16	0.3	0.52	0.28	1.04	2.02	0.37	2.64	4.76	0.44	39.6	44.6	0.49	75.6	87.9
100	0.2	4.88	5.36	0.33	9.02	12.3	0.42	67.1	74.6	0.49	76.4	86.9	0.55	26.6	40.4
200	0.25	48.9	58.9	0.39	100	100	0.49	48.8	77	0.57	15.5	36.1	0.62	24.2	39.7
300	0.33	49.7	51.6	0.49	12.8	40.0	0.59	6.4	29.4	0.66	6.5	29.2	0.71	12.1	27.6
400	0.49	1.2	13.7	0.66	1.28	12.7	0.74	1.18	14.4	0.79	0.1	19.5	0.83	0.9	17.2
Iter.		21.0	26.0		24.8	33.4		25.2	40.0		27.6	43.3		27.9	42.6
Eff.		1	0.99		1	0.99		1	1		1	0.99		1	0.98

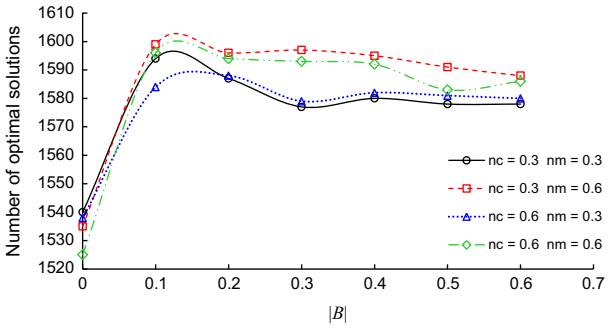

Fig. 8. The effect of the elite group size |*B*| over different configurations of GGA-CGT applied to all benchmark instances.

to calculate the number of bins to eliminate n_b in the mutation operator must be configured differently for normal and cloned individuals. Fig. 9 includes two plots that show the impact of different values of k when a mutation is applied to normal and cloned individuals. From the plots, we observe how, in contrast to the case of non-cloned individuals, it is important to use a higher elimination proportion when we mutate cloned solutions. It is important to remember that all of the cloned solutions come from elite solutions; in this case, a higher elimination proportion in the mutation operator allows for it to escape from a local optimum and explore other good zones of the search space. These results emphasize the importance of making a configuration of the principal parameters and allow for us to understand their impact on the algorithm behavior, to improve the final performance.

7. Conclusions and future work

A new grouping genetic algorithm was developed for the Bin Packing Problem; this algorithm is referred to as GGA-CGT. The algorithm includes heuristic strategies that promote the transmission of the best genes of the chromosomes and that allow for exploration of the search space. GGA-CGT controls the selection of individuals, to create a balance between the selective pressure and population diversity, avoiding the premature convergence of the algorithm and obtaining better solutions in a small number of generations.

Experimental results confirm that GGA-CGT produces good and robust results on a suite of test problems that have very different sizes and structures. GGA-CGT outperforms the results obtained by the state-of-the-art algorithms on the Hard28 set, which is the class of instances that were reported to date as the most difficult. The efficiency of the proposed algorithm is very high when compared to the number of iterations that are required by previous population strategies [7,15,16,18,19]. We are aware of the large number of parameters of GGA-CGT that must be estimated. For this reason, we are working on reducing the number of parameters that must be adjusted. Furthermore, to decrease the tuning effort, we are developing a new method for parameter tuning. We are also developing a new encoding scheme to reduce the search space that is generated by isomorphic solutions; we believe that this approach will improve the execution time of our genetic algorithm.

The study of the results that were obtained by the best algorithms for the BPP solution revealed that there still are standard instances that have a high degree of difficulty; for these instances, the included strategies in the algorithms do not appear to lead to better solutions. It was also observed that none of the strategies in the state-of-the-art have been analyzed to explain the reasons for the good or bad performance. It is important to identify which are the characteristics that distinguish the BPP instances and that could be the cause of its degree of difficulty. Furthermore, it is necessary to understand the algorithms' behavior and to identify the strategies that allow for them to reach their performance. It is expected that the work presented in this study represents a guideline for studying the performance of heuristic algorithms. This knowledge can be used to develop new intelligent procedures for solving NP-hard problems.

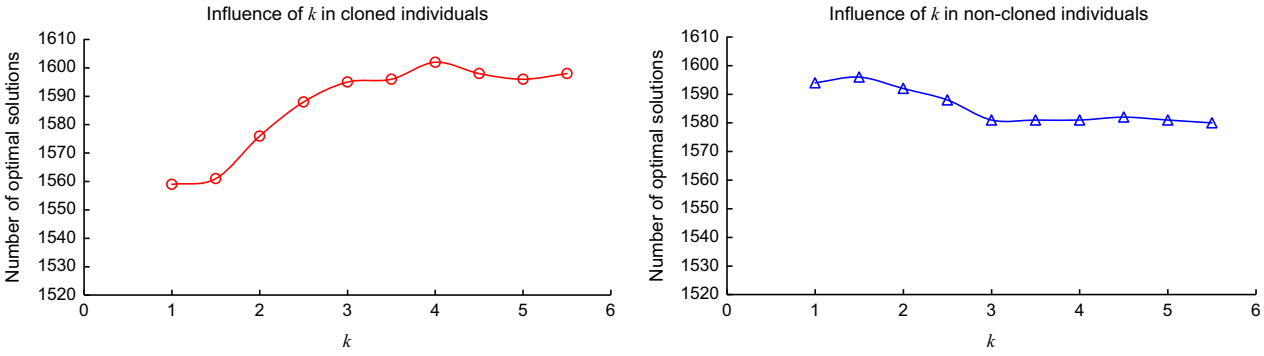

Fig. 9. The effect of the rate of change k over the cloned and non-cloned individuals.

Last, the introduction of a new set of grouping genetic operators and a new reproduction technique opens up an interesting range of possibilities for future research. Currently, we are working on applying a GGA that is similar to the GGA-CGT algorithm to other grouping problems.

Acknowledgments

This research study was partially funded by the following projects: CONACYT 58554, Cálculo de Covering Arrays; 51623, Fondo Mixto CONACyT-Gobierno del Estado de Tamaulipas.

We would like to thank CONACYT, COTACYT, DGEST, TECNM, and ITCM for their support of this research project.

References

[1] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York: W. H. Freeman & Co.; 1979.
[2] Crainic TG, Perboli G, Rei W, Tadei R. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. Comput Oper Res 2011;38(11):1474–82.
[3] Perboli G, Gobbato L, Perfetti F. Packing problems in transportation and supply chain: new problems and trends. Proc: Soc Behav Sci 2014;111:672–81.
[4] Wäscher G, Haussner H, Schumann H. An improved typology of cutting and packing problems. Eur J Oper Res 2007;183(3):1109–30.
[5] Schoenfield JE. Fast exact solution of open bin packing problems without linear programming. Draft, US Army Space & Missile Defense Command, Huntsville, Alabama, USA; 2002.
[6] Fleszar K, Charalambous C. Average-weight-controlled bin-oriented heuristics for the one-dimensional bin-packing problem. Eur J Oper Res 2011;210(2):176–84.
[7] Falkenauer E. A hybrid grouping genetic algorithm for bin packing. J Heuristics 1996;2(1):5–30.
[8] Alvim ACF, Ribeiro CC, Glover F, Aloise DJ. A hybrid improvement heuristic for the one-dimensional bin packing problem. J Heuristics 2004;10(2):205–29.
[9] Martello S, Toth P. Knapsack problems: algorithms and computer implementations. New York: John Wiley & Sons; 1990.
[10] Martello S, Toth P. Lower bounds and reduction procedures for the bin packing problem. Discret Appl Math 1990;28(1):59–70.
[11] Scholl A, Klein R, Jürgens C. BISON: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem. Comput Oper Res. 1997; 24(7): 627-645.
[12] Schwerin P, Wäscher G. A new lower bound for the bin-packing problem and its integration to MTP. Pesqui Oper 1999;19:111–29.
[13] Fleszar K, Hindi KS. New heuristics for one-dimensional bin-packing. Comput Oper Res 2002;29(7):821–39.
[14] Gupta JND, Ho JC. A new heuristic algorithm for the one-dimensional bin packing problem. Prod Plan Control 1999;10(6):598–603.
[15] Levine J, Ducatelle F. Ant colony optimization and local search for bin packing and cutting stock problems. J Oper Res Soc 2004;55(7):705–16.
[16] Bhatia AK, Basu SK. Packing Bins using multi-chromosomal genetic representation and better-fit heuristic. Neural information processing, lecture notes in computer science, 2004; 3316:181-186.
[17] Singh A, Gupta AK. Two heuristics for the one-dimensional bin-packing problem. OR Spectr 2007;29(4):765–81.
[18] Stawowy A. Evolutionary based heuristic for bin packing problem. Comput Ind Eng 2008;55(2):465–74.
[19] Rohlfshagen P, Bullinaria JA. Nature inspired genetic algorithms for hard packing problems. Ann Oper Res 2010;179:393–419.
[20] Loh KH, Golden B, Wasil E. Solving the one-dimensional bin packing problem with a weight annealing heuristic. Comput Oper Res 2008;35(7):2283–91.
[21] Gómez-Meneses P, Randall M. A hybrid extremal optimisation approach for the bin packing problem. artificial life: borrowing from biology, Lecture notes in computer science, 2009; 5865:242-251.
[22] Lewis R. A general-purpose hill-climbing method for order independent minimum grouping problems: a case study in graph colouring and bin packing. Comput Oper Res 2009;36(7):2295–310.
[23] Gen M, Cheng R. Genetic algorithms and engineering optimization. New York: John Wiley & Sons; 2000.
[24] Falkenauer E, Delchambre A. A genetic algorithm for bin packing and line balancing. In: Proceedings of the IEEE international conference on robotics and automation (ICRA '92), vol. 2. Nice; 1992. p. 1186–92.
[25] Johnson DS. Fast algorithms for bin packing. J Comput Syst Sci 1974;8 (3):272–314.
[26] Brown EC, Sumichrast RT. Impact of the replacement heuristic in a grouping genetic algorithm. Comput Oper Res 2003;30(11):1575–93.
[27] Beasley JE. OR-library: distributing test problems by electronic mail. J Oper Res Soc 1990;41(11):1069–72.
[28] Scholl A, Klein R. Bin Packing Benchmark Data Sets. ⟨http://www.wiwi.uni-jena.de/Entscheidung/binpp/⟩.
[29] ESICUP. Euro Especial Interest Group on Cutting and Packing, one Dimensional Cutting and Packing Data Sets; 2013 ⟨http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=1⟩.
[30] CaPaD. Cutting and Packing at Dresden University, Benchmark Data Sets; 2013 ⟨http://www.math.tu-dresden.de/~capad/cpd-ti.html#pmp⟩.
[31] Brown EC, Sumichrast RT. Evaluating performance advantages of grouping genetic algorithms. Eng Appl Artif Intell 2005;18(1):1–12.
[32] Yesil C, Turkyilmaz H, Korkmaz EE. A new hybrid local search algorithm on bin packing problem. In: Proceedings of the 12th IEEE international conference on hybrid intelligent systems (HIS'12), Pune; 2012. p. 161–6.
[33] Smith J. On replacement strategies in steady state evolutionary algorithms. Evol Comput 2007;15(1):29–59.
[34] Lozano M, Herrera F, Cano JR. Replacement strategies to preserve useful diversity in steady-state genetic algorithms. Inf Sci 2008;178(23):4421–33.
[35] Blickle T, Thiele L. A comparison of selection schemes used in evolutionary algorithms. Evol Comput 1996;4(4):361–94.
[36] Zhang BT, Kim JJ. Comparison of selection methods for evolutionary optimization. Evolut Optim 2000;2(1):55–70.
[37] Rangel-Valdez N, Torres-Jimenez J, Bracho-Rios J, Quiz-Ramos P. Problem and algorithm fine-tuning – a case of study using bridge club and simulated annealing. In: Proceedings of the international joint conference on computational intelligence (IJCCI'09). Madeira: Springer; 2009. p. 302–5.
[38] Crainic T, Perboli G, Tadei R. Recent advances in multi-dimensional packing problems. In: Constantin Volosencu, editor. New technologies – trends, innovations and research; 2012. p. 91–110.
[39] Perboli G, Pezzella F, Tadei R. EVE-OPT: a hybrid algorithm for the capacitated vehicle routing problem. Math Methods Oper Res 2008;68(2):361–82.
[40] Falkenauer E. The grouping genetic algorithm—widening the scope of the GAs. Belg J Oper Res Stat Comput Sci 1992;33:79–102.
[41] Agustín-Blas LE, Salcedo-Sanz S, Ortiz-García EG, Portilla-Figueras A, Pérez-Bellido AM, Jiménez-Fernández S. Team formation based on group technology: a hybrid grouping genetic algorithm approach. Comput Oper Res 2011;38(2):484–95.
[42] Höglund H. Estimating discretionary accruals using a grouping genetic algorithm. Expert Syst Appl 2013;40(7):2366–72.