A BIASED RANDOM-KEY GENETIC ALGORITHM FOR A 2D AND 3D BIN PACKING PROBLEM

JOSÉ FERNANDO GONÇALVES AND MAURICIO G.C. RESENDE

Abstract. We present a novel multi-population biased random-key genetic algorithm (BRKGA) for the 2D and 3D bin packing problem. The approach uses a maximal-space representation to manage the free spaces in the bins. The proposed algorithm uses a decoder based on a novel placement procedure within a multi-population genetic algorithm based on random keys. The BRKGA is used to evolve the order in which the boxes are packed into the bins and the parameters used by the placement procedure. Two heuristic procedures are used to determine the bin and the free maximal space where each box is placed. A novel fitness function that improves significantly the quality of the solutions produced is also developed. The new approach is extensively tested on 858 problem instances from the literature. The computational experiments demonstrate not only that the approach performs extremely well, but that it obtains the best overall results when compared with other approaches published in the literature. It reduced the total number of bins used from 9803 to 9772 for the 3D instances and from 7241 to 7234 for the 2D instances.

1. Introduction

The three-dimensional bin packing problem (3D-BPP) consists in packing, with no overlap, a set of three-dimensional rectangular shaped boxes (items) into the minimum number of three-dimensional rectangular shaped bins (containers). All the bins have identical known dimensions (D, W, H) and each box i has dimensions (d_i, w_i, h_i) for $i = 1, \ldots, n$. Without loss of generality one can assume that all the input data are positive integers and that $d_i \leq D$, $h_i \leq H$ and $d_i \leq D$ for $i = 1, \ldots, n$. It is also assumed that the boxes can be rotated. Figure 1 shows an example of a bin packing problem with two bins and more than two hundred boxes. The two-dimensional bin packing problem (2B-BPP) addresses the problem for two-dimensional bins (W, H) and boxes (w_i, h_i) and can be treated as a special case of 3D-BPP when $d_i = D$ for $i = 1, \ldots, n$. According to the typology proposed by Wäscher et al. (2007), bin packing problems are classified as 3D-SBSBPP (3D-Single Bin-Size Bin Packing Problems). The problem is strongly NP-hard as it generalizes the strongly NP-hard one-dimensional bin packing problem (Martello et al., 2000).

Three-dimensional packing problems have numerous relevant industrial applications such as loading cargo into vehicles, containers or pallets, or in package design.

Date: March 30, 2012.

Key words and phrases. Bin packing, heuristic, genetic algorithm, biased random-key genetic algorithm, three-dimensional, random keys.

AT&T Labs Research Technical Report. Research supported by FundaÃ§Ã£o para a CiÃªncia e Tecnologia (FCT) project PTDC/EGE-GES/117692/2010.

The 3D-BPP can also arise as a subproblem of other complex problems not only in packing and cutting but also in some scheduling problems.

An exact method for the 3D-BPP that uses a two-level branch & bound procedure was proposed by Martello et al. (2000). Their proposal initially only solved robot-packable problems (den Boef et al., 2005), but it was subsequently modified to solve the general problem (Martello et al., 2007). Fekete and Schepers (1997) and Fekete and Schepers (2004) define an implicit representation of the packing by means of Interval Graphs (IGs), the Packing Class (PC) representation. The authors consider the relative position of the boxes in a feasible packing and, from the projection of the items on each orthogonal axis, they define a graph describing the overlappings of the items in the container.

A new class of lower bounds was introduced by Fekete and Schepers (1997). The authors extend the use of dual feasible functions, first introduced by Johnson (1973), to two- and three-dimensional packing problems, including 3D-SBSBPP. Boschetti (2004) proposed a lower bound that dominates previous ones. These bounds make use of new dual feasible functions. Boschetti and Mingozzi (2003a;b) propose new lower bounds for the two-dimensional case.

Several constructive and meta-heuristic algorithms have been designed for solving large bin packing problems. Faroe et al. (2003) proposed a *guided local search* (GLS) heuristic for 3D-BPP and 2D-BPP. This heuristic is based on the iterative solution of constraint satisfaction problems. Starting with an upper bound on the number of bins obtained by a greedy heuristic, the algorithm iteratively decreases the number of bins, each time searching for a feasible packing of the boxes using the GLS method. Lodi et al. (1999; 2002) developed tabu search algorithms based on new constructive procedures for the two- and three-dimensional cases and Lodi et al. (2004) propose a unified tabu search code for general multi-dimensional bin packing problems. More recently, Crainic et al. (2009)developed a two-level tabu search algorithm, in which the first level aims to reduce the number of bins and the second optimizes the packing of the bins, using the representation proposed by Fekete and Schepers (2004) and Fekete et al. (2007).

For the two-dimensional bin packing problem (2D-BPP), Boschetti and Mingozzi (2003b) developed an effective constructive heuristic that assigns a score to each box, considers the boxes according to decreasing values of the corresponding scores, updates the scores using a specified criterion, and iterates until either an optimal solution is found or a maximum number of iterations has been reached. Monaci and Toth (2006) designed a set-covering-based heuristic approach in which in a first phase a large number of columns are generated by heuristic procedures and by the exact algorithm of Martello and Vigo (1998) with a time-limit. In the second phase these columns are used for solving a set-covering problem which results in the solution of the original bin packing problem. Parreño et al. (2010) propose a new hybrid GRASP/VND algorithm for solving the three-dimensional bin packing problems which can also be directly applied to the two-dimensional case. The constructive phase is based on a maximal-space heuristic developed for the container loading problem. In the improvement phase, several new moves are designed and combined in a VND structure.

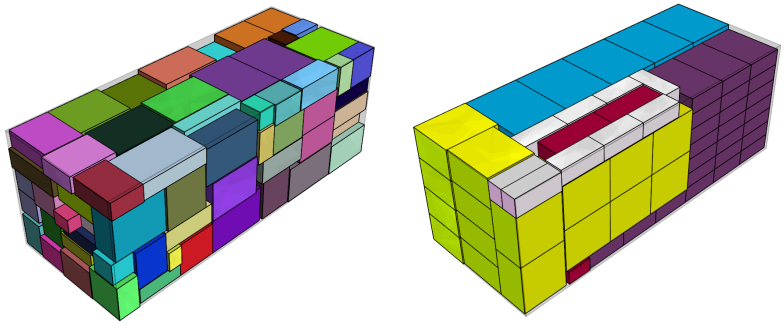

FIGURE 1. Example of a Bin Packing Problem with two bins.

3D-SBSBPP is NP-hard in the strong sense. Therefore, when large instances must be saolved, heuristics are the methods of choice. In this paper we present a novel multi-population biased random-key genetic algorithm (BRKGA) for the 2D-BPP and the 3D-BPP. The approach uses a maximal-space representation to manage the free spaces in the bins. The proposed algorithm uses a novel placement procedure as a component of the decoder within a multi-population genetic algorithm based on random keys. The BRKGA is used to evolve the order in which the boxes are packed into the bins and the parameters used by the placement procedure.

The remainder of the paper is organized as follows. In Section 2 we introduce the new approach. We describe the BRKGA, the new placement strategy, the new fitness function, and the paralell implementation. In Section 3, we report on computational experiments, and in Section 4 make concluding remarks.

2. Biased random-key genetic algorithm

We begin this section with an overview of the proposed solution process. This is followed by a discussion of the biased random-key genetic algorithm, including detailed descriptions of the solution encoding and decoding, evolutionary process, fitness function, and parallel implementation.

2.1. **Overview.**

The new approach is based on a constructive heuristic algorithm which places the boxes, one at a time, in the bins. A new bin is opened whenever a box does not fit in at least one of the bins that are already open. The management of the feasible placement positions is based on a list of empty *maximal-spaces* as described in Lai and Chan (1997). A $2D$ or $3D$ empty space is maximal if it is not contained in any other space in the bin. Each time a box is placed in an empty maximal-space, new empty maximal-spaces are generated. The new approach proposed in this paper combines a multi-population biased random-key genetic algorithm, a new placement strategy, and a novel fitness function.

The role of the genetic algorithm is to evolve the encoded solutions, or *chromosomes*, which represent the *box packing sequence* (BPS), the vector of *placement heuristics* (VPH) and the vector of *box orientations* (VBO) used for packing the boxes into the bins. For each chromosome, the following phases are applied to decode the chromosome:

(1) *Decoding of the box packing sequence*: This first phase decodes part of the chromosome into the *BPS*. i.e. the sequence in which the boxes are packed into the bins.
(2) *Decoding of placement heuristics*: The second phase decodes part of the chromosome into the vector of placement heuristics (VPH) to be used by the placement procedure.
(3) *Decoding of box orientations*: The third phase decodes part of the chromosome into the vector of box orientations (VBO) to be used by the placement procedure.
(4) *Placement strategy*: The fourth phase makes use of *BPS*, *VPH* and *VBO*, defined in phases 1, 2, and 3, and constructs a packing of the boxes into the bins.
(5) *Fitness evaluation*: The final phase computes the fitness of the solution (or measure of quality of the bin packing). For this phase we developed a novel measure of fitness which improves the quality of the solutions significantly.

Figure 2 illustrates the sequence of steps applied to each chromosome generated by the BRKGA.

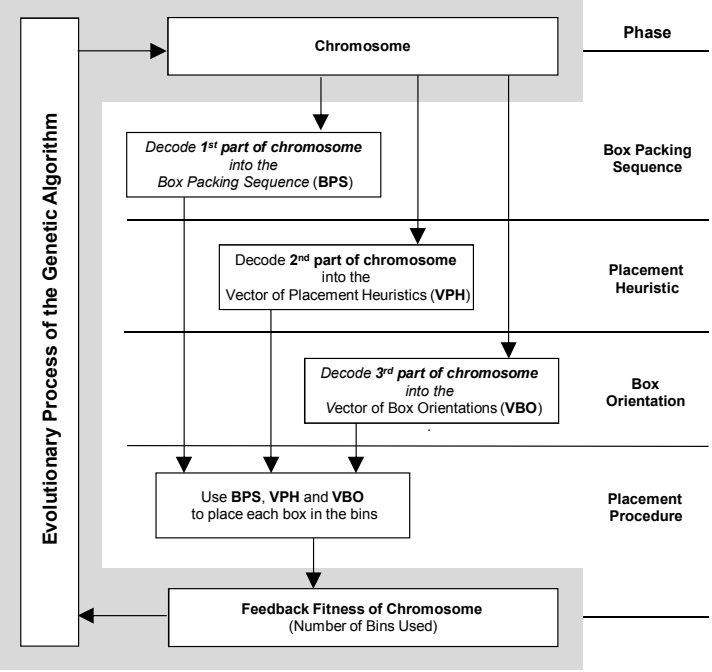

FIGURE 2. Architecture of the algorithm.

The remainder of this section describes the genetic algorithm, the decoding procedure, and the placement strategy in detail.

2.2. **Biased random-key genetic algorithm.** Genetic algorithms with random keys, or *random-key genetic algorithms* (RKGA), for solving sequencing problems

were introduced in Bean (1994). In a RKGA, chromosomes are represented as vectors of randomly generated real numbers in the interval $[0, 1]$. The *decoder*, a deterministic algorithm, takes as input a chromosome and associates with it a solution of the combinatorial optimization problem for which an objective value or fitness can be computed.

A RKGA evolves a *population* of random-key vectors over a number of *generations* (iterations). The initial population is made up of p vectors of r random keys. Each component of the solution vector, or random key, is generated independently at random in the real interval $[0, 1]$. After the fitness of each individual is computed by the decoder in generation g, the population is partitioned into two groups of individuals: a small group of p_e *elite* individuals, i.e. those with the best fitness values, and the remaining set of $p - p_e$ *non-elite* individuals. To evolve a population g a new generation of individuals is produced. All elite individual of the population of generation g are copied without modification to the population of generation $g + 1$. *RKGAs* implement mutation by introducing *mutants* into the population. A mutant is a vector of random keys generated in the same way that an element of the initial population is generated. At each generation, a small number p_m of mutants is introduced into the population. With $p_e + p_m$ individuals accounted for in the population $g + 1$, $p - p_e - p_m$ additional individuals need to be generated to complete the p individuals that make up population $g + 1$. This is done by producing $p - p_e - p_m$ offspring solutions through the process of *mating* or *crossover*.

A *biased random-key genetic algorithm*, or BRKGA (Gonçalves and Resende, 2011b), differs from a RKGA in the way parents are selected for mating. While in the RKGA of Bean (1994) both parents are selected at random from the entire current population, in a BRKGA each element is generated combining a parent selected at random from the elite partition in the current population and one from the non-elite partition. Repetition in the selection of a mate is allowed and therefore an individual can produce more than one offspring in the same generation. As in RKGAs, *parameterized uniform crossover* (Spears and Dejong, 1991) is used to implement mating in BRKGAs. Let ρ_e be the probability that an offspring inherits the vector component of its elite parent. Recall that r denotes the number of components in the solution vector of an individual. For $i = 1, \ldots, r$, the i-th component $c(i)$ of the offspring vector c takes on the value of the i-th component $e(i)$ of the elite parent e with probability ρ_e and the value of the i-th component $\bar{e}(i)$ of the non-elite parent $\bar{e}$ with probability $1 - \rho_e$.

When the next population is complete, i.e. when it has p individuals, fitness values are computed for all of the newly created random-key vectors and the population is partitioned into elite and non-elite individuals to start a new generation.

A BRKGA searches the solution space of the combinatorial optimization problem indirectly by searching the continuous r-dimensional hypercube, using the decoder to map solutions in the hypercube to solutions in the solution space of the combinatorial optimization problem where the fitness is evaluated.

To specify a biased random-key genetic algorithm, we simply need to specify how solutions are encoded and decoded and how their corresponding fitness values are computed. We specify our algorithm next by first showing how the bin packing solutions are encoded and then decoded and how their fitness evaluation is computed.

2.2.1. *Chromosome representation and decoding .* A chromosome encodes a solution to the problem as a vector of random keys. In a direct representation, a chromosome represents a solution of the original problem, and is called *genotype*, while in an indirect representation it does not, and special procedures are needed to obtain from it a solution called a *phenotype*. In the present context, the direct representation of packing patterns as chromosomes is too complicated to encode and manipulate. Instead, solutions will be represented indirectly by parameters that are later used by a decoding procedure to obtain a solution. To obtain the solution (phenotype) we use the decoding procedures described in Section 2.3.3.

Each solution chromosome is made of $3n$ genes as depicted in Figure 3. The first n genes are used to obtain the *Box Packing Sequence*, genes $n+1$ to $2n$ are used to obtain the *Vector of Placement Heuristics*, and genes $2n+1$ to $3n$ are used to obtain the *Vector of Box Orientations*. The placement procedure, described in Section 2.3.3, makes use of *BPS*, *VPH*, and *VBO* to construct a solution corresponding to the chromosome.

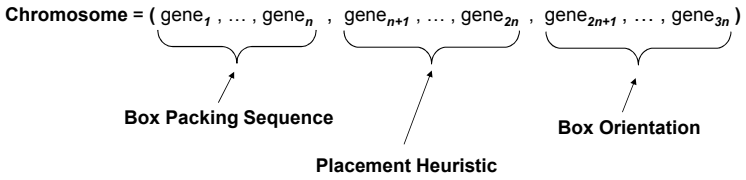

FIGURE 3. Solution encoding.

The decoding (mapping) of the first n genes of each chromosome into a box packing sequence (*BPS*) is accomplished by sorting in ascending order of gene values the corresponding boxes. Figure 4 shows an example of the decoding process for the *BPS*. In this example there are 8 boxes. The sorting of the genes in ascending order of their value produces the following $BPS = (5, 8, 3, 1, 4, 2, 6, 7)$.

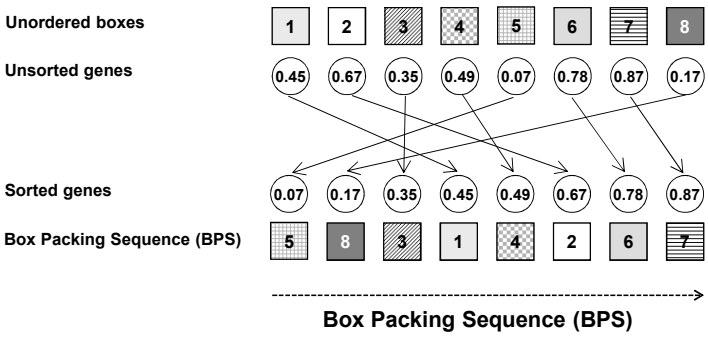

FIGURE 4. Decoding of the Box Packing Sequence.

In the placement procedure presented in Section 2.3.2 we make use of two placement heuristics; *Back-Bottom-Left* (*BBL*) and *Back-Left-Bottom* (*BLB*). The decoding of the vector of placement heuristics (*VPH*) is accomplished for $i = 1, \ldots, n$,

using the expression:

$$VPH_i = \begin{cases} BBL & \text{if} \quad gene_{n+i} \leq \frac{1}{2}, \\ BLB & \text{if} \quad gene_{n+i} > \frac{1}{2}. \end{cases}$$

The *VPH* is used by the placement procedure to select which placement heuristic to apply on each box.

The decoding of the vector of box orientations (*VBO*) is obtained for $i = 1, \ldots, n$, by

$$VBO_i = gene_{2n+i}.$$

The *VBO* is used by the placement procedure to determine what orientation will be applied to each box.

2.2.2. *Fitness function.* The evolutionary process requires a measure of solution fitness, or quality measure. The natural fitness function for this type of problem is the *number of bins* (*NB*) used by a solution. However, since different solutions can have the same *NB* value, this measure does not differentiate well the potential for improvement of solutions having the same value of *NB*.

To better differentiate the potential for improvement we developed a novel measure of fitness which we call *adjusted number of bins* (*aNB*). *aNB* combines *NB* with a measure (in the interval $]0, 1[$) of the potential for improvement of the bin packing solution. The rationale for this new measure is that if we have two solutions that use the same number of bins, then the one having the least loaded bin will have more potential for improvement.

Let *LeastLoad* be the load on the least loaded bin of a solution and let the capacity of the each bin be $BinCap = W \times H \times D$ for the $3D$ case and $BinCap = W \times H$ for the $2D$ case. The value of the adjusted number of bins is given by

$$aNB \; = \; NB \; + \; \frac{LeastLoad}{BinCap}.$$

The computational results in Section 3 show that this novel measure of fitness improves the quality of the solutions in a significant way.

2.2.3. *Multi-population strategy.* Our BRKGA uses a multi-population strategy, where several populations are evolved independently in parallel. In this strategy all populations exchange good-quality chromosomes after a pre-determined number of generations. When evaluating possible interchange strategies, we observed that exchanging too many chromosomes, or exchanging them too frequently, often leads to the disruption of the evolutionary process. Therefore, we adopt a strategy that, after a pre-determined number of generations, inserts the overall two best chromosomes (from the union of all populations) into all populations. In Section 3 we show how this choice was determined empirically.

2.3. **Placement strategy.** In the next sections we describe the main components of the placement strategy.

2.3.1. *Maximal-spaces.* While trying to place a box in the bins we use a list S of empty maximal-spaces (*EMSs*), i.e. largest empty parallelepiped spaces available for filling with boxes. Maximal-spaces are represented by their vertices with minimum and maximum coordinates ($\{x_i, y_i, z_i\}$ and $\{X_i, Y_i, Z_i\}$, respectively). When searching for a place to pack a box we need to consider only the coordinates corresponding to the *EMS* vertices with minimum coordinates (x_i, y_i, z_i). To generate and keep track of the *EMSs*, we make use of the *difference process* (*DP*) of Lai and Chan (1997). Figure 5 depicts an example of the application of the *DP* process. In the example we assume that we have one box to be packed in one bin (see Figure 5a). Initially, since the bin is empty, the box is packed at the origin of the bin as shown in Figure 5b. To pack the next box, we first update the list S of empty maximal-spaces. Figure 5c shows the three new *EMSs* generated by the *DP* process. Every time a box is packed, we reapply the *DP* process to update list S before we pack the next box.

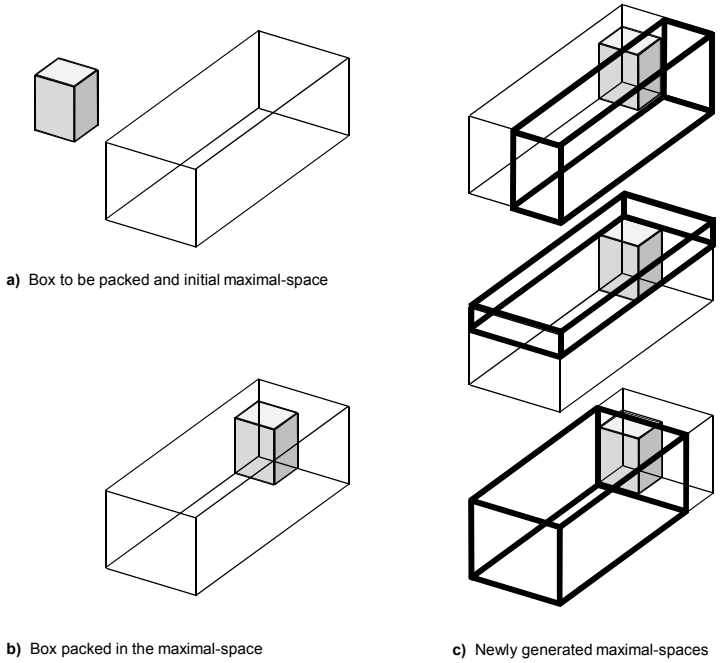

FIGURE 5. Example of difference process (*DP*).

After the list S has been updated by the *DP* one still needs to eliminate *EMSs* with infinite thinness or those that are totally inscribed by other *EMSs*. This process is the most time consuming in the packing process. To reduce the computation time for this task we added the following rules to the difference process:

- If the volume of a newly created *EMS* is smaller than the volume of each of the boxes remaining to be packed, do not add it to S (note that *EMSs* with infinite thinness will automatically be removed by this step);

- If the smallest dimension of a newly created *EMS* is smaller than the smallest dimension of each of the boxes remaining to be packed, do not add it to S (note that this rule is very important for the elimination of *EMSs* where no box fits and that are not removed by the previous rule, since they have a large volume).

With the above rules we where able to reduce, in most problem instances, the computational time by approximately 60%.

2.3.2. *Placement heuristics.* Recall from that $\{x_i, y_i, z_i\}$ denote the minimum coordinates of EMS_i. Initially we considered only the *Back-Bottom-Left* (*BBL*) procedure which orders the *EMSs* in a bin in such a way that $EMS_i < EMS_j$ if $x_i < x_j$, or if $x_i = x_j$ and $z_i < z_j$, or if $x_i = x_j$, $z_i = z_j$, and $y_i < y_j$, and then chooses the first *EMS* in which the box to be packed fits (using any of the possible orientations). Figure 6 shows pseudo-code for the *BBL* placement heuristic. However, as observed by Liu and Teng (1999), we noticed that some optimal solutions could not be constructed by the *BBL* placement heuristic. To overcome this weakness, we combine the *BBL* placement heuristic with a *Back-Left-Bottom* (*BLB*) placement heuristic. This heuristic orders the *EMSs* in a bin in such a way that $EMS_i < EMS_j$ if $x_i < x_j$, or if $x_i = x_j$ and $y_i < y_j$, or if $x_i = x_j$, $y_i = y_j$, and $z_i < z_j$, and then chooses the first *EMS* in which the box to be packed fits (using any of the possible orientations). In summary, our placement strategy uses two placement heuristics, the *Back-Bottom-Left* and the *Back-Left-Bottom*, to construct a packing of the boxes. The vector of placement heuristics (*VPH*), supplied by the genetic algorithm, indicates, for each box to be packed, whether it should be placed using the *BBL* or *BLB* heuristic.

2.3.3. *Placement procedure.* The placement procedure follows a sequential process which packs a box in a bin at each stage. The order in which the boxes are packed is defined by the *BPS* evolved by the BRKGA.

The procedure combines the following elements: the vectors *BPS*, *VPH*, and *VBO* defined by the BRKGA, the lists S_b of empty maximal spaces for every open bin b, and the *BBL* and *BLB* placement heuristics. Each stage is comprised of the following six main steps:

(1) Box selection;
(2) Placement heuristic selection;
(3) Bin and empty maximal space selection;
(4) Box orientation selection;
(5) Box packing;
(6) State information update.

The pseudo-code of the placement procedure is given in Figure 6. The box selection at stage i selects for packing the box in the i-th position of *BPS* (lines 4 of the pseudo-code). The placement heuristic to be used is defined by the i-th position of *VPH* (lines 5 of the pseudo-code). The selection of the bin and empty maximal space searches, in the open bins, for a maximal space where the box BPS_i fits (using the placing heuristic selected and any of the possible box orientations). As soon as a bin is found the bin search stops (first fit rule). If no bin is found a new bin is open (lines 6 to 23 of the pseudo-code). Once a bin and a maximal space is selected the box orientation selection is carried out (lines 24 to 26 of the pseudo-code). The box packing consists in packing the box BPS_i in the bin and the maximal selected

(lines 27 of the pseudo-code). The final step, state information update, consists in updating the list of empty maximal spaces, of the bin where the last box packing occurred using the *DP* procedure (lines 28 of the pseudo-code).

```
procedure BBL( BoxToPack, BIN )
1   Let N_EMS be the number of available EMSs in bin BIN;
2   Initialize X* ← D; Y* ← W; Z* ← H; EMS* ← 0;
3   for i = 1,...,N_EMS do
4       Let x, y, z be the minimum x, y, z coordinates of EMS_i;
5       if BoxToPack fits in EMS_i of bin BIN then
6           if x ≤ X* or
·               (x = X* and z ≤ Z* ) or
·               (x = X* and z = Z* and y ≤ Y* ) then
7               X* ← x, Z* ← z, Y* ← y ;
8               EMS* = EMS_i;
9           end if
10      end if
11  end for
12  return EMS* // returns 0 if it was not possible
·                  to pack BoxToPack in bin BIN;
end BBL
```

FIGURE 6. Pseudo-code of the Back-Bottom-Left (BBL) procedure.

2.4. **Parallel implementation.** Parallelization applies only to the task that performs the evaluation of the chromosome fitness since it is the most time consuming. Since the tasks related with the other steps of the GA consume very little time they are not parallelized. This type of parallelization is easy to implement and in multi-core CPUs allows for a large reduction in computational times (almost a linear speed-up with the number of cores). The parallel implementation of our heuristic was done using the OpenMP Application Program Interface (API) which supports multi-platform shared-memory parallel programming in C/C++.

3. NUMERICAL EXPERIMENTS

We next report on results obtained on a set of experiments conducted to evaluate the performance of the multi-population biased random-key genetic algorithm for the bin packing problem (BRKGA-BPP) proposed in this paper.

3.1. **Benchmark algorithms.** We compare BRKGA-BPP with the six approaches listed in Table 1. These approaches are the most effective in the literature to date.

```
procedure PLACEMENT (BPS, VPH, VBO)
    // ** Initialization
1   Let B be the set of open bins and NB the number of open bins;
2   B ← {1}, NB ← 0;

3   for i = 1, ..., n do
·       // ** Box selection
4       BoxToPack ← BPS_i // select the box in the i^th position of BPS;

·       // ** Placement Heuristic selection
5       PlaceHeur = VPH_i;

·       // ** Bin and Empty Maximal Space selection
6       EMS ← 0, BinSelected ← 0;
7       for k = 1, ..., NB do
·           // try to pack box BoxToPack in a EMS of bin k using PlaceHeur;
8           if PlaceHeur = BBL then
9               EMS ← BBL(BoxToPack, k);
10          else if PlaceHeur = BLB then
11              EMS ← BLB(BoxToPack, k);
12          end if

13          if EMS > 0 then // box is packable in EMS of bin k;
14              BinSelected ← k ;
15              exit for // stop for cycle, go to 18;
16          end if
17      end for

18      if BinSelected = 0 then // open a new empty bin;
19          BinSelected ← NB + 1;
20          B ← B ∪ {BinSelected};
21          NB ← NB + 1;
22          EMS ← 1; // pack BoxToPack at the origin of the new empty bin;
23      end if

·       // ** Box Orientation selection
24      Let BOs be a vector with all the possible packable orientations
·               of box BoxToPack in EMS of bin BinSelected;
25      Let nBOs be number of box orientations in vector BOs ;
26      Let BO* = BOs(⌈VBO_BoxToPack × nBOs⌉) be the box orientation
·               selected to pack BoxtoPack in EMS of bin BinSelected;

·       // ** Box packing
27      Pack BoxToPack at the origin of maximal space EMS
·               of bin BinSelected using orientation BO*;

·       // ** State Information update
28      Update the list of EMSs of bin BinSelected using
·               the DP procedure of Lai and Chan (1997);
29  end for
end PLACEMENT
```

FIGURE 7. Pseudo-code for the PLACEMENT procedure.

TABLE 1. Efficient approaches used for comparison.

Approach	Source of approach	Type of method
TS3	Lodi et al. (1999) and Lodi et al. (2002)	Tabu Search (2D/3D)
HBP	Boschetti and Mingozzi (2003b)	Heuristic (2D)
GLS	Faroe et al. (2003)	Guided Local Search (2D/3D)
SCH	Monaci and Toth (2006)	Set Covering Based Heuristic (2D)
TS^2PACK	Crainic et al. (2009)	Parallel Tabu Search (3D)
GVND	Parreño et al. (2010)	GRASP/VND (2D/3D)

3.2. **Test problem instances.** A standard benchmark set of 320 problems generated by Martello et al. (2000) was used for testing the 3D bin packing algorithm. The instance generator is available at `http://www.diku.dk/~pisinger/codes.html`. These instances are organized into 8 classes with 40 instances each, 10 instances for each value of $n \in \{50, 100, 150, 200\}$. For Classes 1–5, the bin size is $W = H = D = 100$ and there are five types of items which have w_j, h_j, and d_j uniformly random according to the intervals presented in Table 2. For Class k ($k = 1, \ldots, 5$), each item is of type k is chosen with probability 60%, and the other four types with probability 10% each. Classes 6–8 are as follows:

Class 6: bin size $W = H = D = 10$; w_j, h_j, $d_j \in [1, 10]$;
Class 7: bin size $W = H = D = 40$; w_j, h_j, $d_j \in [1, 35]$;
Class 8: bin size $W = H = D = 100$; w_j, h_j, $d_j \in [1, 100]$.

TABLE 2. Type characterization.

Type 1:	$w_j \in \left[1, \frac{1}{2}W\right]$,	$h_j \in \left[\frac{2}{3}H, H\right]$,	$d_j \in \left[\frac{2}{3}D, D\right]$;
Type 2:	$w_j \in \left[\frac{2}{3}W, \frac{1}{2}W\right]$,	$h_j \in \left[1, \frac{1}{2}H\right]$,	$d_j \in \left[\frac{2}{3}D, D\right]$;
Type 3:	$w_j \in \left[\frac{2}{3}W, \frac{1}{2}W\right]$,	$h_j \in \left[\frac{2}{3}H, H\right]$,	$d_j \in \left[1, \frac{1}{2}D\right]$;
Type 4:	$w_j \in \left[\frac{1}{2}W, W\right]$,	$h_j \in \left[\frac{1}{2}H, H\right]$,	$d_j \in \left[\frac{1}{2}D, D\right]$;
Type 5:	$w_j \in \left[1, \frac{1}{2}W\right]$,	$h_j \in \left[1, \frac{1}{2}H\right]$,	$d_j \in \left[1, \frac{1}{2}D\right]$.

For testing the 2D bin packing algorithm we used the following sets of instances which were also used to evaluate the other benchmark algorithms by their authors.

TABLE 3. Datasets used for testing the 2D bin packing algorithm.

Class	Description
bwmv	Includes 500 instances generated by Berkey and Wang (1987) and by Martello and Vigo (1998). These instances are divided into 10 classes where each class comprises 50 instances, 10 for each value of $n \in \{20, 40, 60, 80, 100\}$. All instances, and the corresponding best known solution values, are available at `http://www.or.deis.unibo.it/research_pages/ORinstances/2BP.html`.
cgcut	Proposed by Christofides and Whitlock (1977). Available from ORLIB library.
gcut, ngcut	Proposed by Beasley (1985a;b) (these instances are two-dimensional cutting problems that were transformed into 2D-BPP as described in Faroe et al. (2003)). Available from the ORLIB library.
beng	Proposed by Bengtsson (1982), available in PackLib2 (Fekete and van der Veen, 2007), `http://mo.math.nat.tu-bs.de/packlib/index.html`.

3.3. **GA configuration.** The configuration of genetic algorithms is oftentimes more an art form than a science. In our past experience with genetic algorithms based on the same evolutionary strategy (see Gonçalves and Almeida (2002), Ericsson et al. (2002), Gonçalves and Resende (2004), Gonçalves et al. (2005), Buriol et al. (2005), Buriol et al. (2007), Gonçalves (2007), Gonçalves et al. (2009), Gonçalves et al. (2009), Fontes and Gonçalves (2009) Gonçalves and Resende (2011a), Festa et al. (2010), Gonçalves and Resende (2011b), Goncalves and Sousa (2011), Gonçalves et al. (2011), Silva et al. (2011), Gonçalves and Resende (2012a), Gonçalves and Resende (2012b)), we obtained good results with values of TOP, BOT, and *CrossoverProbability* (*CProb*) in the intervals shown in Table 4.

TABLE 4. Range of parameters in past implementations.

Parameter	Interval
TOP	$0.10 - 0.25$
BOT	$0.15 - 030$
Crossover Probability (CProb)	$0.70 - 0.80$

For the population size, we have obtained good results by indexing it to the dimension of the problem, i.e. we use small size populations for small problems and larger populations for larger problems. With this in mind, we conducted a small pilot study to obtain a reasonable configuration (sixteen 3D instances, two from each class with $n = 200$). We tested all the combinations of the following values:

- $TOP \in \{0.10, 0.15, 0.20, 0.25\}$; $BOT \in \{0.15, 0.20, 0.25, 0.30\}$; $CProb \in \{0.70, 0.75, 0.80\}$;
- *Population size* with 10, 20, 30, and 40 times the number of items in the problem instance.

For each of the possible configurations, we made three independent runs of the algorithm (with three distinct seeds for the random number generator) and computed

the average total value. The configuration that minimized the sum, over the pilot problem instances, was $TOP = 10\%$, $BOT = 15\%$, $CProb = 0.7$, and *Population size* $= 30$ times the number of boxes in the problem instance. After some experimentation with the problem instances in the pilot study we came to the conclusion that using five parallel populations and exchanging information every 15 generations was a reasonable configuration for this type of problem. The configuration presented in Table 5 was held constant for all experiments and all problem instances. The computational results presented in the next section demonstrate that this configuration not only provides excellent results in terms of solution quality but is also very robust.

TABLE 5. Configuration of parameters for BRKGA-BPP.

Parameter	Value
$p =$	$30 \times n$
$p_e =$	$0.10 \times p$
$p_m =$	$0.15 \times p$
$\rho_e =$	0.70
Number of populations =	5
Exchange information between pops =	Every 15 generations
Fitness =	aNB = Adjusted Number of Bins (to minimize)
Stopping Criterion =	300 generations

3.4. **Computational results.** Algorithm BRKGA-BPP was implemented in C++ and the computational experiments were carried out on a computer with a Intel Core i7-2630QM @2.0 GHz CPU running the Linux operating system (Fedora release 16).

As described above, BRKGA-BPP allows each box to use, if possible, several orientations (six orientations for the 3D and two orientations for the 2D). However, the other benchmark algorithms only use one orientation. To make the comparison fair we constrained our algorithm (BRKGA-BPP) to use only the orientation that the other benchmark algorithms use. To show the potential of BRKGA-BPP when all the box orientations are allowed we have also included in the tables columns with header $6r$ and $2r$ which correspond to the results that BRKGA-BPP obtains when all rotations are allowed respectively for 3D and 2D instances (as expected, the results are better).

The computational results corresponding to 3D instances are shown in Table 6, which compares the solutions obtained by BRKGA-BPP with the other benchmark algorithms referred in Table 1. Each row of the table gives the average values for the instances of each class-size. Column 4 shows L_2, the corresponding lower bound obtained by Martello et al. (2000). Columns 5 to 8 contain details of the BRKGA-BPP: the column with header $6r$ represents the average best solution found when all six rotations are allowed and the fitness function used is aNB; the column with header NB represents the average best solution found when no rotations are allowed and fitness function used is NB; the column with header aNB represents the average

best solution found when no rotations are allowed and fitness function used is *aNB*; the column with header *Time* represents the average time to the best solution. The last four rows show the results, first for all classes for a complete comparison with *TS3* and *GVND* and then for classes 1, 4, 5, 6, 7, 8 which allow the comparison with TS^2PACK and *GLS*.

The results show that BRKGA-BPP consistently equals or outperforms algorithms *TS3*,*GVND*, and *GLS*. In relation to algorithm TS^2PACK the results show that BRKGA-BPP obtains better values for 13 subclasses, obtains equal values for 9 subclasses, and obtains worst values for only 2 subclasses (4-100 and 8-100). The statistical analysis of the results using the Wilcoxon test for the matched pairs BRKGA-BPP $<$ TS3, BRKGA-BPP $<$ GVND, BRKGA-BPP $<$ TS^{2}PACK, and BRKGA-BPP $<$ GLS show that BRKGA-BPP is significantly better than all the other algorithms obtaining, respectively, p-values of 0.00002045, 0.003482, 0.001068, and 0.0003597, for the paired comparisons.

The computational results corresponding to the 2D instances are shown in Tables 7 and 8, which compare the solutions obtained by BRKGA-BPP with the other benchmark algorithms referred in Table 1. In Table 7 each row gives the average values for the instances of each class-size. Column 4 shows LB^*, the lower bound reported by Monaci and Toth (2006), computed by applying all the lower-bounding procedures from the literature and an exact algorithm for a long computing time. Columns 5 to 8 contain details for BRKGA-BPP: the column with header $2r$ represents the average best solution found when two rotations are allowed and the fitness function used is *aNB*; the column with header *NB* represents the average best solution found when no rotations are allowed andfitness function used is *NB*; the column with header *aNB* represents the average best solution found when no rotations are allowed and fitness function used is *aNB*, the column with header *Time* represents the average time to the best solution. The last five rows show the results for a complete comparison with *GVND*, *SCH*, *GLS*, *TS3*, and *HBP*. Table 8, has the same structure as Table 7 and compares BRKGA-BPP with the other benchmark algorithms on a set of two-dimensional instances which are well-known in the literature. The results in Table 7 show that the BRKGA-BPP algorithm consistently equals or outperforms algorithms *GVND*, *SCH*, *GLS*, *TS3*, and *HBP* in all subclasses. The statistical analysis of the results using the Wilcoxon test for the matched pairs BRKGA-BPP $<$ GVND, BRKGA-BPP $<$ SCH, BRKGA-BPP $<$ GLS, BRKGA-BPP $<$ TS^{2}PACK, and BRKGA-BPP $<$ HBP show that BRKGA-BPP is significantly better than all the other algorithms obtaining, respectively, p-values of 0.01007, 0.004305, 0.00001403, 0.00000008536, and 0.00003105 for the paired comparisons. For the subclasses in Table 8 the BRKGA-BPP obtains the same values as *GVND* and *GLS* and outperforms *SCH* and *TS3* in one of the subclasses.

TABLE 6. Results for the three-dimensional instances.

				BRKGA-BPP							
Class	*Bin size*	n	L_2	*6r*	*NB*	*aNB*	*Time (s)*	*TS3*	*GVND*	*TS²PACK*	*GLS*
1	100 × 100 × 100	50	12.5	11.5	**13.4**	**13.4**	2.1	**13.4**	**13.4**	**13.4**	**13.4**
		100	25.1	22.9	26.7	**26.6**	17.8	**26.6**	**26.6**	26.7	26.7
		150	34.7	32.0	36.6	**36.3**	45.2	36.7	36.4	37.0	37.0
		200	48.4	43.7	51.0	**50.7**	69.1	51.2	50.9	51.1	51.2
2	100 × 100 × 100	50	12.7	11.7	13.9	**13.8**	3.9	**13.8**	**13.8**	–	–
		100	24.1	22.5	25.7	**25.5**	20.5	25.7	25.7	–	–
		150	35.1	32.2	37.0	**36.7**	39.2	37.2	36.9	–	–
		200	47.5	42.9	49.6	**49.4**	91.6	50.1	**49.4**	–	–
3	100 × 100 × 100	50	12.3	11.6	**13.3**	**13.3**	4.1	**13.3**	**13.3**	–	–
		100	24.7	22.7	26.2	**25.9**	21.2	26.0	26.0	–	–
		150	36.0	32.4	37.6	**37.5**	43.6	37.7	37.6	–	–
		200	47.8	43.0	50.1	**49.8**	78.2	50.5	50.0	–	–
4	100 × 100 × 100	50	28.7	28.9	**29.4**	**29.4**	5.0	**29.4**	**29.4**	**29.4**	**29.4**
		100	57.6	58.4	59.0	59.0	26.4	59.0	59.0	**58.9**	59.0
		150	85.2	86.4	**86.8**	**86.8**	40.7	**86.8**	**86.8**	**86.8**	**86.8**
		200	116.3	118.3	**118.8**	**118.8**	60.3	**118.8**	**118.8**	**118.8**	119.0
5	100 × 100 × 100	50	7.3	7.5	**8.3**	**8.3**	6.9	8.4	**8.3**	**8.3**	**8.3**
		100	12.9	13.7	**15.0**	**15.0**	15.0	**15.0**	**15.0**	15.2	15.1
		150	17.4	18.5	20.1	**19.9**	33.7	20.4	20.1	20.1	20.2
		200	24.4	25.3	**27.1**	**27.1**	67.5	27.6	**27.1**	27.4	27.2
6	10 × 10 × 10	50	8.7	8.9	**9.8**	**9.8**	5.4	9.9	**9.8**	**9.8**	**9.8**
		100	17.5	17.9	19.0	**18.8**	25.9	19.1	19.0	19.1	19.1
		150	26.9	27.6	**29.2**	**29.2**	42.3	29.4	**29.2**	**29.2**	29.4
		200	35.0	35.5	**37.2**	**37.2**	75.0	37.7	37.4	37.7	37.7
7	40 × 40 × 40	50	6.3	6.4	**7.4**	**7.4**	6.5	7.5	**7.4**	**7.4**	**7.4**
		100	10.9	10.8	12.3	**12.2**	12.6	12.5	12.5	12.3	12.3
		150	13.7	13.7	15.5	**15.2**	27.2	16.1	16.0	15.8	15.8
		200	21.0	21.6	**23.4**	**23.4**	72.5	23.9	23.5	23.5	23.5
8	100 × 100 × 100	50	8.0	8.3	**9.2**	**9.2**	11.7	9.3	**9.2**	**9.2**	**9.2**
		100	17.5	17.5	18.9	18.9	21.4	18.9	18.9	**18.8**	18.9
		150	21.3	22.0	23.6	**23.5**	48.2	24.1	24.1	23.9	23.9
		200	26.7	27.5	29.4	**29.2**	64.0	30.3	29.8	30.0	29.9
	Total classes 1, 4-8		6840	6848	7271	**7253**		7320	7286	7298	7302
	Total classes 1-8		9242	9038	9805	**9772**		9863	9813		

Note: The best values for the versions with no rotation appear in **bold**.

TABLE 7. Results for the two-dimensional instances. Part I

				BRKGA-BPP								
]textitClass	*Bin size*	n	*LB**	2*r*	*NB*	*aNB*	*Time (s)*	*GVND*	*SCH*	*GLS*	*TS3*	*HBP*
1	10 × 10	20	7.1	6.6	**7.1**	**7.1**	0.0	**7.1**	**7.1**	**7.1**	**7.1**	**7.1**
		40	13.4	12.8	**13.4**	**13.4**	0.1	**13.4**	**13.4**	**13.4**	13.5	**13.4**
		60	19.7	19.5	**20**	**20**	0.9	**20**	**20**	20.1	20.1	20.1
		80	27.4	27	**27.5**	**27.5**	0.2	**27.5**	**27.5**	**27.5**	28.2	**27.5**
		100	31.7	31.3	**31.7**	**31.7**	3.3	**31.7**	**31.7**	32.1	32.6	31.8
2	30 × 30	20	1.0	1.0	**1.0**	**1.0**	0.0	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		40	1.9	1.9	**1.9**	**1.9**	0.1	**1.9**	**1.9**	**1.9**	2.0	**1.9**
		60	2.5	2.5	**2.5**	**2.5**	0.1	**2.5**	**2.5**	**2.5**	2.7	**2.5**
		80	3.1	3.1	**3.1**	**3.1**	0.7	**3.1**	**3.1**	**3.1**	3.3	**3.1**
		100	3.9	3.9	**3.9**	**3.9**	0.5	**3.9**	**3.9**	**3.9**	4.0	**3.9**
3	40 × 40	20	5.1	4.7	**5.1**	**5.1**	0.1	**5.1**	**5.1**	**5.1**	5.5	**5.1**
		40	9.2	9.2	**9.4**	**9.4**	0.9	**9.4**	**9.4**	**9.4**	9.7	9.5
		60	13.6	13.4	**13.9**	**13.9**	2.9	**13.9**	**13.9**	14.0	14.0	14.0
		80	18.7	18.2	**18.9**	**18.9**	3.3	**18.9**	**18.9**	19.1	19.8	19.1
		100	22.1	22	**22.3**	**22.3**	3.7	**22.3**	**22.3**	22.6	23.6	22.6
4	100 × 100	20	1.0	1.0	**1.0**	**1.0**	0.2	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		40	1.9	1.9	**1.9**	**1.9**	0.5	**1.9**	**1.9**	**1.9**	**1.9**	**1.9**
		60	2.3	2.3	**2.5**	**2.5**	2.3	**2.5**	**2.5**	**2.5**	2.6	**2.5**
		80	3	3.1	**3.1**	**3.1**	1.2	**3.1**	3.2	3.3	3.3	3.3
		100	3.7	3.7	3.8	**3.7**	1.1	3.8	3.8	3.8	4.0	3.8
5	100 × 100	20	6.5	5.9	**6.5**	**6.5**	0.2	**6.5**	**6.5**	**6.5**	6.6	**6.5**
		40	11.9	11.4	**11.9**	**11.9**	0.5	**11.9**	**11.9**	**11.9**	**11.9**	**11.9**
		60	17.9	17.2	**18**	**18**	2.7	**18**	**18**	18.1	18.2	**18**
		80	24.1	23.9	**24.7**	**24.7**	6.3	**24.7**	**24.7**	24.9	25.1	24.8
		100	27.9	27.7	28.2	**28.1**	3.4	28.2	28.2	28.8	29.5	28.7
6	300 × 300	20	1.0	1.0	**1.0**	**1.0**	0.2	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		40	1.5	1.6	1.7	**1.6**	3.5	1.7	1.7	1.8	1.9	1.8
		60	2.1	2.1	**2.1**	**2.1**	1.2	**2.1**	**2.1**	2.2	2.2	**2.1**
		80	3.0	3.0	**3.0**	**3.0**	2.3	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**
		100	3.2	3.2	3.4	**3.3**	3.1	3.4	3.4	3.4	3.4	3.4
7	100 × 100	20	5.5	5.2	**5.5**	**5.5**	0.1	**5.5**	**5.5**	**5.5**	**5.5**	**5.5**
		40	10.9	10.2	**11.1**	**11.1**	0.4	**11.1**	**11.1**	11.3	11.4	**11.1**
		60	15.6	14.6	15.9	**15.8**	3.1	15.9	**15.8**	15.9	16.2	16.0
		80	22.4	20.8	**23.2**	**23.2**	7.9	**23.2**	**23.2**	**23.2**	**23.2**	**23.2**
		100	26.9	25	**27.1**	**27.1**	4.0	**27.1**	**27.1**	27.5	27.7	27.4
8	100 × 100	20	5.8	5.3	**5.8**	**5.8**	0.1	**5.8**	**5.8**	**5.8**	**5.8**	**5.8**
		40	11.2	10.3	**11.3**	**11.3**	0.7	**11.3**	**11.3**	11.4	11.4	**11.3**
		60	15.9	14.7	**16.1**	**16.1**	3.6	**16.1**	16.2	16.3	16.2	16.2
		80	22.3	20.4	**22.4**	**22.4**	2.3	**22.4**	**22.4**	22.5	22.6	22.6
		100	27.4	25.2	**27.8**	**27.8**	4.5	**27.8**	27.9	28.1	28.4	28.0
9	100 × 100	20	14.3	14.3	**14.3**	**14.3**	0.1	**14.3**	**14.3**	**14.3**	**14.3**	**14.3**
		40	27.8	27.5	**27.8**	**27.8**	0.4	**27.8**	**27.8**	**27.8**	**27.8**	**27.8**
		60	43.7	43.5	**43.7**	**43.7**	0.9	**43.7**	**43.7**	**43.7**	43.8	**43.7**
		80	57.7	57.3	**57.7**	**57.7**	1.7	**57.7**	**57.7**	**57.7**	**57.7**	**57.7**
		100	69.5	69.3	**69.5**	**69.5**	2.8	**69.5**	**69.5**	**69.5**	**69.5**	**69.5**
10	100 × 100	20	4.2	4.1	**4.2**	**4.2**	1.0	**4.2**	**4.2**	**4.2**	4.3	4.3
		40	7.4	7.2	**7.4**	**7.4**	0.4	**7.4**	**7.4**	**7.4**	7.5	**7.4**
		60	9.8	9.9	**10.0**	**10.0**	3.2	**10.0**	10.1	10.2	10.4	10.2
		80	12.3	12.5	12.9	**12.8**	8.1	12.9	**12.8**	13.0	13.0	13.0
		100	15.3	15.4	15.9	**15.8**	7.9	15.9	15.9	16.2	16.6	16.2
Total classes 1-10			7173	6988	7241	**7234**		7241	7243	7284	7360	7275

Note: The best values for the versions with no rotation appear in **bold**.

The performance of the novel fitness measure (adjusted number of bins, or *aNB*), in relation to the the usual number of bins, *NB*, is also demonstrated in Tables 6 and 7 where *aNB* produces better results than *NB* for 16 3D-subclasses and for 4 2D-subclasses. Overall the use of *aNB* instead of *NB* reduced the total number of bins from 9803 to 9772 for the 3D instances and from 7241 to 7234 for the 2D instances.

In terms of computational times we cannot make any fair and meaningful comments since all the other approaches were implemented and tested on computers with different computing power. Instead, we limit ourselves to reporting the average running times approaches.

TABLE 8. Results for the two-dimensional instances. Part II

				BRKGA-BPP							
Class	*n*	Instances	LB^*	*2r*	*NB*	*aNB*	*Time (s)*	*GVND*	*SCH*	*GLS*	*TS3*
cgcut	16-62	3	9	7.33	9	**9**	1.66	**9**	**9**	**9**	**9**
gcut	10-50	13	8	7.31	8	**8**	0.25	**8**	**8**	**8**	8.31
ngcut	7-22	12	2.67	2.5	2.67	**2.67**	0.01	**2.67**	**2.67**	**2.67**	3
beng 1-8	20-120	8	6.75	6.75	6.75	**6.75**	0.19	**6.75**	6.88		
beng 9-10	160-200	2	6.5	6.5	6.5	**6.5**	1.24	**6.5**			

Note: The best values for the versions with no rotation appear in **bold**.

4. Concluding remarks

In this paper we addressed the three-dimensional bin packing problem which consists in packing, with no overlap, a set of three-dimensional rectangular shaped boxes into the minimum number of three-dimensional rectangular shaped bins. All the bins have identical known dimensions and each box i is has dimensions (d_i, w_i, h_i) for $i = 1, \ldots, n$. It is assumed that the boxes can be rotated.

A novel multi-population biased random-key genetic algorithm (BRKGA) for the 2D-BPP and 3D-BPP was developed. The approach uses a maximal-space representation to manage the free spaces in the bins (Lai and Chan, 1997). The proposed algorithm hybridizes a novel placement procedure with a multi-population genetic algorithm based on random keys. The BRKGA is used to evolve the order in which the boxes are packed into the bins and the parameters used by the placement procedure. Two heuristic procedures, *Back-Bottom-Left* and the *Back-Left-Bottom,* are used to determine the bin and the free maximal space where each box is placed. A novel fitness function that improves significantly the solutions quality is also developed.

The new approach is extensively tested on 858 problem instances. The computational experiments demonstrate not only that the approach performs extremely well but that it obtains the best overall results when compared with other approaches published in the literature. It reduced the total number of bins used from 9803 to 9772 for the 3D instances and from 7241 to 7234 for the 2D instances.

References

J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. on Computing*, 6:154–160, 1994.

J.E. Beasley. Algorithms for unconstrained two-dimensional guillotine cutting. *J. of the Operational Research Society*, 36:297–306, 1985a.

J.E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33:49–64, 1985b.

J.O. Berkey and P.Y. Wang. Two-dimensional finite bin-packing algorithms. *J. of the operational research society*, 38:423–429, 1987.

M.A. Boschetti. New lower bounds for the three-dimensional finite bin packing problem. *Discrete Applied Mathematics*, 140:241–258, 2004.

M.A. Boschetti and A. Mingozzi. The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case. *4OR: A Quarterly J. of Operations Research*, 1:27–42, 2003a.

M.A. Boschetti and A. Mingozzi. The two-dimensional finite bin packing problem. Part II: New lower and upper bounds. *4OR: A Quarterly J. of Operations Research*, 1:135–147, 2003b.

L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46: 36–56, 2005.

L.S. Buriol, M.G.C. Resende, and M. Thorup. Survivable IP network design with OSPF routing. *Networks*, 49:51–64, 2007.

N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25:30–44, 1977.

T.G. Crainic, G. Perboli, and R. Tadei. TS^{2}PACK: A two-level tabu search for the three-dimensional bin packing problem. *European J. of Operational Research*, 195:744–760, 2009.

E. den Boef, J. Korst, S. Martello, D. Pisinger, and D. Vigo. Erratum to "The three-dimensional bin packing problem": Robot-packable and orthogonal variants of packing problems. *Operations Research*, 53:735–736, 2005.

M. Ericsson, M.G.C. Resende, and P.M. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *J. of Combinatorial Optimization*, 6: 299–333, 2002.

O. Faroe, D. Pisinger, and M. Zachariasen. Guided local search for the three-dimensional bin-packing problem. *INFORMS J. on Computing*, 15:267–283, 2003.

S. Fekete and J. Schepers. A new exact algorithm for general orthogonal d-dimensional knapsack problems. In *Algorithms–ESA'97*, pages 144–156. Springer, 1997.

S.P. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29:353–368, 2004.

S.P. Fekete and J.C. van der Veen. PackLib2: An integrated library of multi-dimensional packing problems. *European J. of Operational Research*, 183:1131–1135, 2007.

S.P. Fekete, Schepers J., and VanderVeen J.C. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55:569–587, 2007.

P. Festa, J.F. Gonçalves, M.G.C. Resende, and R.M.A. Silva. Automatic tuning of GRASP with path-relinking heuristics with a biased random-key genetic algorithm. In P. Festa, editor, *Experimental Algorithms*, volume 6049 of *Lecture Notes in Computer Science*, pages 338–349. Springer, 2010.

D.B.M.M. Fontes and J.F. Gonçalves. A multi-population genetic algorithm for tree-shaped network design problems. In *IJCCI 2009 – International Joint Conference on Computational Intelligence, Proceedings*, pages 177–182, October 2009.

J.F. Goncalves and P.S.A. Sousa. A genetic algorithm for lot sizing and scheduling under capacity constraints and allowing backorders. *International J. of Production Research*, 49:2683–2703, 2011.

J.F. Gonçalves. A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European J. of Operational Research*, 183:1212–1229, 2007.

J.F. Gonçalves and J.R. Almeida. A hybrid genetic algorithm for assembly line balancing. *J. of Heuristics*, 8:629–642, 2002.

J.F. Gonçalves and M.G.C. Resende. An evolutionary algorithm for manufacturing cell formation. *Computers and Industrial Engineering*, 47:247–273, 2004.

J.F. Gonçalves and M.G.C. Resende. A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. *J. of Combinatorial Optimization*, 22:180–201, 2011a.

J.F. Gonçalves and M.G.C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *J. of Heuristics*, 17:487–525, 2011b.

J.F. Gonçalves and M.G.C. Resende. A biased random-key genetic algorithm for job-shop scheduling. Technical report, AT&T Labs Research, Florham Park, New Jersey, March 2012a.

J.F. Gonçalves and M.G.C. Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers and Operations Research*, 39:179–190, 2012b.

J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European J. of Operational Research*, 167: 77–95, 2005.

J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A genetic algorithm for the resource constrained multi-project scheduling problem. *European J. of Operational Research*, 189:1171–1190, 2009.

J.F. Gonçalves, M.G.C. Resende, and J.J.M. Mendes. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *J. of Heuristics*, 17:467–486, 2011.

D.S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

K.K. Lai and J.W.M. Chan. Developing a simulated annealing algorithm for the cutting stock problem. *Computers and Industrial Engineering*, 32:115–127, 1997.

D. Liu and H. Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European J. of Operational Research*, 112:413–420, 1999.

A. Lodi, S. Martello, and D. Vigo. Approximation algorithms for the oriented two-dimensional bin packing problem. *European J. of Operational Research*, 112: 158–166, 1999.

A. Lodi, S. Martello, and D. Vigo. Heuristic algorithms for the three-dimensional bin packing problem. *European J. of Operational Research*, 141:410–420, 2002.

A. Lodi, S. Martello, and D. Vigo. TSpack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research*, 131:203–213, 2004.

S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management science*, 44:388–399, 1998.

S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48:256–267, 2000.

S. Martello, D. Pisinger, D. Vigo, E.D. Boef, and J. Korst. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software*, 33:7:1–7:12, 2007.

M. Monaci and P. Toth. A set-covering-based heuristic approach for bin-packing problems. *INFORMS J. on Computing*, 18:71–85, 2006.

F. Parreño, R. Alvarez-Valdes, J.F. Oliveira, and J.M. Tamarit. A hybrid GRASP/VND algorithm for two-and three-dimensional bin packing. *Annals of Operations Research*, 179:203–220, 2010.

D.M. Silva, R.M.A. Silva, G.R. Mateus, J.F. Gonçalves, M.G.C. Resende, and P. Festa. *An iterative refinement algorithm for the minimum branch vertices problem*, volume 6630 of *Lecture Notes in Computer Science*. Springer, 2011.

W.M. Spears and K.A. Dejong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

G. Wäscher, H. Haussner, and H. Schumann. An improved typology of cutting and packing problems. *European J. of Operational Research*, 183:1109–1130, 2007.

(José Fernando Gonçalves) Faculdade de Economia do Porto / NIAAD, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal.

E-mail address: jfgoncal@fep.up.pt

(Mauricio G.C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.

E-mail address, M.G.C. Resende: mgcr@research.att.com