A biased random key genetic algorithm for 2D and 3D bin packing problems

José Fernando Gonçalves, Mauricio G.C. Resende

A R T I C L E   I N F O
Article history:
Received 29 February 2012
Accepted 10 April 2013
Available online 18 April 2013

Keywords:
Bin packing
Genetic algorithm
Three-dimensional
Random keys

A B S T R A C T
In this paper we present a novel biased random-key genetic algorithm (BRKGA) for 2D and 3D bin packing problems. The approach uses a maximal-space representation to manage the free spaces in the bins. The proposed algorithm hybridizes a novel placement procedure with a 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 new placement heuristics are used to determine the bin and the free maximal space where each box is placed. A novel fitness function that improves significantly the solution quality is also developed. The new approach is extensively tested on 858 problem instances and compared with other approaches published in the literature. The computational experiment results demonstrate that the new approach consistently equals or outperforms the other approaches and the statistical analysis confirms that the approach is significantly better than all the other approaches.

1. Introduction

The three-dimensional bin packing problem (3D-BPP) consists in packing, with no overlapping, 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 input data are positive integers and that \(d_i \leq D, w_i \leq W, h_i \leq H\) for \(i = 1, \ldots, n\). It is assumed that the boxes can be rotated. Fig. 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 for cutting and packing problems proposed by Wäscher et al. (2007) bin packing problems can be classified as Single Stock-Size Cutting Stock Problem (SSSCP) for weakly heterogeneous item sets or as 3D-SBSBPP (3D-Single Bin-Size Bin Packing Problems) for strongly heterogeneous item sets. The bin packing problem addressed in this paper is classified as 3D-SBSBPP (3D-Single Bin-Size Bin Packing Problems). The 2D-BPP and 3D-BPP are strongly NP-hard as they generalize 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 packaging design. 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 (Park et al., 1996; Hartmann, 2000).

An exact method for the 3D-SBSBPP that uses a two-level Branch & Bound method was proposed by Martello et al. (2000). Initially their proposal only solved robot-packable problems (den Boef et al., 2005), but later it was modified for solving the general problem (Martello et al., 2007). Fekete and Schepers (1997, 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 the most recent lower bound, which introduces new dual feasible functions. This new bound dominates previous ones. Boschetti and Mingozzi (2003a, 2003b) 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 heuristic for 3D-SBSBPP and 2D-SBSBPP, 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) have developed tabu search algorithms based on new constructive procedures for two-dimensional and three-dimensional cases and in 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, using the representation proposed for nD-SBSBPP by Fekete and Schepers (2004) and Fekete et al. (2007), in which the first level aims to reduce the number of bins and the second optimizes the packing of the bins.

For the two-dimensional bin packing problem (2D-SBSBPP), Boscetti 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 is 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 execution of the exact algorithm by Martello and Vigo (1998) with a time-limit. In the second phase these columns are used for solving a set-covering problem which gives the solution to the original bin packing problem. Parreño et al. (2010) propose a new hybrid GRASP/VND algorithm for solving the 3D-SBSBPP bin packing problem which can also be directly applied to the two-dimensional case (2D-SBSBPP). 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. Mack and Bortfeldt (2012) present a straightforward heuristic for the 3D-SSSCSP where all items may be rotated and the guillotine cut constraint has to be respected. The heuristic is based on a method for the container loading problem following a wall-building approach and on a method for the one-dimensional BPP.

3D-SBSBPP is NP-hard in the strong sense. Therefore, when large instances are considered, heuristics are the methods of choice. In this paper we present a novel biased random-key genetic algorithm (BRKGA) for the 2D-SBSBPP and 3D-SBSBPP. The approach uses a maximal-space representation to manage the free spaces in the bins. The proposed algorithm hybridizes a novel placement procedure with a 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, describing in detail the BRKGA, the novel placement strategy, the novel fitness function, and the parallel implementation. Finally, 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 when the box that we are trying to place does not fit in the bins that are already open (note that all the bins stay open until all boxes are packed). 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 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) 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 box orientations: The second phase decodes part of the chromosome into the vector of box orientations VBO to be used by the placement procedure.
3. Placement strategy: The third phase makes use of BPS and VBO, defined in phases 1 and 2, and constructs a packing of the boxes into the bins.
4. 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.

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

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.

Random key GAs are particularly attractive for sequencing problems and/or when the chromosomes have several parts (see for example Gonçalves and Almeida, 2002; Gonçalves and Resende, 2004, or Gonçalves and Sousa, 2011). Unlike traditional GAs, which need to use special repair procedures to handle permutations or sequences, RKGAs move all the feasibility issues into the objective evaluation procedure and guarantee that all offspring formed by crossover are feasible solutions. When the chromosomes have several parts traditional GAs need to use different genetic operators for each part. However, since RKGAs use parametrized uniform crossovers (instead of the traditional one-point or two-point crossover), they do not need to have different genetic operators for each part.

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, 2011), 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 BRKGAs each element is generated combining a parent selected at random from the elite partition in the current population and one selected at random from the rest of the population. 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 \( p_r \) 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 \( i = 1, ..., r \), 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 \( p_r \), and the value of the \( i \)-th component \( f(i) \) of the non-elite parent \( f \) with probability \( 1-p_r \).
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 2.3.4.

Each solution chromosome is made of \( 2n \) genes as depicted in Fig. 3. The first \( n \) genes are used to obtain the Box Packing Sequence and the genes \( n+1 \) to \( 2n \) are used to obtain the Vector of Box Orientations. The placement procedure described in Section 2.3.4 makes use of \( BPS \) and \( VBO \) to construct a solution corresponding to the chromosome.

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 the corresponding gene values, the boxes. Fig. 4 shows an example of the decoding process for the \( BPS \). In this example there are 8 boxes. The sorted genes correspond the \( BPS = (5, 8, 3, 1, 4, 2, 6) \).

![Chromosome](image)

**Box Packing Sequence**

**Box Orientation**

**Fig. 3.** Solution encoding.

![Unordered boxes](image)

**Unsorted genes**

![Sorted genes](image)

**Box Packing Sequence (BPS)**

**Fig. 4.** Decoding of the box packing sequence.

### 2.2.2. Fitness function

To evolve the solutions the evolutionary process needs a measure of solution fitness, or quality measure. A 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 \( NB \), 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 propose a novel measure of fitness which we call adjusted number of bins, \( aNB \). The \( aNB \) combines \( NB \) with a measure of the potential for improvement of the bin packing solution which has values in the interval \([0,1]\). 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, \( \text{LeastLoad} \) be the load on the least loaded bin of a solution and let the capacity of the each bin be \( \text{BinCap} = W \times H \times D \) for the 3D \((2D)\) case. The value of the adjusted number of bins is given by

\[
aNB = NB + \frac{\text{LeastLoad}}{\text{BinCap}}.
\]

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

### 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 rectangular spaces available for filling with boxes. Maximal-spaces are represented by their vertices with minimum and maximum coordinates \( (x_l, y_l, z_l) \) and \( (x_u, y_u, z_u) \) respectively. To generate and keep track of the EMSs, we make use of the difference process (DP), developed by Lai and Chan (1997). The DP process consists of the following three main steps:

- Place a box in an EMS;
- Generate new EMSs resulting from the intersection of the box being placed with the existing EMSs and remove the intersected EMSs;
- Eliminate the EMSs which have infinite thinness or those that are totally inscribed by other EMSs.

**Fig. 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 Fig. 5a). Initially, since the bin is empty, the box is packed at the origin of the bin as shown in Fig. 5b. Fig. 5c shows the three new EMSs resulting from the intersection of the box placed with the initial EMS (the empty container).

The elimination of EMSs that are totally inscribed by other EMSs is the most time consuming task in the DP process. In order 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 were able to reduce, for most problem instances, the computational time by approximately 60%.

2.3.2. Placement heuristics

When searching for a position to pack a box we will consider as candidate positions only the minimum coordinates $(x_i, y_i, z_i)$ of the feasible EMSs (an EMS will be considered feasible for a box if the box fits in it using at least one of its possible orientations). The placement heuristic will be used to choose, in a bin, the EMS where a box will be placed. Ideally the placement heuristic should produce a packing as compact as possible in order to leave free space for the remaining boxes.

Initially we considered the Back-Bottom-Left (BBL) heuristic rule. 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 looked for
improved heuristics and developed two new placement heuristics rules which use the Distance to the Front-Top-Right Corner (DFTRC) of the container as measure of compactness. The greater the DFTRC the more compact we will consider the packing of the box. The first heuristic, denoted as DFTRC-1, calculates DFTRC for the minimum coordinates \((x_i, y_i, z_i)\) of all feasible EMSs for the box being packed and chooses the EMS which maximizes DFTRC. The second heuristic, denoted as DFTRC-2, calculates, for all feasible EMSs and all feasible box orientations in each EMS, the DFTRC of the front-top-right corner of the box being packed and chooses the EMS which maximizes DFTRC. Heuristics DFTRC-1 and DFTRC-2 are illustrated using an example in Figs. 6 and 7, respectively. In the example we have a 2D partial packing of boxes 1, 3, and 4 and we want to choose a place where to pack box 2. If we use heuristic DFTRC-1 box 2 will be positioned in point A and if we use heuristic DFTRC-2 box 2 will be positioned in point C. Note that the EMSs having as minimum coordinates point B is not feasible for box 2 and that the EMSs having as minimum coordinates points A and D only allow one feasible orientation of box 2.

Fig. 8 presents the pseudo-code for the DFTRC-2 placement heuristic.

### 2.3.3. Box orientation

To select the orientation of a box we have considered two alternatives. The first alternative uses the vector VBO supplied by the BRKGA and can be combined with both of the placement heuristics DFTRC-1 and DFTRC-2. Let EMS be the empty maximal space selected for placing a box \(i\), by any of the heuristic placement heuristics, and let \(B Os\) be a vector with all the feasible orientations of the box \(i\) in EMS, the orientation selected, \(BO^*\), is given by

\[
BO^* = BO_s(\{VBO_i \times nBOs\})
\]

where \(nBOs\) is the number of box orientations in vector \(BOs\) and \(\lceil x \rceil\) is the smallest integer greater than \(x\).

The second alternative does not use any information supplied by the BRKGA and can only be used in combination with the placement heuristic DFTRC-2. In this case the orientation selected for a box is equal to the winning orientation \(BO^*\) returned by the DFTRC-2 (see Fig. 8).

### 2.3.4. Placement procedure

The placement procedure follows a sequential process which packs one 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 and VBO defined by the BRKGA, the lists \(S_b\) of empty maximal spaces for every open bin \(b\), and one of the placement heuristics defined above (DFTRC-1 or DFTRC-2). Each stage is comprised of the following five main steps:

1. Box selection;
2. Bin and empty maximal space selection;
3. Box orientation selection;
4. Box packing;
5. State information update.

We tried three alternative placement procedures: two resulting from the combination of the heuristics DFTRC-1 and DFTRC-2 with VBO supplied by the BRKGA to obtain the orientation and one resulting from the use of DFTRC-2 to select where to pack a box and its orientation. Table 1 names and summarizes the alternatives according to the heuristic placement and the box orientation method used.

As will be shown in the numerical experiments (Section 3) the alternative DFTRC-2-VBO obtains the best results.

The pseudo-code of the placement procedure for alternative DFTRC-2-VBO is given in Fig. 9. The box selection at stage \(i\) selects

**procedure** PLACEMENT \((BPS, VBO)\)

```
1. Let \(N_{EMS}\) be the number of available EMSs in bin \(BIN\);
2. Initialize maxDist\(^2\) = -1, \(BO^* = -1, EMS^* = 0\);
3. for \(i = 1, \ldots, N_{EMS} \) do
   a. Let \(x, y, z\) be the \(x, y, z\) coordinates of \(EMS_i\);
   b. for all possible orientations \(BO\) of BoxToPack do
      i. Let \(d^{BO}\) \(w^{BO}\) \(k^{BO}\) be the \(x, y, z\) dimensions of BoxToPack
         corresponding to orientation \(BO\);
      ii. if (BoxToPack fits in \(EMS_i\) using orientation \(BO\)) then
         a. Dist\(^2\) = \((d - x)^2 + (w - y)^2 + (k - z)^2\);
         b. if (Dist\(^2\) > maxDist\(^2\)) then
            i. maxDist\(^2\) = Dist\(^2\);
            ii. \(BO^* = BO\); // save the winning orientation
         c. EMS\(^*\) = EMS\(^*\); // save the winning EMS
      end if
   end for
3. end for
4. Return EMS\(^*\); // returns 0 if it was not possible
   to pack BoxToPack in bin \(BIN\);
```

**Table 1** Alternative placement procedures.

<table>
<thead>
<tr>
<th>Name</th>
<th>Placement heuristic</th>
<th>Box orientation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFTRC-1-VBO</td>
<td>DFTRC-1</td>
<td>VBO</td>
</tr>
<tr>
<td>DFTRC-2-VBO</td>
<td>DFTRC-2</td>
<td>VBO</td>
</tr>
<tr>
<td>DFTRC-2</td>
<td>DFTRC-2</td>
<td>DFTRC-2</td>
</tr>
</tbody>
</table>

**Fig. 9.** Pseudo-code for the PLACEMENT procedure.
for packing the box in the i-th position of BPS (lines 4 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, fits, using DFTRC-2 (line 9 of the pseudo-code). As soon as a bin is found, the bin search stops (first fit rule). If no bin is found, a new bin is opened (lines 15-20 of the pseudo-code). Once a bin and a maximal space is selected, the box orientation selection is carried out (lines 21–23 of the pseudo-code). The box packing consists in packing the box BPS, in the bin and the maximal selected (line 24 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 (line 25 of the pseudo-code).

2.4. Parallel implementation

The 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 GA logic consume very little time compared to decoding they were 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

In this section we report on results obtained on a set of experiments conducted to evaluate the performance of the 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 2. These approaches are the most effective in the literature to date.

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 \( w_j, h_j, d_j \) drawn at random from a uniform distribution according to the intervals presented in Table 3. For Class \( k (k = 1, \ldots, 5) \) each type of item \( k \) 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 \( W = H = D = 100 \); \( w_j, h_j, d_j \in [1, 100] \).

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 4).

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 et al., 2005, 2009, 2011; Gonçalves and Resende, 2012), we obtained good results with values of TOP, BOT, and Crossover Probability (CProb) in the intervals shown in Table 5.

For the population size, we 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 (16 3D instances, two from each class with \( n = 200 \)). We tested all the combinations of the following values:

The 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 GA logic consume very little time compared to decoding they were 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++.

### Table 2

<table>
<thead>
<tr>
<th>Approach</th>
<th>Source of approach</th>
<th>Type of method</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS3</td>
<td>Lodi et al. (1999, 2002)</td>
<td>Tabu search (2D/3D)</td>
</tr>
<tr>
<td>HBP</td>
<td>Boschetti and Mingozzi (2003b)</td>
<td>Heuristic (2D)</td>
</tr>
<tr>
<td>GLS</td>
<td>Faroo et al. (2003)</td>
<td>Guided local search (2D/3D)</td>
</tr>
<tr>
<td>SCH</td>
<td>Monaci and Toth (2006)</td>
<td>Set covering based heuristic (2D)</td>
</tr>
<tr>
<td>TS2P</td>
<td>Crain et al. (2009)</td>
<td>Parallel tabu search (3D)</td>
</tr>
<tr>
<td>GVND</td>
<td>Parrello et al. (2010)</td>
<td>GRASP/VND (2D/3D)</td>
</tr>
</tbody>
</table>

### Table 3

<table>
<thead>
<tr>
<th>Type</th>
<th>( w_j )</th>
<th>( h_j )</th>
<th>( d_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1:</td>
<td>([1, 4])</td>
<td>([1, 4])</td>
<td>([2, D])</td>
</tr>
<tr>
<td>Type 2:</td>
<td>([2, W])</td>
<td>([1, H])</td>
<td>([2, D])</td>
</tr>
<tr>
<td>Type 3:</td>
<td>([2, W])</td>
<td>([1, H])</td>
<td>([D, D])</td>
</tr>
<tr>
<td>Type 4:</td>
<td>([4, W])</td>
<td>([1, H])</td>
<td>([2, D])</td>
</tr>
<tr>
<td>Type 5:</td>
<td>([1, W])</td>
<td>([1, H])</td>
<td>([2, D])</td>
</tr>
</tbody>
</table>

### Table 4

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bwmv</td>
<td>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 <a href="http://www.or.deis.unibo.it/research_pages/ORInstances/2BP.html">http://www.or.deis.unibo.it/research_pages/ORInstances/2BP.html</a>.</td>
</tr>
<tr>
<td>gcut</td>
<td>Proposed by Beasley (1985a, 1985b). These instances are two-dimensional cutting problems that were transformed into 2D-BPP as explained by Faroo et al. (2003). Available from the ORLIB library.</td>
</tr>
</tbody>
</table>
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 \( \text{TOP} = 10\% \), \( \text{BOT} = 15\% \), \( \text{CProb} = 0.7 \), and \( \text{Population size} = 30 \) times the number of items in the problem instance.

Table 6

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>( 30 \times n )</td>
</tr>
<tr>
<td>( \rho_e )</td>
<td>( 0.10 \times p )</td>
</tr>
<tr>
<td>( \rho_m )</td>
<td>( 0.15 \times p )</td>
</tr>
<tr>
<td>( \rho_f )</td>
<td>( 0.70 )</td>
</tr>
<tr>
<td>Fitness</td>
<td>( a\text{NB}-\text{adjusted number of bins (to minimize)} )</td>
</tr>
<tr>
<td>Stopping criterion</td>
<td>( 200 ) generations</td>
</tr>
</tbody>
</table>

Table 7

<table>
<thead>
<tr>
<th>Data set</th>
<th>DFTRC-1-VBO</th>
<th>DFTRC-2-VBO</th>
<th>DFTRC-2\textsuperscript{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>7001</td>
<td>6990</td>
<td>7012</td>
</tr>
<tr>
<td>3D</td>
<td>9023</td>
<td>9009</td>
<td>9031</td>
</tr>
</tbody>
</table>

3.4. Computational results

Algorithm \( \text{BRKGA-BPP} \) was implemented in C++ and the computational experiments were carried out on a computer with an Intel Core i7-2630QM @2.0 GHZ CPU running the Linux operating system with Fedora release 16.

As described above algorithm \( \text{BRKGA-BPP} \) 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 \( \text{BRKGA-BPP} \) to use only the orientation that the other benchmark algorithms use. Too show the potential of \( \text{BRKGA-BPP} \) when all the box orientations are allowed we have also included in the tables columns with header 6\( r \) and 2\( r \) which correspond to the results that \( \text{BRKGA-BPP} \) obtains when all rotations are allowed, respectively, for 3D and 2D instances (as expected the results are better).

As mentioned above, we considered three alternative placement procedures: \( \text{DFTRC-1-VBO} \), \( \text{DFTRC-2-VBO} \), and \( \text{DFTRC-2}\textsuperscript{2} \). We
will first present the overall results obtained by each of the alternatives and then compare in detail the best alternative with the other approaches. Table 7 presents the overall number of bins used for each alternative for the 2D and 3D datasets.

Statistical tests show that the placement procedure DFTRC-2-VBO performs significantly better than the other two.

The results reported below the algorithm BRKGA-BPP refer to BRKGA-BPP using placement heuristic DFTRC-2-VBO.

The computational results corresponding to 3D instances are shown in Table 8, which compares the solutions obtained by BRKGA-BPP with the other benchmark algorithms referred in Table 2. 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 10 contain details of our BRKGA-BPP algorithm: column with header \( L_2 \) represents the average best solution found when all the six rotations are allowed and the fitness function used is \( N_B \); column with header \( NB \) represents the average best solution found when no rotations are allowed and fitness function used is \( a_B \); column with header \( a_B \) represents the average best solution found when no rotations are allowed and fitness function used is \( a_B \), column with headers \( a_C, a_T \), and \( T-200 \) represent, respectively, the average number of generations to the best solution, the average time to the best solution and the average time for 200 generations. 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 TS3P and GLS.

The results show that our BRKGA-BPP algorithm consistently equals or outperforms algorithms TS3, GVND, and GLS. In relation
to algorithm TS2P the results show that BRKGA-BPP obtains better values for 15 subclasses, obtains equal values for 7 subclasses and obtains worse 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 < TS2PACK and BRKGA-BPP < GLS show that BRKGA-BPP is significantly better than all the other algorithms obtaining p-values for all paired comparisons smaller than 0.001.

The computational results corresponding to the 2D instances are shown in Tables 9 and 10, which compare the solutions obtained by BRKGA-BPP with the other benchmark algorithms referred in Table 2. In Table 9, each row shows 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 5–10 contain details of our BRKGA-BPP algorithm: the column with header 2r represents the average best solution found when all the 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 and the 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 columns with headers aG, aT, and T-200 represent, respectively, the average number of generations to the best solution, the average time to the best solution, and the average time for 200 generations. The last five rows show the results for a complete comparison with GVND, SCH, GLS, TS3, and HBP. Table 10, has the same structure as Table 9 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 9 show that BRKGA-BPP 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 < TS2P, and BRKGA-BPP < HBP show that BRKGA-BPP is significantly better than all the other algorithms obtaining, p-values for all paired comparisons smaller than 0.01.

For the subclasses in Table 10 the BRKGA-BPP algorithm obtains the same values as GVND and GLS and outperforms SCH and TS3 in one of the subclasses.

The improved performance of the novel fitness measure, adjusted number of bins, aNB, in relation to the usual number of bins, NB, is clearly demonstrated in Tables 8 and 9 where aNB produces better results than NB for 17 3D-subclasses and for 7 2D-subclasses. Overall, the use of aNB instead of NB reduced the total number of bins from 9806 to 9777 for the 3D instances and from 7241 to 7234 for the 2D instances.

For the subclasses in Table 10 the BRKGA-BPP algorithm obtains the same values as GVND and GLS and outperforms SCH and TS3 in one of the subclasses.

4. Concluding remarks

In this paper we addressed the three-dimensional bin packing problem which consists in packing, with no overlapping, 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 has dimensions (di, wi, hi) for i = 1, ..., n. It is assumed that the boxes can be rotated.

A novel biased random-key genetic algorithm (BRKGA) for the 2D-SBSBPP and 3D-SBSBPP 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 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, DIFTRC-1 and DIFTRC-2, 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 and compared with other approaches published in the literature. The computational experiments results demonstrate that the new approach consistently equals or outperforms the other approaches and the statistical analysis confirms that the approach is significantly better than all the other approaches.

Acknowledgments

This work has been supported by funds granted by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT – Foundation for Science and Technology, project PTDC/EGE-GES/117692/2010.

References