Int. J. Production Economics 145 (2013) 500–510

Contents lists available at ScienceDirect

Int. J. Production Economics

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

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

José Fernando Gonçalves [a,*], Mauricio G.C. Resende [b]

[a] LIAAD, INESC TEC, Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
[b] Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932, USA

ARTICLE INFO

Article history:
Received 29 February 2012
Accepted 10 April 2013
Available online 18 April 2013

Keywords:
Bin packing
Genetic algorithm
Three-dimensional
Random keys

ABSTRACT

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 $d_i \le D$, $w_i \le W$, and $h_i \le H$ 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 (SSSCSP) 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.

☆Supported by Fundação para a Ciência e Tecnologia (FCT) project PTDC/EGE-GES/117692/2010.

* Corresponding author. Tel.: +351 937061379.
E-mail addresses: jfgoncal@fep.up.pt, jfgoncal@gmail.com (J.F. Gonçalves), mgcr@research.att.com (M.G.C. Resende).

0925-5273/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2013.04.019

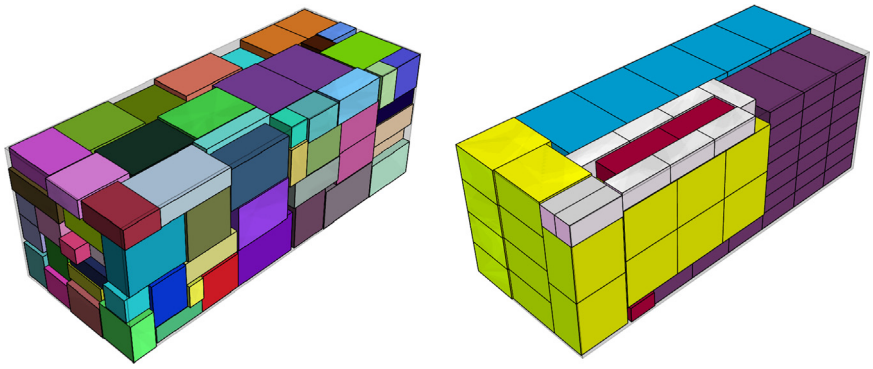

Fig. 1. Example of a bin packing problem with two bins.

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), 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 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

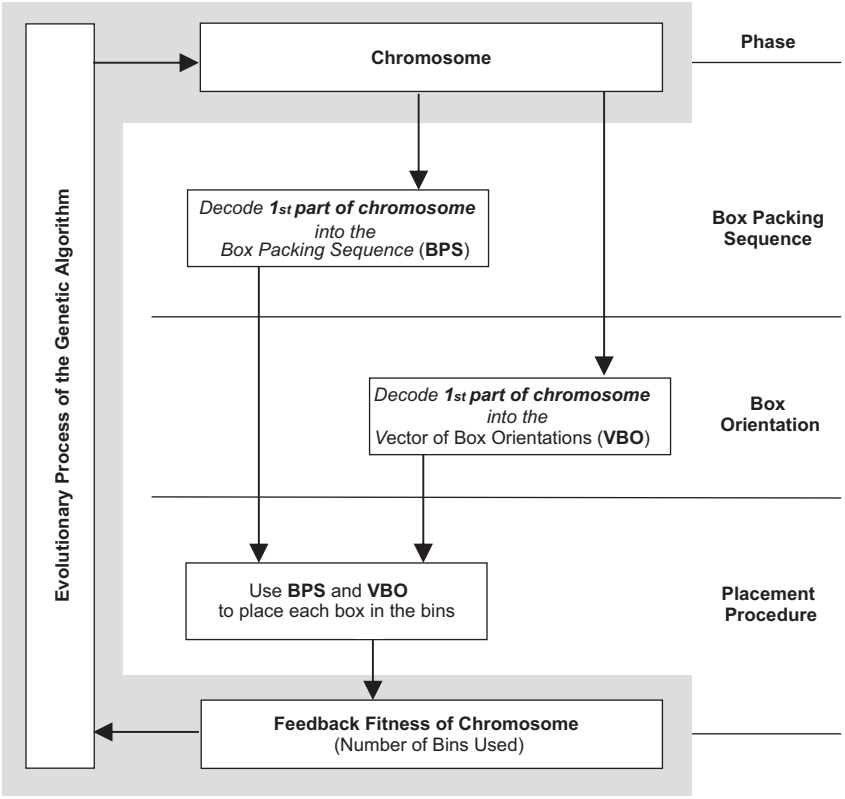

Fig. 2. Architecture of the algorithm.

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 Goncalves 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 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 2.3.4.

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 $BPS = (5, 8, 3, 1, 4, 2, 6, 7)$.

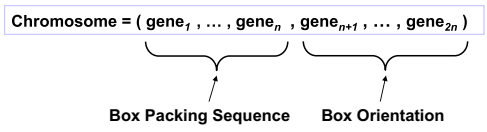

Fig. 3. Solution encoding.

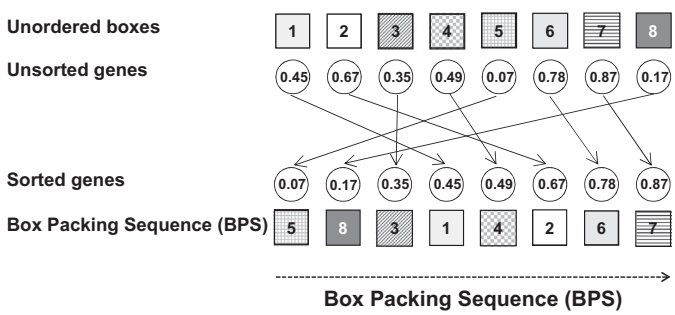

Fig. 4. Decoding of the box packing sequence.

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

$VBO_i = Gene_{n+i}$.

Note that the *VBO* does not directly indicate the orientation used for a box.

Only after having selected an *EMS* in which to place a box can the *VBO* be used to determine what orientation to use for a box (see Section 2.3.3 for more details about the decoding of the box orientation).

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, *LeastLoad* be the load on the least loaded bin of a solution and let the capacity of the each bin be $BinCap = W \times H \times D$ ($W \times H$) for the 3*D* (2*D*) 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 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_i, y_i, z_i and X_i, Y_i, Z_i 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 a *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:

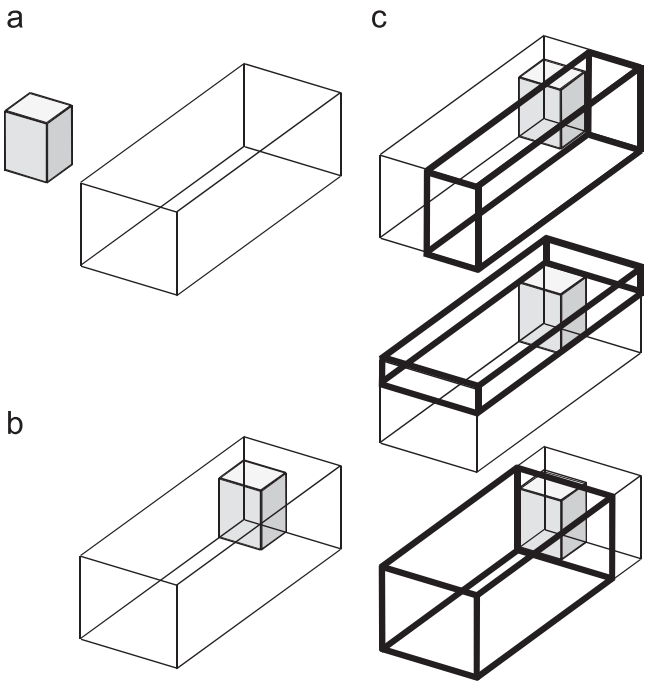

Fig. 5. Example of difference process (*DP*). a) Box to be packed and initial maximal-space; b) Box packed in the maximal-space and c) Newly generated maximal-spaces.

- 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 the 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

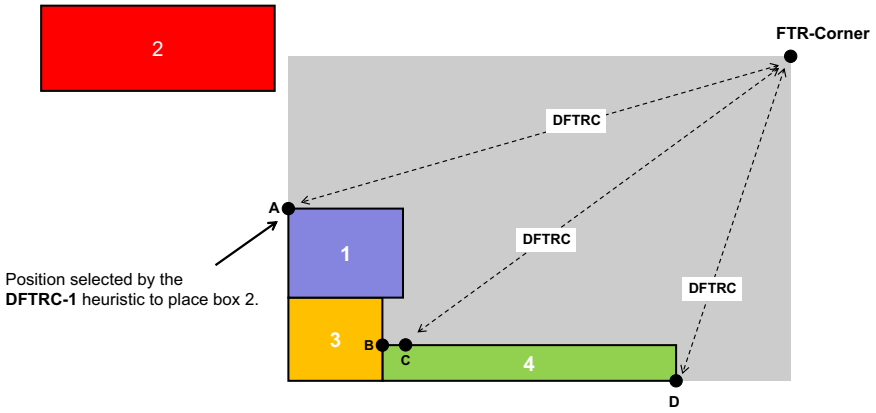

Fig. 6. Example of heuristic DFTRC-1.

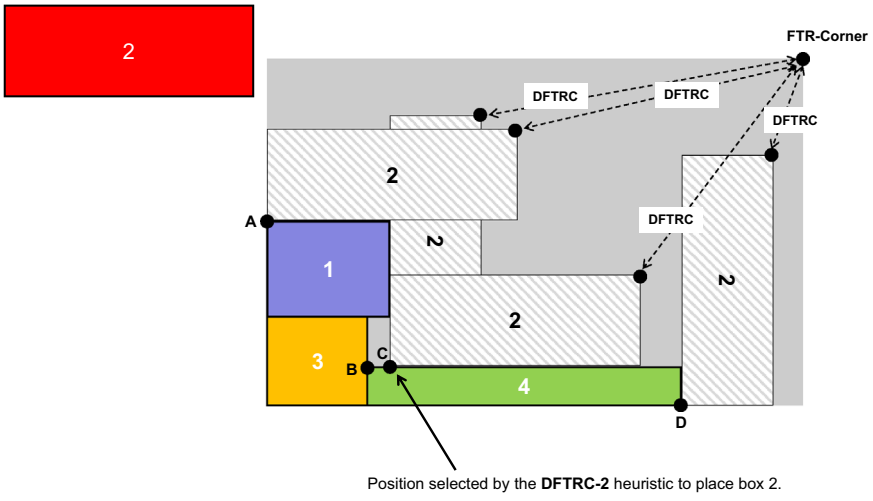

Fig. 7. Example of heuristic DFTRC-2.

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 DFFTRC-2 box 2 will be positioned in point C. Note that the *EMS* 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 *BOs* be a vector with all the feasible orientations of the box *i* in *EMS*, the orientation selected, *BO**, is given by

$$BO^* = BOs(\lceil VBO_i \times nBOs \rceil)$$

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).

```
procedure DFTRC-2( BoxToPack, BIN, BO*)
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,...,N_EMS do
4     Let x, y, z be the minimum x, y, z coordinates of EMS_i;
4     for all possible orientations BO of BoxToPack do
5       Let d^BO, w^BO, h^BO be the x, y, z dimensions of BoxToPack
·         corresponding to orientation BO ;
6       if (BoxToPack fits in EMS_i of bin BIN using orientation BO ) then
7         Dist^2 ← (D − x − d^BO)^2 + (W − y − w^BO)^2 + (H − z − h^BO)^2;
8         if (Dist^2 > maxDist^2 ) then
9           maxDist^2 ← Dist^2;
10          BO* ← BO; // save the winning orientation
11          EMS* = EMS_i;// save the winning EMS
12        end if
13      end if
14    end for
15  end for
16  Return EMS* // returns 0 if it was not possible
·               to pack BoxToPack in bin BIN;
end DFTRC-2;
```

Fig. 8. Pseudo-code of the DTRC-2 heuristic procedure.

Table 1
Alternative placement procedures.

Name	Placement heuristic	Box orientation
DFTRC*−1−*VBO	*DFTRC*-1	*VBO*
DFTRC*−2−*VBO	*DFTRC*-2	*VBO*
***DFTRC*−2**^{2}	*DFTRC*-2	*DFTRC*-2

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 *DFRTC-2-VBO* obtains the best results.

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

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

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

·       // ** 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 DFTRC-2
9           EMS =DFTRC-2(BoxToPack, k, BO*);
10          if EMS > 0 then //  box is packable in EMS of bin k;
11              BinSelected ← k ;
12              exit for // stop for cycle, go to 18;
13          end if
14      end for

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

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

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

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

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_i 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_i 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, and 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 item 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, $d_j \in [1, 100]$.

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:

Table 2
Efficient approaches used for comparison.

Approach	Source of approach	Type of method
TS3	Lodi et al. (1999, 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)
TS2P	Crainic et al. (2009)	Parallel tabu search (3D)
GVND	Parreño et al. (2010)	GRASP/VND (2D/3D)

Table 3
Type characterization.

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

Table 5
Range of parameters in past implementations.

Parameter	Interval
TOP	0.10–0.25
BOT	0.15–0.30
Crossover Probability (CProb)	0.70–0.80

Table 4
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 the ORLIB library.
gcut, ngcut	Proposed by Beasley (1985a, 1985b) (these instances are two-dimensional cutting problems that were transformed into 2D-BPP as explained by Faroe et al. (2003)). Available from the ORLIB library.
beng	Proposed by Bengtsson (1982), available in PackLib2 (Fekete et al., 2007), http://mo.math.nat.tu-bs.de/packlib/index.html.

- *TOP*∈{0.10, 0.15, 0.20, 0.25};
- *BOT*∈{0.15, 0.20, 0.25, 0.30};
- *CProb*∈{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. The configuration presented in Table 6 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 6
Configuration parameters for the *BRKGA–BPP* algorithm.

Parameter	Value
$p=$	$30 \times n$
$p_e=$	$0.10 \times p$
$p_m=$	$0.15 \times p$
$\rho_e=$	0.70
Fitness=	*aNB*=adjusted number of bins (to minimize)
Stopping criterion=	200 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 with Fedora release 16.

As described above algorithm *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. Too 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).

As mentioned above, we considered three alternative placement procedures: *DFTRC-1-VBO*, *DFTRC-2-VBO*, and *DFTRC-2*^{2}. We

Table 7
Overall results obtained by each of the alternative placement procedures.

Data set	*DFTRC-1-VBO*	*DFTRC-2-VBO*	*DFTRC-2*^{2}
2D	7001	6990	7012
3D	9023	9009	9031

Table 8
Results for the three-dimensional instances.

Class	*Bin size*	*n*	L_2	*BRKGA-BPP*						*TS3*	*GVND*	*TS2P*	*GLS*
				6r	*NB*	*aNB*	aG	aT	T-200				
1	100 × 100 × 100	50	12.5	11.8	**13.4**	**13.4**	7.3	0.1	3.5	**13.4**	**13.4**	**13.4**	**13.4**
		100	25.1	23	26.7	**26.6**	19.1	1.5	16.1	**26.6**	**26.6**	26.7	26.7
		150	34.7	31.7	36.6	**36.4**	47.5	9.8	41.3	36.7	**36.4**	37.0	37.0
		200	48.4	43.4	51	**50.8**	51.0	20.8	81.7	51.2	50.9	51.1	51.2
2	100 × 100 × 100	50	12.7	11.8	13.9	**13.8**	10.1	0.2	3.1	**13.8**	**13.8**	–	–
		100	24.1	22.5	25.7	**25.6**	34.1	2.6	15.4	25.7	25.7	–	–
		150	35.1	31.5	37	**36.6**	72.5	14.4	39.7	37.2	36.9	–	–
		200	47.5	42.5	49.6	**49.4**	52.5	20.5	78.1	50.1	**49.4**	–	–
3	100 × 100 × 100	50	12.3	11.6	**13.3**	**13.3**	8.5	0.1	3.2	**13.3**	**13.3**	–	–
		100	24.7	22.6	26.2	**25.9**	50.5	4.0	15.7	26.0	26.0	–	–
		150	36.0	32.4	37.6	**37.5**	43.6	8.7	39.7	37.7	37.6	–	–
		200	47.8	42.3	50.1	**49.8**	55.1	21.6	78.5	50.5	50.0	–	–
4	100 × 100 × 100	50	28.7	28.9	**29.4**	**29.4**	1.0	0.0	3.1	**29.4**	**29.4**	**29.4**	**29.4**
		100	57.6	58.4	59.0	59.0	1.0	0.1	17.1	59.0	59.0	**58.9**	59.0
		150	85.2	86.4	**86.8**	**86.8**	1.0	0.2	48.4	**86.8**	**86.8**	**86.8**	**86.8**
		200	116.3	118.3	**118.8**	**118.8**	1.0	0.5	101.5	**118.8**	**118.8**	**118.8**	119.0
5	100 × 100 × 100	50	7.3	7.5	8.3	**8.3**	3.4	0.1	7.1	8.4	**8.3**	**8.3**	**8.3**
		100	12.9	13.7	**15.0**	**15.0**	10.4	1.5	29.0	**15.0**	**15.0**	15.2	15.1
		150	17.4	18.6	20.1	**20.0**	43.5	15.0	68.9	20.4	20.1	20.1	20.2
		200	24.4	25.3	27.1	**27.1**	30.2	19.7	130.6	27.6	**27.1**	27.4	27.2
6	10 × 10 × 10	50	8.7	9.4	9.8	**9.7**	10.1	0.1	2.7	9.9	9.8	9.8	9.8
		100	17.5	18.9	19.0	**18.9**	14.0	0.8	11.3	19.1	19.0	19.1	19.1
		150	26.9	28.2	29.2	**29.0**	34.7	4.6	26.4	29.4	29.2	29.2	29.4
		200	35.0	33.3	**37.3**	**37.3**	63.6	15.3	48.2	37.7	37.4	37.7	37.7
7	40 × 40 × 40	50	6.3	6.4	**7.4**	**7.4**	2.8	0.1	5.9	7.5	**7.4**	**7.4**	**7.4**
		100	10.9	11.3	12.3	**12.2**	31.7	3.8	23.9	12.5	12.5	12.3	12.3
		150	13.7	14.6	15.5	**15.3**	31.1	8.9	57.1	16.1	16.0	15.8	15.8
		200	21.0	20.3	**23.4**	**23.4**	24.5	12.4	101.4	23.9	23.5	23.5	23.5
8	100 × 100 × 100	50	8.0	9.2	**9.2**	**9.2**	4.6	0.1	5.6	9.3	**9.2**	**9.2**	**9.2**
		100	17.5	18.2	18.9	18.9	9.1	1.1	23.6	18.9	18.9	**18.8**	18.9
		150	21.3	22.1	**23.6**	**23.6**	47.5	12.3	51.9	24.1	24.1	23.9	23.9
		200	26.7	24.8	29.4	**29.3**	42.5	21.5	101.1	30.3	29.8	30.0	29.9
Total classes 1, 4-8			6840	6837	7272	7258				7320	7286	7298	7302
Total classes 1–8			9242	9009	9806	9777				9863	9813		

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

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 6*r* represents the average best solution found when all the six rotations are allowed and the fitness function used is *aNB*; column with header *NB* represents the average best solution found when no rotations are allowed and fitness function used is *NB*; column with header *aNB* represents the average best solution found when no rotations are allowed and fitness function used is *aNB*, 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 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 *TS2P* and *GLS*.

The results show that our *BRKGA-BPP* algorithm consistently equals or outperforms algorithms *TS3*, *GVND*, and *GLS*. In relation

Table 9
Results for the two-dimensional instances. Part I.

Class	*Bin size*	*n*	LB_*	*BRKGA-BPP*						*GVND*	*SCH*	*GLS*	*TS3*	*HBP*
				2*r*	*NB*	*aNB*	aG	aT	T-200					
1	10 × 10	20	7.1	6.6	**7.1**	**7.1**	1.0	0.0	0.1	**7.1**	**7.1**	**7.1**	**7.1**	**7.1**
		40	13.4	12.8	**13.4**	**13.4**	4.3	0.0	0.6	**13.4**	**13.4**	**13.4**	13.5	**13.4**
		60	19.7	19.5	**20**	**20**	14.3	0.1	1.3	**20**	**20**	20.1	20.1	20.1
		80	27.4	27	**27.5**	**27.5**	1.9	0.0	3.3	**27.5**	**27.5**	**27.5**	28.2	**27.5**
		100	31.7	31.3	**31.7**	**31.7**	30.4	1.2	7.8	**31.7**	**31.7**	32.1	32.6	31.8
2	30 × 30	20	1.0	1.0	**1.0**	**1.0**	1.0	0.0	0.8	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		40	1.9	1.9	**1.9**	**1.9**	5.9	0.0	1.6	**1.9**	**1.9**	**1.9**	2.0	**1.9**
		60	2.5	2.5	**2.5**	**2.5**	3.4	0.0	2.3	**2.5**	**2.5**	**2.5**	2.7	**2.5**
		80	3.1	3.1	**3.1**	**3.1**	4.3	0.1	3.1	**3.1**	**3.1**	**3.1**	3.3	**3.1**
		100	3.9	3.9	**3.9**	**3.9**	13.4	0.3	5.1	**3.9**	**3.9**	**3.9**	4.0	**3.9**
3	40 × 40	20	5.1	4.7	**5.1**	**5.1**	1.9	0.0	0.4	**5.1**	**5.1**	**5.1**	5.5	**5.1**
		40	9.2	9.2	**9.4**	**9.4**	5.2	0.0	1.0	**9.4**	**9.4**	**9.4**	9.7	9.5
		60	13.6	13.4	**13.9**	**13.9**	11.3	0.1	2.0	**13.9**	**13.9**	14.0	14.0	14.0
		80	18.7	18.2	**18.9**	**18.9**	16.8	0.3	3.0	**18.9**	**18.9**	19.1	19.8	19.1
		100	22.1	22	**22.3**	**22.3**	17.6	0.4	4.6	**22.3**	**22.3**	22.6	23.6	22.6
4	100 × 100	20	1.0	1.0	**1.0**	**1.0**	1.0	0.0	0.5	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		40	1.9	1.9	**1.9**	**1.9**	1.0	0.0	1.2	**1.9**	**1.9**	**1.9**	**1.9**	**1.9**
		60	2.3	2.3	**2.5**	**2.5**	1.0	0.0	2.4	**2.5**	**2.5**	**2.5**	2.6	**2.5**
		80	3	3.1	**3.1**	**3.1**	13.3	0.2	3.5	**3.1**	3.2	3.3	3.3	3.3
		100	3.7	3.7	3.8	**3.7**	13.3	0.4	5.7	3.8	3.8	3.8	4.0	3.8
5	100 × 100	20	6.5	5.9	**6.5**	**6.5**	1.9	0.0	0.5	**6.5**	**6.5**	**6.5**	6.6	**6.5**
		40	11.9	11.4	**11.9**	**11.9**	3.7	0.0	1.0	**11.9**	**11.9**	**11.9**	**11.9**	**11.9**
		60	17.9	17.2	**18**	**18**	8.5	0.1	2.1	**18**	**18**	18.1	18.2	**18**
		80	24.1	23.9	**24.7**	**24.7**	7.0	0.1	3.2	**24.7**	**24.7**	24.9	25.1	24.8
		100	27.9	27.7	28.2	**28.1**	28.7	0.7	4.8	28.2	28.2	28.8	29.5	28.7
6	300 × 300	20	1.0	1.0	**1.0**	**1.0**	1.0	0.0	0.6	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
		40	1.5	1.6	1.7	**1.6**	16.7	0.1	1.3	1.7	1.7	1.8	1.9	1.8
		60	2.1	2.1	**2.1**	**2.1**	1.9	0.0	2.6	**2.1**	**2.1**	2.2	2.2	**2.1**
		80	3.0	3.0	**3.0**	**3.0**	1.0	0.0	3.7	**3.0**	**3.0**	**3.0**	**3.0**	**3.0**
		100	3.2	3.2	3.4	**3.3**	16.8	0.5	6.2	3.4	3.4	3.4	3.4	3.4
7	100 × 100	20	5.5	5.2	**5.5**	**5.5**	1.0	0.0	0.4	**5.5**	**5.5**	**5.5**	**5.5**	**5.5**
		40	10.9	10.2	**11.1**	**11.1**	10.1	0.1	1.0	**11.1**	**11.1**	11.3	11.4	**11.1**
		60	15.6	14.6	15.9	**15.8**	18.2	0.2	2.1	15.9	**15.8**	15.9	16.2	16.0
		80	22.4	20.8	**23.2**	**23.2**	2.8	0.0	3.5	**23.2**	**23.2**	**23.2**	**23.2**	**23.2**
		100	26.9	25	**27.1**	**27.1**	12.5	0.3	4.7	**27.1**	**27.1**	27.5	27.7	27.4
8	100 × 100	20	5.8	5.3	**5.8**	**5.8**	1.9	0.0	0.4	**5.8**	**5.8**	**5.8**	**5.8**	**5.8**
		40	11.2	10.3	**11.3**	**11.3**	5.2	0.0	1.0	**11.3**	**11.3**	11.4	11.4	**11.3**
		60	15.9	14.7	**16.1**	**16.1**	18.5	0.2	2.0	**16.1**	16.2	16.3	16.2	16.2
		80	22.3	20.4	**22.4**	**22.4**	11.6	0.2	3.3	**22.4**	**22.4**	22.5	22.6	22.6
		100	27.4	25.2	**27.8**	**27.8**	10.1	0.2	4.8	**27.8**	27.9	28.1	28.4	28.0
9	100 × 100	20	14.3	14.3	**14.3**	**14.3**	1.0	0.0	0.4	**14.3**	**14.3**	**14.3**	**14.3**	**14.3**
		40	27.8	27.5	**27.8**	**27.8**	1.0	0.0	1.0	**27.8**	**27.8**	**27.8**	**27.8**	**27.8**
		60	43.7	43.5	**43.7**	**43.7**	1.0	0.0	2.4	**43.7**	**43.7**	**43.7**	43.8	**43.7**
		80	57.7	57.3	**57.7**	**57.7**	1.0	0.0	4.0	**57.7**	**57.7**	**57.7**	**57.7**	**57.7**
		100	69.5	69.3	**69.5**	**69.5**	1.0	0.0	6.6	**69.5**	**69.5**	**69.5**	**69.5**	**69.5**
10	100 × 100	20	4.2	4.1	**4.2**	**4.2**	4.3	0.0	0.5	**4.2**	**4.2**	**4.2**	4.3	4.3
		40	7.4	7.2	**7.4**	**7.4**	1.9	0.0	1.1	**7.4**	**7.4**	**7.4**	7.5	**7.4**
		60	9.8	9.9	**10.0**	**10.0**	22.5	0.2	2.2	**10.0**	10.1	10.2	10.4	10.2
		80	12.3	12.5	12.9	**12.8**	12.6	0.2	3.4	12.9	**12.8**	13.0	13.0	13.0
		100	15.3	15.4	15.9	**15.8**	23.0	0.5	4.7	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**.

Table 10
Results for the two-dimensional instances. Part II.

Class	*n*	Inst	*LB**	*BRKGA-BPP*						*GVND*	*SCH*	*GLS*	*TS*3
				2*r*	*NB*	*aNB*	aG	aT	T-200				
cgcut	16–62	3	9	7.67	9	**9**	1.0	0.01	1.84	**9**	**9**	**9**	**9**
gcut	10–50	13	8	7.31	8	**8**	2.0	0.01	0.69	**8**	**8**	**8**	8.31
ngcut	7–22	12	2.67	2.5	2.67	**2.67**	1.0	0.00	0.96	**2.67**	**2.67**	**2.67**	3
beng 1–8	20–120	8	6.75	6.75	6.75	**6.75**	14.1	0.09	3.29	**6.75**	6.88		
beng 9–10	160–200	2	6.5	6.5	6.5	**6.5**	1.0	0.13	26.99	**6.5**			

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

Table 11
Evolution of the total number of bins, *aNB*, and the percentage of best solutions found, *%Best*, as the number of generations increases.

Datasets		**Number of Generations**									
		1	**10**	**25**	**50**	**75**	**100**	**125**	**150**	**175**	**200**
3*D*	***a*ND**	10066	9959	9862	9806	9798	9791	9786	9781	9778	9777
	%Best	34%	50%	74%	91%	93%	96%	97%	99%	100%	100%
2*D*−1	***a*ND**	7359	7299	7261	7248	7245	7241	7234	7234	7234	7234
	%Best	75%	87%	95%	97%	98%	98%	100%	100%	100%	100%
2*D*−2	***a*ND**	236	232	231	231	231	230	230	230	230	230
	%Best	84%	95%	97%	97%	97%	100%	100%	100%	100%	100%

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 $BRKGA{-}BPP < TS3$, $BRKGA{-}BPP < GVND$, $BRKGA{-}BPP < TS^2PACK$ 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 2*r* 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*, *TS*3, 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*, *TS*3, 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.

Table 11 presents the evolution of the total number of bins, *aNB*, and the percentage of best solutions found, *%Best*, as the number of generations increases for all the three datasets. From Table 11 it is clear that *BRKGA-BPP* would still obtain better results than the other approaches even if only 50 generations were used for the 3D datasets and if only 125 generations were used for the 2D datasets.

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 to the best solution and for 200 generations.

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 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 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, *DTFTRC*-1 and *DTFTRC*-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

Bean, J., 1994. Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 6, 154–160.
Beasley, J., 1985a. Algorithms for unconstrained two-dimensional guillotine cutting. Journal of the Operational Research Society 36, 297–306.
Beasley, J., 1985b. An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research 33, 49–64.
Bengtsson, B., 1982. Packing rectangular pieces – a heuristic approach. The Computer Journal 25, 353–357.
Berkey, J., Wang, P., 1987. Two-dimensional finite bin-packing algorithms. Journal of the Operational Research Society 38, 423–429.
Boschetti, M., 2004. New lower bounds for the three-dimensional finite bin packing problem. Discrete Applied Mathematics 140, 241–258.
Boschetti, M., Mingozzi, A., 2003a. The two-dimensional finite bin packing problem. Part I. New lower bounds for the oriented case. 4OR: A Quarterly Journal of Operations Research 1, 27–42.
Boschetti, M., Mingozzi, A., 2003b. The two-dimensional finite bin packing problem. Part II. New lower and upper bounds. 4OR: A Quarterly Journal of Operations Research 1, 135–147.
Christofides, N., Whitlock, C., 1977. An algorithm for two-dimensional cutting problems. Operations Research 25, 30–44.
Crainic, T., Perboli, G., Tadei, R., 2009. TS^{2}PACK: a two-level tabu search for the three-dimensional bin packing problem. European Journal of Operational Research 195, 744–760.
den Boef, E., Korst, J., Martello, S., Pisinger, D., Vigo, D., 2005. Erratum to "The three-dimensional bin packing problem": robot-packable and orthogonal variants of packing problems. Operations Research 53, 735–736.
Faroe, O., Pisinger, D., Zachariasen, M., 2003. Guided local search for the three-dimensional bin-packing problem. INFORMS Journal on Computing 15, 267–283.
Fekete, S., Schepers, J., 1997. A new exact algorithm for general orthogonal *d*-dimensional knapsack problems. In: Algorithms–ESA'97, Springer, pp. 144–156.
Fekete, S., Schepers, J., 2004. A combinatorial characterization of higher-dimensional orthogonal packing. Mathematics of Operations Research 29, 353–368.
Fekete, S., Schepers, J., VanderVeen, J., 2007. An exact algorithm for higher-dimensional orthogonal packing. Operations Research 55, 569–587.
Goncalves, J.F., Sousa, P.S.A., 2011. A genetic algorithm for lot sizing and scheduling under capacity constraints and allowing backorders. International Journal of Production Research 49, 2683–2703.
Gonçalves, J.F., Almeida, J.R., 2002. A hybrid genetic algorithm for assembly line balancing. Journal of Heuristics 8, 629–642.
Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C., 2005. A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167, 77–95.
Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C., 2009. A genetic algorithm for the resource constrained multi-project scheduling problem. European Journal of Operational Research 189, 1171–1190.
Gonçalves, J.F., Resende, M.G.C., 2004. An evolutionary algorithm for manufacturing cell formation. Computers and Industrial Engineering 47, 247–273.
Gonçalves, J.F., Resende, M.G.C., 2011. Biased random-key genetic algorithms for combinatorial optimization. Journal of Heuristics 17, 487–525.
Gonçalves, J.F., Resende, M.G.C., 2012. A parallel multi-population biased random-key genetic algorithm for a container loading problem. Computers and Operations Research 39, 179–190.
Gonçalves, J.F., Resende, M.G.C., Mendes, J.J.M., 2011. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. Journal of Heuristics 17, 467–486.
Hartmann, S., 2000. Packing problems and project scheduling models: an integrating perspective. Journal of the Operational Research Society, 1083–1092.
Johnson, D., 1973. Near-Optimal Bin Packing Algorithms. Ph.D. Thesis, Massachusetts Institute of Technology.
Lai, K., Chan, J., 1997. Developing a simulated annealing algorithm for the cutting stock problem. Computers and Industrial Engineering 32, 115–127.
Liu, D., Teng, H., 1999. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research 112, 413–420.
Lodi, A., Martello, S., Vigo, D., 1999. Approximation algorithms for the oriented two-dimensional bin packing problem. European Journal of Operational Research 112, 158–166.
Lodi, A., Martello, S., Vigo, D., 2002. Heuristic algorithms for the three-dimensional bin packing problem. European Journal of Operational Research 141, 410–420.
Lodi, A., Martello, S., Vigo, D., 2004. TSpack: a unified tabu search code for multi-dimensional bin packing problems. Annals of Operations Research 131, 203–213.
Mack, D., Bortfeldt, A., 2012. A heuristic for solving large bin packing problems in two and three dimensions. Central European Journal of Operations Research 20 (2), 337–354.
Martello, S., Pisinger, D., Vigo, D., 2000. The three-dimensional bin packing problem. Operations Research 48, 256–267.
Martello, S., Pisinger, D., Vigo, D., Boef, E., Korst, J., 2007. Algorithm 864: general and robot-packable variants of the three-dimensional bin packing problem. ACM Transactions on Mathematical Software 33, 7:1–7:12.
Martello, S., Vigo, D., 1998. Exact solution of the two-dimensional finite bin packing problem. Management Science 44, 388–399.
Monaci, M., Toth, P., 2006. A set-covering-based heuristic approach for bin-packing problems. INFORMS Journal on Computing 18, 71–85.
Park, K., Lee, K., Park, S., Kim, S., 1996. Modeling and solving the spatial block scheduling problem in a shipbuilding company. Computers and Industrial Engineering 30, 357–364.
Parreño, F., Alvarez-Valdes, R., Oliveira, J., Tamarit, J., 2010. A hybrid GRASP/VND algorithm for two-and three-dimensional bin packing. Annals of Operations Research 179, 203–220.
Spears, W., Dejong, K., 1991. On the virtues of parameterized uniform crossover. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 230–236.
Wäscher, G., Haussner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 183, 1109–1130.