

A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem

José Fernando Gonçalves · Mauricio G. C. Resende

Published online: 8 April 2010
© Springer Science+Business Media, LLC 2010

Abstract This paper addresses a constrained two-dimensional (2D), non-guillotine restricted, packing problem, where a fixed set of small rectangles has to be placed into a larger stock rectangle so as to maximize the value of the rectangles packed. The algorithm we propose hybridizes a novel placement procedure with a genetic algorithm based on random keys. We propose also a new fitness function to drive the optimization. The approach is tested on a set of instances taken from the literature and compared with other approaches. The experimental results validate the quality of the solutions and the effectiveness of the proposed algorithm.

Keywords Packing · Cutting · Two-dimensional packing · Two-dimensional cutting · Non-guillotine cutting · Genetic algorithm

1 Introduction

The constrained two-dimensional (2D), non-guillotine restricted, packing problem addressed in this paper consists of packing rectangular pieces into a large rectangular sheet of material (stock rectangle) in order to maximize the value of the rectangles packed. We consider the special case in which the items cannot be rotated and must be packed with their edges always parallel to the edges of the large rectangular sheet. The problem is relevant both from a theoretical point of view as well as practical one. It is

Supported by Fundação para a Ciência e Tecnologia (FCT) project PTDC/GES/72244/2006.

J.F. Gonçalves

LIAAD, Faculdade de Economia do Porto, Rua Dr. Roberto Frias, s/n, 4200-464 Porto, Portugal
e-mail: jfgoncal@fep.up.pt

M.G.C. Resende (✉)

Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue,
Room C241, Florham Park, NJ 07932, USA
e-mail: mgcr@research.att.com

NP-hard (Garey and Johnson 1979) and arises in various production processes with applications varying from the home-textile to the glass, steel, wood, and paper industries, where rectangular figures are cut from large rectangular sheets of materials.

Various types of two-dimensional cutting problems have been considered in the literature. Dyckhoff (1990) provided a classification of the various types of cutting problems. Surveys for multidimensional cutting and packing problems are given by Dowsland and Dowsland (1992), Haessler and Sweeney (1991), and Sweeney and Paternoster (1992). Few authors have considered general two-dimensional non-guillotine cutting problems (constrained or unconstrained). The unconstrained non-guillotine cutting problem has been considered by a few authors, namely: Tsai et al. (1988) presented an integer programming approach; Arenales and Morabito (1995) developed an approach based on an AND/OR graph together with a branch and bound search; and Healy et al. (1999) introduced an algorithm based on the identification of the empty space that can be used to cut new items.

Optimal procedures for the constrained two-dimensional non-guillotine cutting problem have been proposed by several authors. Beasley (1985) proposed a branch and bound algorithm with an upper bound derived from a Lagrangian relaxation of a 0-1 integer linear programming formulation. Scheithauer and Terno (1993) presented an integer programming formulation in which binary variables are used to indicate whether a piece item is cut above/below or to the right/left of another piece. Hadjiconstantinou and Christofides (1995) developed a branch and bound algorithm in which the search was limited by using an upper bound based on a Lagrangian relaxation procedure and improved it using subgradient optimization. Computational gains were achieved by applying new reduction tests. Fekete and Schepers (1997a, 1997b, 1997c, 2004a, 2004b) developed a two-level tree search algorithm for solving the d -dimensional knapsack problem. In this algorithm, projections of cut items were made onto both the horizontal and vertical edges of the stock rectangle. These projections were represented by graphs in which the nodes are the cut items and an edge joins two nodes if the projections of the corresponding cut items overlap. By looking at the properties of the graphs, the authors were able to check the feasibility of the corresponding patterns. Amaral and Letchford (2001) presented an upper bound which involved the solution of a large linear program by a column generation algorithm. Boschetti et al. (2002) proposed new upper bounds derived from different relaxations of a new integer programming formulation of the constrained two-dimensional non-guillotine cutting problem. The new formulation is based on the observation that any feasible solution can be represented by two sequences in which each element is the subset of items covering the x and y positions of the master surface, respectively. Caprara and Monaci (2004) compared four new algorithms based on the natural relaxation of the two-dimensional knapsack problem. In the relaxation, the knapsack has a capacity equal to the area of the master surface and the item weights are equal to their areas.

Heuristic procedures for the constrained two-dimensional non-guillotine cutting problem have also been developed by several authors. Lai and Chan (1997b) presented a heuristic based on simulated annealing. They used a problem representation which encodes the order in which pieces should be cut. Lai and Chan (1997a) proposed a heuristic based on an evolutionary strategy approach. They used the same representation as in Lai and Chan (1997b). Their algorithm includes an improvement

procedure based on dividing the ordered list into active (cut) and inactive (uncut) pieces and seeing if any (currently) uncut pieces can be cut. Their mutation process involves swapping two pieces in the ordered list. Leung et al. (2001) used the ordered representation of Lai and Chan (1997b) and discussed producing a cutting pattern from it using both the difference process algorithm of Lai and Chan (1997b) and a standard bottom-left algorithm (see Jakobs 1996). They showed that there exist problems for which the optimal solution cannot be found by these two approaches. They presented a simulated annealing heuristic with a move corresponding to either swapping two pieces in the ordered representation or moving a single piece to a new position in the representation. They also presented a genetic algorithm involving five different crossover operators. In an attempt to alleviate the problem of premature convergence, Leung et al. (2003) developed a hybrid heuristic blending simulated annealing with a genetic algorithm. Beasley (2004) proposes a genetic algorithm based on a nonlinear formulation of the problem, where variables indicate if a piece is cut or not and its position on the stock sheet. No placement algorithm is needed since the solutions are lists of variables and show directly the cutting pattern. Alvarez-Valdes et al. (2005) developed a new heuristic based on GRASP (Feo and Resende 1989, 1995) for the non-guillotine two-dimensional cutting stock problem. The constructive phase explicitly considers the possibility of simultaneously cutting several pieces of the same type and forming a block, as one would do in the pallet loading problem. Alvarez-Valdes et al. (2007) propose a tabu search algorithm that define several moves based on reducing and inserting blocks of pieces and include intensification and diversification procedures based on long-term memory.

In this paper, we present a hybrid heuristic for a two-dimensional non-guillotine cutting problem which combines a random-keys based genetic algorithm with a novel fitness function and a new heuristic placement policy. The remainder of the paper is organized as follows. In Sect. 2, we define the problem formally, and in Sect. 3, propose the new approach, describing the genetic algorithm, the placement strategy, and the fitness function. In Sect. 4, we present experimental results and in Sect. 5 make concluding remarks.

2 The problem

The two-dimensional packing problem addressed in this paper is the problem of packing into a single large planar stock rectangle (W, H) , of width W and height H , smaller rectangles types (w_i, h_i) , $i = 1, \dots, n$, each of width w_i and height h_i . Each rectangle type i has a fixed orientation (i.e. cannot be rotated); must be packed with its edges parallel to the edges of the stock rectangle; and the number x_i of pieces of each rectangle type that are to be packed must lie between P_i and Q_i , i.e. $0 \leq P_i \leq x_i \leq Q_i$, for all $i = 1, \dots, n$. Each rectangle type $i = 1, \dots, n$ has an associated value equal to v_i and the objective is to maximize the total value of the rectangles cut $\sum_{i=1}^n v_i x_i$. Without significant loss of generality, it is usual to assume that all dimensions W, H , and (w_i, h_i) , $i = 1, \dots, n$, are integers. To simplify notation, we shall use $N = \sum_{i=1}^n Q_i$ as the maximum number of rectangles that can be packed. According to the newly developed typology by Wäscher et al. (2007), the problem falls into the *output maximization assignment* kind and can be classified

as a 2D-SKP problem type (two-dimensional single knapsack problem) given that it handles instances where a strong heterogeneous assortment of small items exist.

Depending on the values of P_i and Q_i , we can distinguish the following three types of problems:

1. In the *unconstrained* type, we have $P_i = 0$, $Q_i = \lfloor (W \times H)/(w_i \times h_i) \rfloor$, for $i = 1, \dots, n$, which is a trivial bound;
2. In the *constrained* type, we have: for all $i = 1, \dots, n$, $P_i = 0$ and $\exists j \in \{1, \dots, n\}$ such that $Q_j < \lfloor (W \times H)/(w_j \times h_j) \rfloor$;
3. In the *doubly constrained* type, we have: $\exists i \in \{1, \dots, n\}$ such that $P_i > 0$ and $\exists j \in \{1, \dots, n\}$ such that $Q_j < \lfloor (W \times H)/(w_j \times h_j) \rfloor$.

A simple upper bound for the problem can be obtained by solving the following bounded knapsack problem, where variable x_i represents the number of pieces of type i to be cut in excess of its lower bound P_i :

$$\begin{aligned} & \max \sum_{i=1}^n v_i x_i + \sum_{i=1}^n v_i P_i \\ & \text{subject to} \quad \sum_{i=1}^n (w_i \cdot h_i) x_i \leq W \cdot H - \sum_{i=1}^n (w_i \cdot h_i) P_i \\ & \quad x_i \leq Q_i - P_i, \quad i = 1, \dots, n, \\ & \quad x_i \geq 0, \quad \text{integer}, \quad i = 1, \dots, n. \end{aligned}$$

This simple upper bound will be used to evaluate the performance of the hybrid genetic algorithm described in this paper.

3 New approach

3.1 Overview about the new approach

The new approach proposed in this paper combines a random-keys based multi population genetic algorithm, a new placement strategy, and a novel measure of solution quality that we call *modified total value*. Instead of describing the algorithms directly as cutting problems, we use the equivalent notion of packing, where we are given rectangles and wish to pack them on the stock rectangle.

The role of the genetic algorithm is to evolve the encoded solutions, or *chromosomes*, which represent the rectangle packing sequence and the type of placement procedure used to place each rectangle. For each chromosome, the following four phases are applied:

1. *Decoding of the rectangle type packing sequence.* The first phase is responsible for transforming part of the chromosome supplied by the genetic algorithm into the sequence in which the rectangles are to be packed in the stock rectangle.
2. *Decoding of the placement procedure.* In the second phase, part of the chromosome supplied by the genetic algorithm is transformed into a vector that indicates

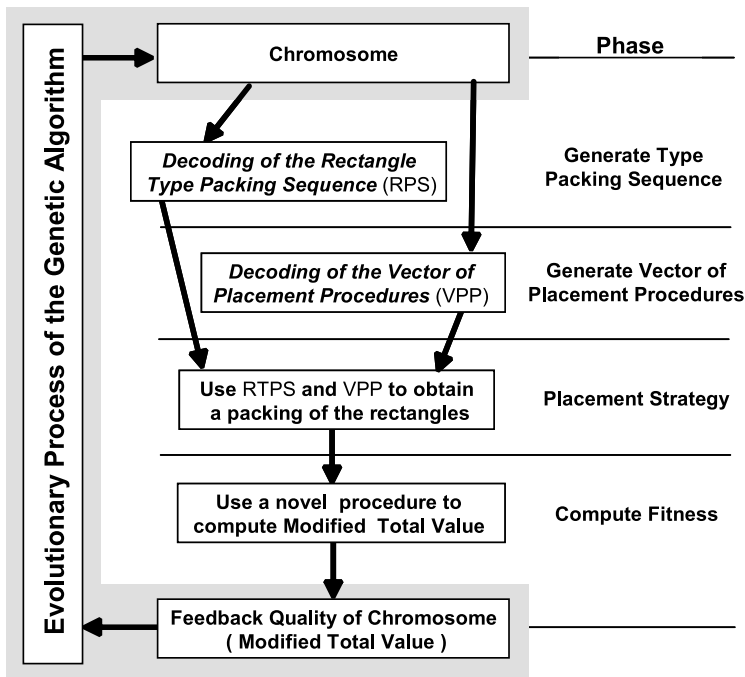


Fig. 1 Architecture of the new approach

which placement procedure is to be used to position each rectangle in the stock rectangle.

3. *Placement strategy.* The third phase makes use of the rectangle packing sequence defined in the first phase and the vector of placement procedures defined in second phase and constructs a packing of the rectangles.
4. *Fitness evaluation.* In the final phase, we make use of a novel procedure to compute a modified total value which is used as a fitness measure (quality measure) to feedback to the genetic algorithm.

Figure 1 illustrates the sequence of steps applied to each chromosome generated by the genetic algorithm.

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

3.2 Genetic algorithm

The following sub sections present the chromosome representation and decoding, the evolutionary process, the initialization process and the multi-population strategy.

3.2.1 Chromosome representation and decoding

The genetic algorithm described in this paper uses a random-keys alphabet comprised of random real numbers between 0 and 1. The evolutionary strategy used is similar to

the one proposed by Bean (1994), the main difference occurring in the crossover operator. The important feature of random keys is that all offspring formed by crossover are feasible solutions. This is accomplished by moving much of the feasibility issue into the objective function evaluation. If any random-key vector can be interpreted as a feasible solution, then any crossover vector is also feasible. Through the dynamics of the genetic algorithm, the system learns the relationship between random-key vectors and solutions with good objective function values.

A chromosome represents a solution to the problem and is encoded 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 derive from it a solution called a *phenotype*. In the present context, the direct use of cutting patterns as chromosomes is too complicated to represent and manipulate. In particular, it is difficult to develop corresponding crossover and mutation operations. Instead, solutions are represented indirectly by parameters that are later used by a decoding procedure to obtain a solution. To obtain the solution (phenotype) we use the placement strategy to be described in Sect. 3.3.

Recall that there are n rectangle types and that at most Q_i rectangles of type i can be packed into the stock rectangle. In the description of the genetic algorithm, we take a total of $N = \sum_{i=1}^n Q_i$ rectangles, i.e. Q_i rectangles of type $i = 1, \dots, n$.

Each solution chromosome is made of $2N$ genes, where N is the number of rectangles to be packed, i.e.

$$\text{Chromosome} = (\underbrace{gene_1, \dots, gene_N}_{\text{Rectangle Type Packing Sequence}} , \underbrace{gene_{N+1}, \dots, gene_{2N}}_{\text{Vector of Placement Procedures}}).$$

The first N genes are used to obtain the *Rectangle Type Packing Sequence* (RTPS) while the last N genes are used to obtain the *Vector of Placement Procedures* (VPP).

Both the RTPS and VPP are used by the placement strategy. The decoding (mapping) of the first N genes of each chromosome into an RTPS is accomplished by sorting the genes and rectangles types in ascending order. Figure 2 presents an example of the decoding process for the RTPS. In the example there are four types of

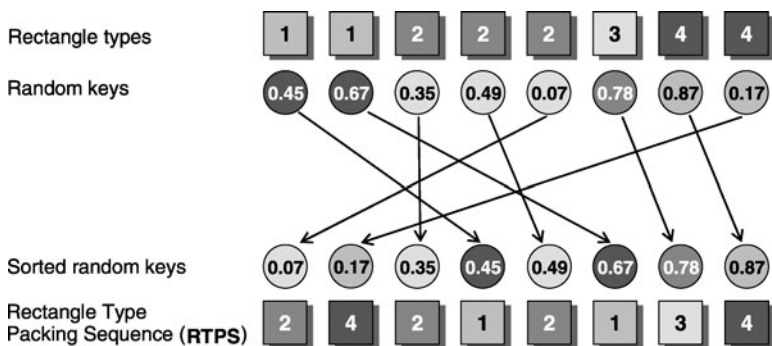


Fig. 2 Chromosome decoding procedure

rectangles and the values of Q_i are $Q_1=2$, $Q_2=3$, $Q_3=1$ and $Q_4=2$. According to the ordering obtained the rectangles types should be packed in the following order 2, 4, 2, 1, 2, 1, 3, 4.

In the placement strategy we make use of two placement procedures; *BL* (bottom-left) and *LB* (Left-Bottom) (see Sect. 3.3). The decoding (mapping) of the last N genes of each chromosome into a VPP is accomplished using, for $i = 1, \dots, N$, the following expression:

$$VPP(i) = \begin{cases} BL & \text{if } gene(N+i) \leq \frac{1}{2}, \\ LB & \text{if } \frac{1}{2} < gene(N+i). \end{cases}$$

Note that with this representation we are able to handle the upper bounds $x_i \leq Q_i$, $i = 1, \dots, n$, implicitly. Later in Sect. 3.4 we discuss how we deal with the lower bounds $x_i \geq P_i$, for one or more $i \in \{1, \dots, n\}$, in the case of doubly constrained problems.

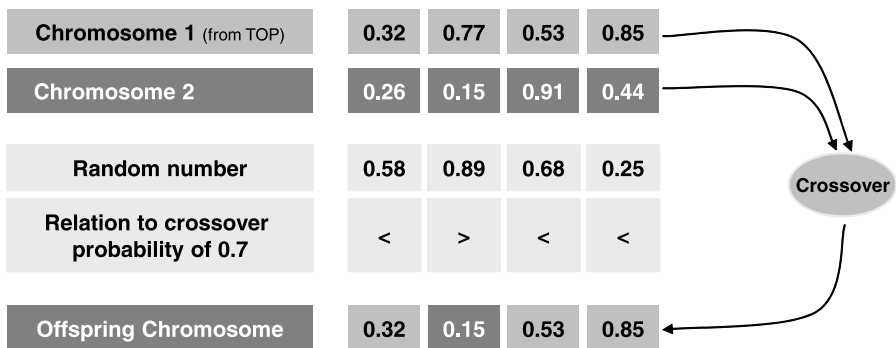
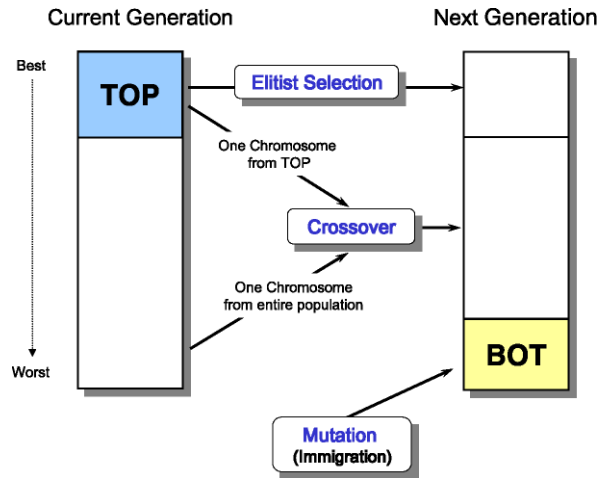
3.2.2 Evolutionary process

To breed good solutions, the random-key vector population is operated upon by a genetic algorithm. There are many variations of genetic algorithms obtained by altering the reproduction, crossover, and mutation operators. The reproduction and crossover operators determine which parents will have offspring, and how genetic material is exchanged between the parents to create those offspring. Mutation allows for random alteration of genetic material. Reproduction and crossover operators tend to increase the quality of the populations and force convergence. Mutation opposes convergence and replaces genetic material lost during reproduction and crossover.

The population is initialized with random-key vectors whose components are random real numbers uniformly sampled from the interval $[0, 1]$. *Reproduction* is accomplished by first copying some of the best individuals from one generation to the next, in what is called an *elitist strategy* (Goldberg 1989). The advantage of an elitist strategy over traditional probabilistic reproduction is that the best solution is monotonically improving from one generation to the next. The potential downside is population convergence to a local minimum. This can, however, be overcome by an appropriate amount of mutation as described below.

Parametrized uniform crossovers (Spears and Dejong 1991) are employed in place of the traditional one-point or two-point crossover. After two parents are chosen, one chosen randomly from the best (unlike Bean 1994 we always choose one parent from the best, Gonçalves and Resende 2009 show that this change produces results with better quality and faster) and the other chosen randomly from the full old population (including chromosomes copied to the next generation in the elitist selection), at each gene we toss a biased coin to select which parent will contribute the allele, see *TOP* in Fig. 3. Figure 4 presents an example of the crossover operator. It assumes that a coin toss of heads selects the gene from the first parent, a tails chooses the gene from the second parent, and that the probability of tossing a heads, crossover probability $CProb = 0.7$. In Sect. 4 we describe how we determine this value empirically.

Rather than using the traditional gene-by-gene mutation with very small probability at each generation, some new members of the population are randomly generated

Fig. 3 Transitional process between consecutive generations**Fig. 4** Example of parametrized uniform crossover with crossover probability equal to 0.7

from the same distribution as the initial population (see *BOT* in Fig. 3). The purpose of this process is to prevent premature convergence of the population, like in a mutation operator, and leads to a simple statement of convergence. Figure 3 depicts the transitional process between two consecutive generations.

3.2.3 Population Initialization

In our approach the initial population is not fully randomly generated. We introduce in the initial population four non-random chromosomes. These non-random chromosomes are created so that their corresponding RTPSs are equivalent to packing the rectangles in descending order of their value, i.e., rectangles with higher value appear first in the packing sequence. The four chromosomes result from considering four variations for the placement procedures: random, all BL, all LB and alternatively BL and LB. As will be shown in the numerical experiments the introduction of the four non-random chromosomes improves significantly the quality of the solutions obtained.

3.2.4 Multi-population strategy

In our multi-population strategy several populations are evolved independently. After a pre-determined number of generations all the populations exchange information. The information exchanged is the chromosomes of good quality. When evaluating possible interchange strategies we noticed that exchanging too much information (exchanging too many chromosomes) leads to the disruption of the evolutionary process. Also, if the populations exchange information with a high frequency they do not have enough time to produce good results because their evolutionary process is disrupted before good solutions can be achieved. Having this information in mind we chose a multi-population strategy that after a pre-determined number of generations (this will be determined empirically in Sect. 4) selects the two best chromosomes from all current chromosomes and then copies them to the populations where they don't currently exist and removes the corresponding number of the worst chromosomes.

3.3 Placement strategy

Rectangles types are placed in the stock rectangle, one at a time, in the order defined by the RTPS supplied by the genetic algorithm (see Sect. 3.2.2). While trying to place a rectangle type in the stock rectangle we consider only maximal *empty rectangular spaces* (ERSs), i.e. ERSs that are not contained in any other ERS. A rectangle type is placed in an ERS where it fits and according to the placement procedure defined for its placement by the genetic algorithm.

Initially we considered only a *Bottom-Left* (BL) placement procedure which first orders the ERSs in such way that $ERS_i < ERS_j$ if $y_i < y_j$ or $y_i = y_j, x_i < x_j$ and then chooses the first ERS in which the rectangle type to be packed fits (note that x_i/y_i is the coordinate x/y of the bottom left corner of the ERS), see pseudo-code in Fig. 5. However, as observed by Liu and Teng (1999), we noticed that some optimal solutions could not be constructed by the BL placement procedure. In other words, given an optimal solution to a problem, it is possible that no RTPS exists that, combined with the BL placement procedure, produces the given optimal solution. Figure 6 shows an optimal solution for one problem where the BL placement procedure cannot find the optimal solution.

In Fig. 7, we present, for the problem in Fig. 6, all the solutions obtained by the BL placement procedure for all the possible RTPSs starting with rectangle 2 (Similar sub-optimal solutions are produced when rectangles 1, 3, and 4 are fixed in the first position). As can be observed, none of those RTPSs, when combined with BL placement procedure, produces the optimal solution in Fig. 6.

To overcome this weakness, we combine the BL placement procedure with a *Left-Bottom* (LB) placement procedure which first orders the ERSs in such way that $ERS_i < ERS_j$ if $x_i < x_j$ or $x_i = x_j, y_i < y_j$ and then chooses the first ERS in which the rectangle type to be packed fits (note that x_i/y_i is the coordinate x/y of the bottom left corner of the ERS). In summary, our placement strategy uses two placement procedures, the Bottom-Left and the Left-Bottom, to construct a packing of the rectangle types. The vector of placement procedures (VPP), supplied by the genetic

```

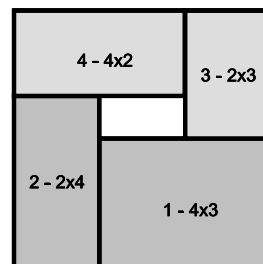
procedure BL( $r_i$ )
1  Let  $r_i$  be a rectangle of type  $i$  to be packed in the stock rectangle;
2  Let  $N_{ERS}$  be the number of available ERSs;
3  Initialize  $L \leftarrow W$ ;  $B \leftarrow H$ ;  $ERS_i^* \leftarrow 0$ ;
4  for  $k = 1, \dots, N_{ERS}$  do
5      Let  $x(ERS_k)$  be the  $x$  coordinate of the bottom left corner of  $ERS_k$ ;
6      Let  $y(ERS_k)$  be the  $y$  coordinate of the bottom left corner of  $ERS_k$ ;
7      if  $r_i$  fits in  $ERS_k$  then
8          if  $x(ERS_k) \leq B$  or ( $x(ERS_k) = B$  and  $y(ERS_k) \leq L$ ) then
9               $L \leftarrow x(ERS_k)$ ;  $B \leftarrow y(ERS_k)$ ;
10              $ERS_i^* = ERS_k$ 
11         end if
12     end if
13 end for
14 Return  $ERS_i^*$ ;
end BL;

```

Fig. 5 Pseudo-code of the Bottom-Left (BL) placement procedure

Fig. 6 Optimal solution found by combining the Bottom-Left (BL) and the Left-Bottom (LB) placement procedures

$$\frac{RTPS = 2 \quad - \quad 1 \quad - \quad 4 \quad - \quad 3}{VPP = BL \quad - \quad BL \quad - \quad LB \quad - \quad BL}$$



algorithm, indicates, for each rectangle type to be packed, whether it should be placed using the BL or LB procedure. In Fig. 6 we present the optimal solution found using $RTPS = (2, 1, 4, 3)$ and $VPP = (BL, BL, LB, BL)$.

When placing a rectangle type in a ERS we try to build a layer containing several rectangles of that rectangle type. We use two types of layers: horizontal and vertical. When we use the BL placement procedure we try to create a horizontal layer and when we use the LB placement procedure we try to create a vertical layer (see Fig. 8).

Let i be the type of j^{th} , $j = 1, \dots, N$, rectangle to be placed as defined by the RTPS. Furthermore, let VPP_j be the placement procedure that the genetic algorithm assigned for the placement of j^{th} rectangle type. The pseudo-code of the placement

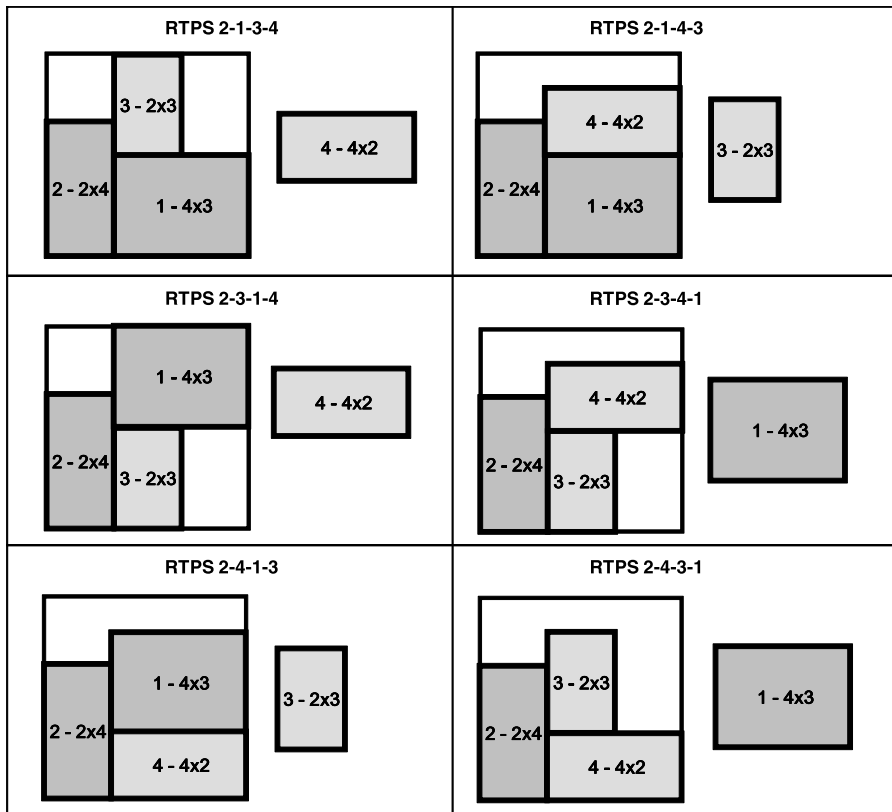


Fig. 7 Solutions found by the Bottom-Left (BL) placement procedure for the problem in Fig. 6 for all Rectangle Packing Sequences with rectangle 2 in the first position

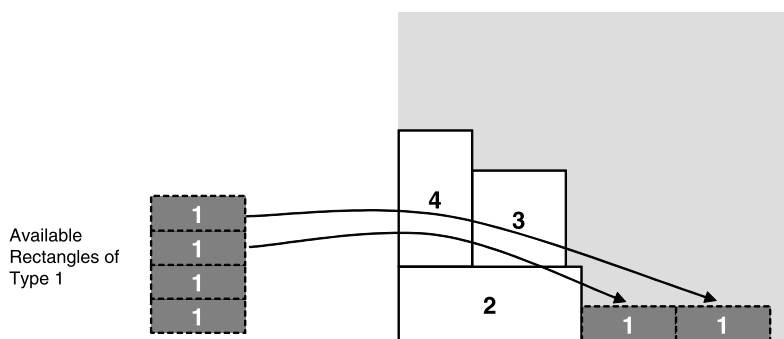
procedure is presented in Fig. 9. To generate and keep track of the ERSs, we make use of the *Difference Process* (DP), developed by Lai and Chan (1997b).

3.4 Modified total value fitness function

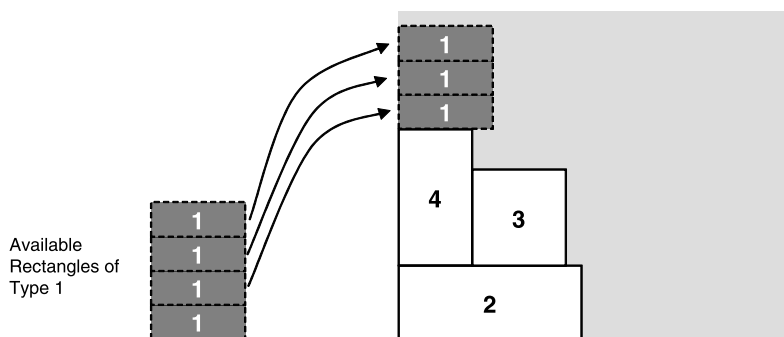
A natural fitness function (measure of quality) for this type of problem is the *total value* given by:

$$\text{Total value} = \sum_{i=1}^n v_i x_i,$$

where x_i is the number of rectangular pieces of type i to be cut in a solution and v_i is the value of each rectangular piece of type i . This measure, however, is not ideal because it does not capture well the potential for improvement of a solution. To be able to capture the improvement potential of different packings which have the same total value, we use a fitness measure that we call *modified total value* which is an adaptation of the fitness measure proposed by Gonçalves (2007) and is given by the



a) Packing a horizontal layer of type 1 rectangles using the **Bottom-Left** procedure



b) Packing a vertical layer of type 1 rectangles using the **Left-Bottom** procedure

Fig. 8 Packing of a layer using (a) the Bottom-Left procedure and (b) the Left-Bottom procedure

following expression:

Modified Total Value

$$= \text{Total Value} + K \times \frac{\text{Minimum Value Rectangle} \times \text{Area of Largest ERS}}{\text{Area of Stock Rectangle}}.$$

In the form presented above, our algorithm does not handle doubly constrained problems since lower bounds are not enforced. To enable the algorithm to handle these problems, we made one further modification to the fitness function. The lower bounds P_i for each rectangle of type i are treated indirectly through the use of a penalty term in the fitness function. If any lower bound P_i is not satisfied in a solution, then a penalty parameter PF is subtracted from the modified total value. All the computational tests were performed using $PF = 10^{10}$.

procedure PLACEMENT

```

1  Let  $R_i$  be the remaining quantity of unpacked rectangles of type  $i$ ;
2  Initialize  $R_i \leftarrow Q_i$   $i = 1, \dots, n$ ;
3  for  $j = 1, \dots, N$  do
4      Let  $i = RTPS(j)$ , be the type of  $j^{\text{th}}$  rectangle to be packed
      · according to the RTPS;
5      if  $R_i > 0$  then // check if there are any rectangles of type  $i$  remaining
      · to pack;
6           $ERS_i^* \leftarrow \emptyset$ 
7           $nL \leftarrow \emptyset$ 
8          if  $VPP_m = BL$  then
9              Let  $ERS_i^*$  be the ERS, in the list of available ERSs,
              · in which a rectangle of type  $i$  is placed when the Bottom-Left
              · placement heuristic is applied;
10             else if  $VPP_m = LB$  then
11                 Let  $ERS_i^*$  be the ERS, in the list of available ERSs,
                  · in which a rectangle of type  $i$  is placed when the Left-Bottom
                  · placement heuristic is applied;
12             end if
13             if  $ERS_i^* > \emptyset$  then // an ERS was found;
14                 if  $VPP_m = BL$  then
15                     Let  $nL$  = number of rectangles of type  $i$ 
                     · that can be placed in a horizontal layer in  $ERS_i^*$ 
                     · (see Fig. 8a);
16                     Place a horizontal layer with  $nL$  rectangles of type  $i$ 
                     · at the bottom left corner of  $ERS_i^*$ ;
17                 else if  $VPP_m = LB$  then
18                     Let  $nL$  = number of rectangles of type  $i$ 
                     · that can be placed in a vertical layer in  $ERS_i^*$ 
                     · (see Fig. 8b);
19                     Place a vertical layer with  $nL$  rectangles of type  $i$ 
                     · at the bottom left corner of  $ERS_i^*$ ;
20                 end if
21                 Update the list of available ERSs using the DP process
                  · of Lai and Chan (1997b);
22             end if
23              $R_i \leftarrow R_i - nL$ 
24         end if
25     end for
end PLACEMENT;

```

Fig. 9 Pseudo-code of the placement procedure of the hybrid heuristic

3.5 Parallel implementation

In our parallel implementation we parallelized only the task that performs the evaluation of the chromosomes fitness since it is the most time consuming. The tasks related with the GA logic were not parallelized since they consume very little time. 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 version of our approach was implemented using the OpenMP Application Program Interface (API) which supports multi-platform shared-memory parallel programming in C/C++.

4 Numerical experiments

In this section we report the results obtained on a set of experiments conducted to evaluate the performance of the Multi-Population Genetic Algorithm (MPGA) proposed in this paper.

4.1 Benchmark algorithms

We compare the MPGA with the following four recently proposed heuristics, which present the best computational results to date:

- *PH*—a population heuristic, proposed by Beasley (2004), where a population of solutions to the problem is progressively evolved.
- *GA*—proposed by Hadjiconstantinou and Iori (2007), this genetic algorithm uses an elitist theory, immigration, online heuristics, and tailored crossover operators.
- *GRASP*—a greedy randomized adaptive search procedure proposed by Alvarez-Valdes et al. (2005).
- *TABU*—a tabu search approach proposed by Alvarez-Valdes et al. (2007).

4.2 Test problem instances

The effectiveness of MPGA is evaluated by solving the same four sets of test problem instances used by Alvarez-Valdes et al. (2007). These problem instances sets are:

1. A set of 21 problem instances taken from the literature: 15 from Beasley (1985), two from Hadjiconstantinou and Christofides (1995), one from Wang (1983), one from Christofides and Whitlock (1977), and five from Fekete and Schepers (2004c). All these problem instances have known optimal solutions.
2. A set of 630 large random problems generated by Beasley (2004), following Fekete and Schepers (2004c).
3. A set of 31 zero-waste problem instances used by Leung et al. (2003).
4. A set of 21 doubly constrained problems. These problem instances are the result of transformation of the first set by Beasley (2004) into doubly constrained problems by defining, for some types of rectangles, non-zero lower bounds.

Table 1 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

4.3 GA configuration

Configuring genetic algorithms is oftentimes more an art form than a science. In our past experience with genetic algorithms based on the same evolutionary strategy (see Gonçalves and Almeida 2002; Ericsson et al. 2002; Gonçalves and Resende 2004; Gonçalves et al. 2005; Buriol et al. 2005, 2007, and Gonçalves 2007), we obtained good results with values of TOP, BOT, and Crossover Probability (CProb) in the intervals shown in Table 1.

For the population size, we have obtained good results by indexing it to the size of the problem, i.e. use small size populations for small problems and larger populations for larger problems. With this in mind and to obtain a reasonable configuration we conducted a small pilot study including 21×3 instances from the set with large random problems and 4 instances from the set with zero-waste problems. We tested all the combinations of the following values:

- $TOP \in \{0.10, 0.15, 0.20, 0.25\}$;
- $BOT \in \{0.15, 0.20, 0.25, 0.30\}$;
- $CProb \in \{0.70, 0.75, 0.80\}$;
- *Population size* with 2, 5, 10 and 15 times the number of rectangles in the problem instance.

For each of the 192 possible configurations, we made three independent runs of the algorithm (with three distinct random number generator seeds) and computed the average total value. The configuration that minimized the sum, over the pilot problem instances, was $TOP = 25\%$, $BOT = 15\%$, $CProb = 0.7$, and *Population size* = 15 \times the number of rectangles in the problem instance.

To determine the appropriate value of K for the modified total value, we tested the algorithm using values of K between 0.01 and 0.10, in steps of 0.01, on the instances of the pilot problem instances. We made three independent runs of the algorithm, using the best configuration determined previously, and computed the average modified total value. The value of K that maximized the sum, over all the pilot problem instances, of the average modified total packing value was chosen, i.e. $K = 0.03$.

After some experimentation with the problem instances in a pilot study we come to the conclusion that using 3 parallel populations and exchanging information every 15 generations was the best configuration for this type of problem.

The configuration presented in Table 2 was held constant for all experiments and all problems instances.

The computational results presented in the next section demonstrate that this configuration not only provides excellent results in terms of solution quality but also is very robust.

Table 2 Configuration used on all runs in computational experiments

Population size	$\min\{15 \times \text{number of input rectangles}, 2000\}$
Crossover probability	0.7
TOP	The 25% most fit chromosomes from the previous generation are copied to the next generation
BOT	The 15% least fit chromosomes from the previous generation are replaced with randomly generated chromosomes
Number of populations	3
Exchange of information between populations	Every 15 generations
Fitness	Maximize modified total value using $k = 0.03$
Stopping Criterion	Stop after 1000 generations

Table 3 Versions of the approach

Version	Description
BL	Using only the Bottom Left procedure
BL-L	Using only the Bottom Left procedure with Layers
BL-LB-L	Using only the Bottom Left and Left Bottom procedures with Layers
BL-LB-L-4NR	Using the Bottom Left and Left Bottom procedures with Layers and with the introduction of four non-random chromosomes in the first population

Table 4 Overall average percentage deviation from optimum/lower bound per version

Set	Description	BL	BL-L	BL-LB-L	BL-LB-L-4NR
1	Problems from literature	0	0	0	0
2	Large random problems	1.04	1.00	0.87	0.83
3	Zero-waste problems	0.48	0.48	0.24	0.17
4	Doubly constrained problems	6.36	6.36	6.36	6.36

4.4 Computational results

Our algorithm (MPGA) was implemented in C++ and the computational experiments were carried out on a computer with a Intel 2.66 GHz Xeon Quadcore CPU with the Linux CentOS 5 operating system.

Before presenting the final and detailed computational results we will present a summary of the impact of each of the different changes that we introduced in our approach. We compare the impact of each change using the four versions presented in Table 3.

Table 4 presents for each version the overall average percentage deviation from optimum/lower bound. Test set number 1 and number 4 consist of very small instances and so even the most basic version **BL** is able to obtain the best values. The impact of using layers is positive and can only be seen for test set number 2 because in test

Table 5 Computational results—problems from literature

Source of problem	Inst.	Dim. (L × W)	<i>n</i>	<i>N</i>	Opt.	PH	GA	GRASP	TABU	MPGA	CPU Time (s)
Beasley (1985)	1	10 × 10	5	10	164	164	164	164	164	164	0.00
	2	10 × 10	7	17	230	230	230	230	230	230	0.00
	3	10 × 10	10	21	247	247	247	247	247	247	0.00
	4	15 × 10	5	7	268	268	268	268	268	268	0.00
	5	15 × 10	7	14	358	358	358	358	358	358	0.01
	6	15 × 10	10	15	289	289	289	289	289	289	0.03
	7	20 × 20	5	8	430	430	430	430	430	430	0.01
	8	20 × 20	7	13	834	834	834	834	834	834	0.01
	9	20 × 20	10	18	924	924	924	924	924	924	0.01
	10	30 × 30	5	13	1452	1452	1452	1452	1452	1452	0.01
	11	30 × 30	7	15	1688	1688	1688	1688	1688	1688	0.01
	12	30 × 30	1	22	1865	1801	1865	1865	1865	1865	0.04
Hadjiconstantinou and Christofides (1995)	3	30 × 30	7	7	1178	1178	1178	1178	1178	1178	0.01
	11	30 × 30	15	15	1270	1270	1270	1270	1270	1270	0.01
Wang (1983)		70 × 40	19	42	2726	2721	2726	2726	2726	2726	0.02
Christofides and Whitlock (1977)	3	40 × 70	20	62	1860	1720	1860	1860	1860	1860	0.05
Fekete and Schepers (1997a)	1	100 × 100	15	50	27718	27486	27718	27589	27718	27718	0.05
	2	100 × 100	30	30	22502	21976	22502	21976	22502	22502	0.14
	3	100 × 100	30	30	24019	2343	24019	23743	24019	24019	0.55
	4	100 × 100	33	61	32893	31269	32893	32893	32893	32893	0.06
	5	100 × 100	29	97	27923	26332	27923	27923	27923	27923	0.02
Mean Percentage deviation from upper bound						5.49	0	0.19	0	0	

set number 3 the large instances ($M > 30$) have one rectangle per type. Finally, the results for test sets number 2 and number 3 show clearly that the combination of BL with LB and the introduction of four non-random chromosomes in the initial population provide a very positive impact on the overall average percentage deviation from optimum/lower bound.

The complete computational results for version BL-LB-L-4NR appear in Tables 5–8. All tests were performed using the configuration summarized in Table 2. In terms of computational times we cannot make any fair and meaningful comments since all the other approaches were implemented with different programming languages and tested on computers with different computing power. Instead, we limit ourselves to reporting the average running times for MPGA.

Table 5 includes a direct comparison with the results for PH (Beasley 2004), GA (Hadjiconstantinou and Iori 2007), GRASP (Alvarez-Valdes et al. 2005), and TABU (Alvarez-Valdes et al. 2007) in terms of solution quality. The results show that the

Table 6 Computational results—large random problems

<i>n</i>	Qt per type	<i>N</i>	PH	GA	GRASP	TABU	MPGA	CPU Time (s)
40	1	40	7.77	6.12	6.97	6.55	5.96	1.78
	3	120	3.54	2.82	2.22	1.95	1.60	6.39
	4	160	3.24	2.40	1.81	1.65	1.40	12.94
50	1	50	5.48	4.56	4.80	4.85	4.25	2.69
	3	150	2.35	1.89	1.50	1.27	1.03	10.90
	4	200	2.63	1.86	1.18	0.96	0.71	22.90
100	1	100	2.26	1.69	1.51	1.50	1.17	11.81
	3	300	1.27	0.99	0.47	0.31	0.21	40.72
	4	400	1.06	0.85	0.26	0.18	0.11	22.70
150	1	150	1.31	1.06	0.89	0.84	0.55	23.44
	3	450	0.60	0.32	0.14	0.07	0.05	37.56
	4	600	0.92	0.60	0.11	0.05	0.03	47.07
250	1	250	0.88	0.75	0.51	0.45	0.25	39.11
	3	750	0.57	0.51	0.04	0.01	0.01	72.70
	4	1000	0.39	0.28	0.03	0.00	0.00	20.40
500	1	500	0.26	0.21	0.07	0.00	0.03	54.81
	3	1500	0.18	0.19	0.00	0.00	0.00	6.07
	4	2000	0.18	0.19	0.00	0.00	0.00	4.71
1000	1	1000	0.09	0.15	0.00	0.00	0.00	47.24
	3	3000	0.07	0.12	0.00	0.00	0.00	8.26
	4	4000	0.07	0.17	0.00	0.00	0.00	6.75
Type I			1.64	1.24	1.04	0.95	0.86	21.27
Type II			1.70	1.37	1.14	1.06	0.88	23.50
Type III			1.66	1.35	1.03	0.94	0.74	26.80
All			1.67	1.32	1.07	0.98	0.83	23.86
Mean percentage deviations from knapsack upper bound								

MPGA algorithm, finds the optimal solutions for all the problem instances and is, therefore, as good as algorithms GA and TABU.

The results for the large random problems are presented in Table 6. This table displays aggregate results, showing that MPGA produces overall average deviations from the upper bound for all problem types (Type I, Type II, and Type III) that are always lower than those produced by all the other heuristics (see bottom of Table 6). A close look at the results shows that MPGA outperforms heuristics PH and GA for all problem types and sizes. From the table it is also clear that MPGA outperforms the GRASP and TABU heuristics not only because it obtained better average deviations from the upper bound than both (GRASP = 1.07%, TABU = 0.98% and MPGA = **0.83%**) but also because it obtained better number of best results for the 21 combinations of sizes and types (GRASP = 5/21, TABU = 8/21 and MPGA = **20/21**).

Table 7 Computational results—zero-waste problems

Source of problem	Inst.	Dim. (L × W)	<i>n</i>	<i>N</i>	Opt.	GRASP	TABU	MPGA	CPU Time (s)
Lai and Chan (1997b)	1	400 × 200	9	10	80000	80000	80000	80000	0.67
	2	400 × 200	7	15	79000	79000	79000	79000	0.01
	3	400 × 200	5	20	160000	154600	160000	160000	3.10
Jakobs (1996)	1	70 × 80	14	20	5600	5447	5600	5600	0.58
	2	70 × 80	16	25	5600	5455	5540	5540	0.70
	3	120 × 45	22	25	5400	5328	5400	5400	11.52
	4	90 × 45	16	30	4050	3978	4050	4050	3.47
	5	65 × 45	18	30	2925	2871	2925	2925	5.19
Leung et al. (2003)	1	150 × 110	40	40	16500	15856	16280	16340	23.71
	2	160 × 120	50	50	19200	18628	19044	19116	15.18
Hopper and Turton (2001)	1-1	20 × 20	16	16	400	400	400	400	0.29
	1-2	20 × 20	17	17	400	386	400	400	0.82
	1-3	20 × 20	16	16	400	400	400	400	1.41
	2-1	40 × 15	25	25	600	590	600	600	5.02
	2-2	40 × 15	25	25	600	597	600	600	8.83
	2-3	40 × 15	25	25	600	600	600	600	5.96
	3-1	60 × 30	28	28	1800	1765	1800	1800	9.22
	3-2	60 × 30	29	29	1800	1755	1800	1796	6.16
	3-3	60 × 30	28	28	1800	1774	1800	1800	7.16
	4-1	60 × 60	49	49	3600	3528	3580	3591	16.36
	4-2	60 × 60	49	49	3600	3524	3564	3588	6.96
	4-3	60 × 60	49	49	3600	3544	3580	3594	15.98
	5-1	60 × 90	73	73	5400	5308	5342	5396	62.57
	5-2	60 × 90	73	73	5400	5313	5361	5400	69.53
	5-3	60 × 90	73	73	5400	5312	5375	5392	80.83
	6-1	80 × 120	97	97	9600	9470	9548	9582	150.05
	6-2	80 × 120	97	97	9600	9453	9448	9595	92.27
	6-3	80 × 120	97	97	9600	9450	9565	9582	129.01
	7-1	160 × 240	196	196	38400	37661	38026	38146	808.03
	7-2	160 × 240	197	197	38400	37939	38145	38374	357.85
	7-3	160 × 240	196	196	38400	37745	37867	38254	650.02
Mean Percentage deviation from optimum						1.68%	0.42%	0.17%	

Table 7 presents a direct comparison between MPGA and the GRASP of Alvarez-Valdes et al. (2005) and the TABU of Alvarez-Valdes et al. (2007) on the set of zero-waste problem instances (the other benchmark heuristics have not run these instances). MPGA produces overall average deviations from the optimal values that are lower than those produced by the GRASP and TABU heuristics (GRASP = 1.68%, TABU = 0.42% and MPGA = **0.17%**). In terms of the number of best results over

Table 8 Computational results—doubly constrained problems

Source of problem	Inst.	Dim. ($L \times W$)	n	N	UB	PH	GRASP	TABU	MPGA	CPU time (s)
Beasley (1985)	1	10×10	5	10	164	164	164	164	164	0.00
	2	10×10	7	17	230	225	225	225	225	0.00
	3	10×10	10	21	247	220	220	220	220	0.00
	4	15×10	5	7	268	268	268	268	268	0.01
	5	15×10	7	14	358	301	301	301	301	0.01
	6	15×10	10	15	289	265	252	265	265	0.08
	7	20×20	5	8	430	430	430	430	430	0.01
	8	20×20	7	13	834	819	819	819	819	0.01
	9	20×20	10	18	924	924	924	924	924	0.01
	10	30×30	5	13	n/f	n/f	n/f	n/f	n/f	0.02
	11	30×30	7	15	1688	1505	1518	1518	1518	0.47
	12	30×30	10	22	1865	1666	1648	1672	1672	0.05
Hadjiconstantinou and Christofides (1995)	3	30×30	7	7	1178	1178	1178	1178	1178	0.01
	11	30×30	15	15	1270	1216	1216	1216	1216	0.02
Wang (1983)		70×40	19	42	2726	2499	2700	2716	2716	0.03
Christofides and Whitlock (1977)	3	40×70	20	62	1860	1600	1720	1720	1720	0.03
Fekete and Schepers (1997a)	1	100×100	15	50	27718	25373	24869	25384	25384	16.87
	2	100×100	30	30	22502	17789	19083	19657	20678	1.22
	3	100×100	30	30	n/f	n/f	n/f	n/f	n/f	0.58
	4	100×100	33	61	32893	27556	27898	28974	28974	0.42
	5	100×100	29	97	27923	21997	22011	22011	22140	4.53
Mean Percentage deviation from upper bound						8.11	7.36	6.62	6.36	

all the instance in this set MPGA clearly outperforms the other two (GRASP = 5/31, TABU = 17/31, MPGA = **30/31**).

Finally, Tables 8 show the results for the doubly-constrained test problem instance of the fourth set, where the heuristics PH (Beasley 2004), GRASP (Alvarez-Valdes et al. 2005), TABU Alvarez-Valdes et al. (2007) and MPGA are contrasted. The upper bound (UB) corresponds to the solution of the constrained problem using the knapsack problem formulation of Sect. 2. The problems for which the algorithms do not find solutions are not feasible, but are maintained in the set of test problems and therefore are included in the table. For these problem instances, MPGA clearly outperforms the other two both in terms of average deviation from the lower bound (PH = 8.11%, GRASP = 7.36%, TABU = 6.62%, MPGA = **6.36%**) as well as in terms of the number of best solutions obtained out of the 21 problems (11/21, 12/21, 17/21, **19/21**).

5 Concluding remarks

In this paper we addressed a constrained two-dimensional (2D) packing problem, where a fixed set of small rectangles has to be cut from a larger stock rectangle so as to maximize the value of the rectangles packed. An algorithm which hybridizes a placement strategy with a multi-population genetic algorithm based on random keys was proposed. The approach was tested on four sets of instances taken from the literature and compared with other four approaches. The experimental results demonstrate the effectiveness and robustness of the proposed heuristic when compared with other approaches.

Acknowledgements This work has been supported by funds granted by Fundação para a Ciência e Tecnologia (FCT) project PTDC/GES/72244/2006.

References

- Alvarez-Valdes R, Parreño F, Tamarit J (2005) A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems. *J Oper Res Soc* 56:414–425
- Alvarez-Valdes R, Parreño F, Tamarit J (2007) A tabu search algorithm for a two-dimensional non-guillotine cutting problem. *Eur J Oper Res* 183:1167–1182
- Amaral A, Letchford AN (2001) An improved upper bound for the two-dimensional non-guillotine cutting problem. Technical report, Lancaster University, UK. Available online at <http://www.lancs.ac.uk/staff/letchfoa/ngc.doc>
- Arenales M, Morabito R (1995) An and/or-graph approach to the solution of two dimensional guillotine cutting problems. *Eur J Oper Res* 84:599–617
- Bean JC (1994) Genetics and random keys for sequencing and optimization. *ORSA J Comput* 6:154–160
- Beasley JE (1985) An exact two-dimensional non-guillotine cutting tree search procedure. *Oper Res* 33:49–64
- Beasley JE (2004) A population heuristic for constrained two-dimensional non-guillotine cutting. *Eur J Oper Res* 156:601–627
- Boschetti MA, Hadjiconstantinou E, Mingozzi A (2002) New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem. *IMA J Manag Math* 13:95–119
- Buriol LS, Resende MGC, Ribeiro CC, Thorup M (2005) A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks* 46:36–56
- Buriol LS, Resende MGC, Thorup M (2007) Survivable IP network design with OSPF routing. *Networks* 49:51–64
- Caprara A, Monaci M (2004) On the 2-dimensional knapsack problem. *Oper Res Lett* 32:5–14
- Christofides N, Whitlock C (1977) An algorithm for two dimensional cutting problems. *Oper Res* 25:31–44
- Dowsland KA, Dowsland WB (1992) Packing problems. *Eur J Oper Res* 56:2–14
- Dyckhoff H (1990) A typology of cutting and packing problems. *Eur J Oper Res* 44:145–159
- Ericsson M, Resende MGC, Pardalos PM (2002) A genetic algorithm for the weight setting problem in OSPF routing. *J Combin Optim* 6:299–333
- Fekete S, Schepers J (1997a) A new exact algorithm for general orthogonal d -dimensional knapsack problems. In: *Algorithms—ESA '97*. Springer lecture notes in computer science, vol 1284. Springer, Berlin, pp 144–156
- Fekete SP, Schepers J (1997b) On higher-dimensional packing I: Modeling. Technical Report ZPR 97-288, Mathematisches Institut, Universität zu Köln
- Fekete SP, Schepers J (1997c) On higher-dimensional packing II: Bounds. Technical Report ZPR97-289, Mathematisches Institut, Universität zu Köln
- Fekete SP, Schepers J (2004a) A combinatorial characterization of higher-dimensional orthogonal packing. *Math Oper Res* 29:353–368
- Fekete SP, Schepers J (2004b) A general framework for bounds for higher-dimensional orthogonal packing problems. *Math Methods Oper Res* 60:311–329

- Fekete SP, Schepers J (2004c) An exact algorithm for higher-dimensional orthogonal packing. Technical report, Mathematisches Institut, Universität zu Köln. Working paper. Available online at <http://www.math.tu-bs.de/~fekete>
- Feo TA, Resende MGC (1989) A probabilistic heuristic for a computationally difficult set covering problem. *Oper Res Lett* 8:67–71
- Feo TA, Resende MGC (1995) Greedy randomized adaptive search procedures. *J Glob Optim* 6:109–133
- Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York
- Goldberg D (1989) Genetic algorithms in search optimization and machine learning. Addison-Wesley, Reading
- Gonçalves JF (2007) A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *Eur J Oper Res* 183:1212–1229
- Gonçalves JF, Almeida JR (2002) A hybrid genetic algorithm for assembly line balancing. *J Heuristics* 8:629–642
- Gonçalves JF, Resende MGC (2004) An evolutionary algorithm for manufacturing cell formation. *Comput Ind Eng* 47:247–273
- Gonçalves JF, Resende MGC (2009) Biased random key genetic algorithms for combinatorial optimization. Technical report, AT&T Labs Research Technical Report, Florham Park, NJ 07733 USA
- Gonçalves JF, Mendes JJM, Resende MGC (2005) A hybrid genetic algorithm for the job shop scheduling problem. *Eur J Oper Res* 167:77–95
- Hadjiconstantinou E, Christofides N (1995) An exact algorithm for general, orthogonal, two dimensional knapsack problems. *Eur J Oper Res* 83:39–56
- Hadjiconstantinou E, Iori M (2007) A hybrid genetic algorithm for the two-dimensional knapsack problem. *Eur J Oper Res* 183:1150–1166
- Haessler RW, Sweeney PE (1991) Cutting stock problems and solution procedures. *Eur J Oper Res* 54:141–150
- Healy P, Creavin M, Kuusik A (1999) An optimal algorithm for rectangle placement. *Oper Res Lett* 24:73–80
- Hopper E, Turton BCH (2001) An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *Eur J Oper Res* 128:34–57
- Jakobs S (1996) On genetic algorithms for the packing of polygons. *Eur J Oper Res* 88:165–181
- Lai KK, Chan JWM (1997a) An evolutionary algorithm for the rectangular cutting stock problem. *Int J Ind Eng* 4:130–139
- Lai KK, Chan JWM (1997b) Developing a simulated annealing algorithm for the cutting stock problem. *Comput Ind Eng* 32:115–127
- Leung TW, Chan CK, Troutt MD (2001) Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem. *Comput Ind Eng* 40:201–214
- Leung TW, Chan CK, Troutt MD (2003) Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. *Eur J Oper Res* 141:241–252
- Liu D, Teng H (1999) An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *Eur J Oper Res* 112:413–420
- Scheithauer G, Terno J (1993) Modeling of packing problems. *Optimization* 28:63–84
- Spears WM, Dejong KA (1991) On the virtues of parameterized uniform crossover. In: Proceedings of the fourth international conference on genetic algorithms, pp 230–236
- Sweeney PE, Paternoster ER (1992) Cutting and packing problems: A categorized, application-orientated research bibliography. *J Oper Res Soc* 43:691–706
- Tsai RD, Malstrom EM, Meeks HD (1988) A two-dimensional palletizing procedure for warehouse loading operations. *IIE Trans* 20:418–425
- Wang PY (1983) Two algorithms for constrained two-dimensional cutting stock problems. *Oper Res* 31:573–586
- Wäscher G, Haussner H, Schumann H (2007) An improved typology of cutting and packing problems. *Eur J Oper Res* 183:1109–1130