# A multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem[*]

**José Fernando Gonçalves**

**Faculdade de Economia do Porto, LIAAD**

**Rua Dr. Roberto Frias, s/n, 4200-464 Porto, Portugal**

**Mauricio G. C. Resende**

**Algorithms and Optimization Research Department, AT&T Labs Research,**

**180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.**

This paper addresses a constrained two-dimensional (2D), non-guillotine restricted, packing problem, where a fixed set of small rectangles has to be paced 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

[*] Supported by Fundação para a Ciência e Tecnologia (FCT) project PTDC/GES/7224/2006. AT&T Labs Research Technical Report TD-7M7QJG. Date: 2008-12-10

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 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,b,c, 2004a,b) 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 (997a) 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 (997b) proposed a heuristic based on an evolutionary strategy approach. They used the same representation as in Lai and Chan (997a). 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 (997a) and discussed producing a cutting pattern from it using both the difference process algorithm of Lai and Chan (997a) 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 Section 2, we define the problem formally, and in Section 3, propose the new approach, describing the genetic algorithm, the placement strategy, and the fitness function. In Section 4, we present experimental results and in Section 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 $(w_i, h_i)$, $i = 1, \dots, n$, each of width $w_i$ and height $h_i$. Each rectangle $i$ 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 $i = 1, \ldots, n$. Each rectangle $i = 1, \ldots, n$ 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 $(w_i, h_i)$, $i = 1, \ldots, n$, are integers. To simplify notation, we shall use $M = \sum_{i=1}^{n} Q_i$ as the maximum number of pieces 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 $i = 1, \ldots, n$, which is a trivial bound;

2. In the *constrained* type, we have: for all $i = 1, \ldots, n$, $P_i = 0$ and $\exists\ j \in \{1, \ldots, n\}$ such that $Q_j < \lfloor (W \times H)/(w_j \times h_j) \rfloor$;

3. In the *doubly constrained* type, we have: $\exists\ i \in \{1, \ldots, n\}$ such that $P_i > 0$ and $\exists\ j \in \{1, \ldots, n\}$ 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 $P_i$:

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

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 giving 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 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 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

Genetic algorithms are adaptive methods that are used to solve search and optimization problems (Goldberg, 1989; Beasley et al., 1993). They are based on the genetic process of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection, i.e. *survival of the fittest*, first clearly stated by Charles Darwin (1859), in *The Origin of Species.* By mimicking this process, genetic algorithms, if suitably encoded, are able to *evolve* solutions to real world problems. Before a genetic algorithm can be run, an *encoding* (or *representation*) for the problem must be devised. A *fitness function*, which assigns a figure of merit to each encoded solution, is also required. During the run, parents are *selected* for reproduction and *recombined* to generate offspring (see high-level pseudo-code in Figure 2).

It is assumed that a potential solution to a problem may be represented as a set of parameters. These parameters (known as *genes*) are joined together to form a string of values (*chromosome*). In genetic terminology, the set of parameters represented by a particular chromosome is referred to as an *individual.* The fitness of an individual depends on its chromosome and is evaluated by the fitness function. The individuals, during the reproductive phase, are selected from the population and *recombined*, producing offspring, which comprise the next generation. Parents are randomly selected from the population using a scheme that favors fitter individuals. Having selected two parents,

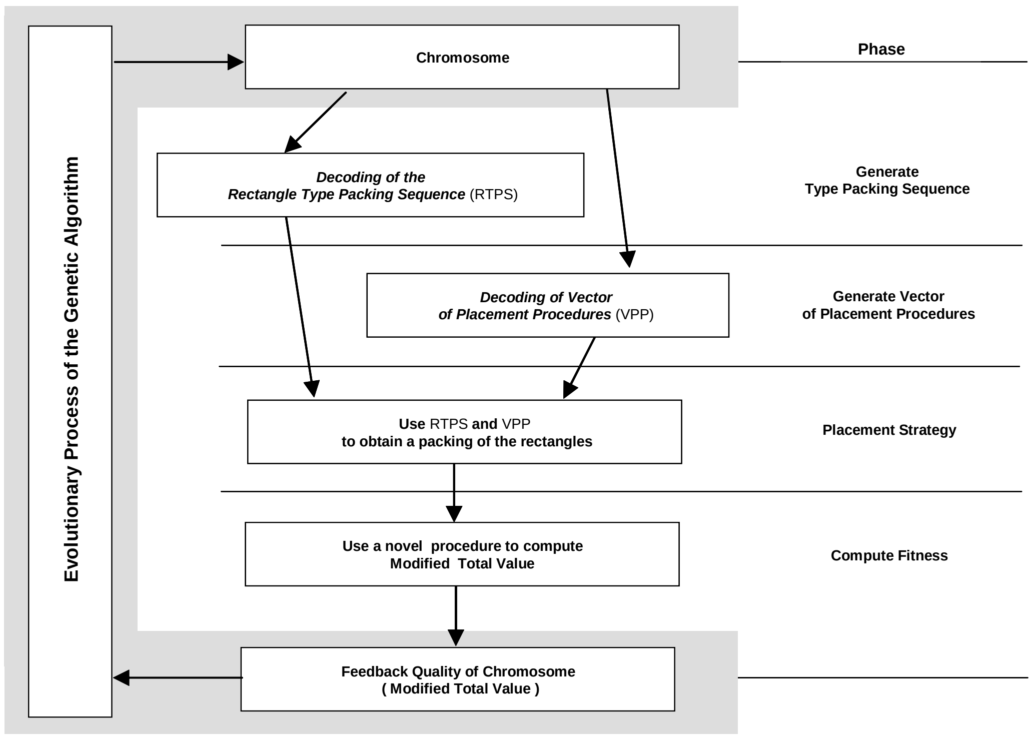

Evolutionary Process of the Genetic Algorithm
Chromosome
Phase
Decoding of the Rectangle Type Packing Sequence (RTPS)
Generate Type Packing Sequence
Decoding of Vector of Placement Procedures (VPP)
Generate Vector of Placement Procedures
Use RTPS and VPP to obtain a packing of the rectangles
Placement Strategy
Use a novel procedure to compute Modified Total Value
Compute Fitness
Feedback Quality of Chromosome ( Modified Total Value )

Figure 1: Architecture of the new approach.

their chromosomes are *recombined*, typically using mechanisms of *crossover*. Mutation is usually applied to some individuals, to guarantee population diversity.

### 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

```
procedure GENETIC-ALGORITHM
1  Generate initial population $P_0$;
2  Evaluate population $P_0$;
3  Initialize generation counter $g \leftarrow 0$;
4  while stopping criteria not satisfied repeat
5      Select some elements from $P_g$ to copy into $P_{g+1}$;
6      Crossover some elements of $P_g$ and put into $P_{g+1}$;
7      Mutate some elements of $P_g$ and put into $P_{g+1}$;
8      Evaluate new population $P_{g+1}$;
9      Increment generation counter: $g \leftarrow g + 1$;
10 end while;
end GENETIC-ALGORITHM;
```

Figure 2: Pseudo-code of a standard genetic algorithm.

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 Section 3.3.

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

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

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

The first $M$ genes are used to obtain the *Rectangle Type Packing Sequence* (RTPS) while the last $M$ 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 $M$ genes of each chromosome into an RTPS is accomplished by sorting the genes and rectangles types in ascending order. Figure 3 presents an example of the decoding process for the RTPS. In the example the are there four types of 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 in 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 Section 3.3). The decoding (mapping) of the last $M$ genes of each chromosome into a VPP is accomplished using, for $i = 1, \dots, M$, the following expression:

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

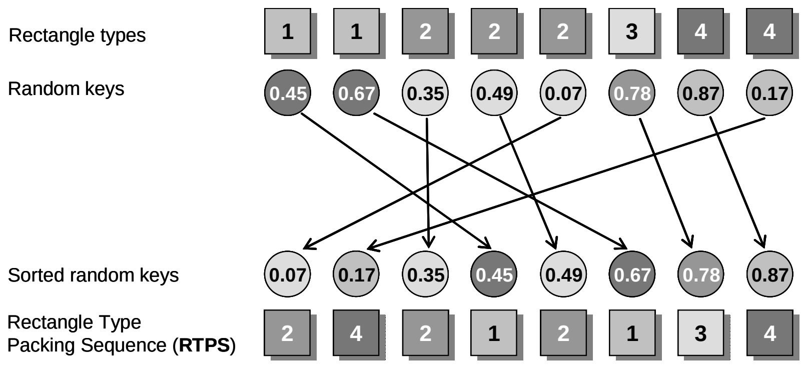


Figure 3: Chromosome decoding procedure.

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

*Parameterized 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 (*TOP* in Figure 5) and the other chosen randomly from the full old population (including chromosomes copied to the next generation in the elitist pass), at each gene we toss a biased coin to select which parent will contribute the allele. 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 $CProb = 0.7$.

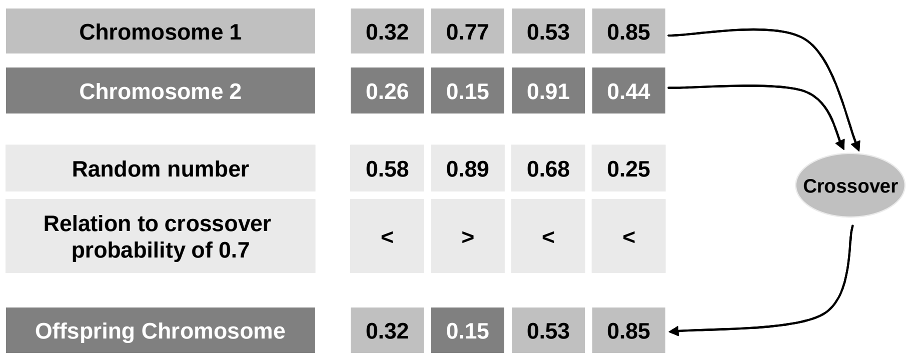


Figure 4: Example of parameterized uniform crossover with crossover probability equal to 0.7.

In Section 4 we describe how we determine this value empirically. Figure 4 shows one potential crossover outcome.

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 from the same distribution as the initial population (see $BOT$ in Figure 5). 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 5 depicts the transitional process between two consecutive generations.

#### 3.2.3 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 Section 4) inserts only the best two chromosomes in all populations.

### 3.3 Placement strategy

Rectangles are placed in the stock rectangle, one at a time, in the order defined by the RTPS supplied by the genetic algorithm (see Section 3.2.2). While trying to place a rectangle 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 is placed in an ERS where it fits and selected according to the placement procedure defined for its placement by the genetic algorithm.

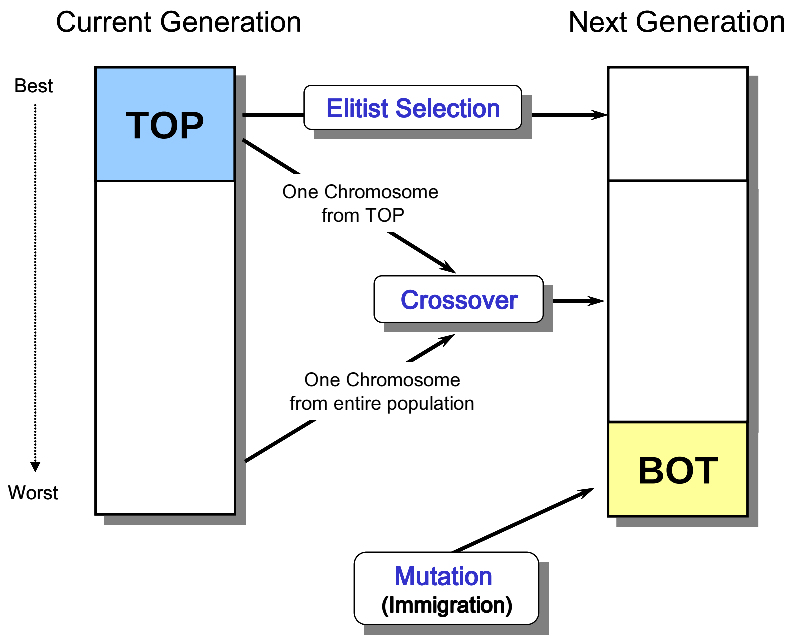


Figure 5: Transitional process between consecutive generations.

Initially we considered only a *Bottom-Left* (BL) 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), see pseudo-code in Figure 7. 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 Figure 8, we present, for the problem in Figure 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 Figure 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 $y_i < y_j$ 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 Figure 10. In summary, our placement strategy uses two placement procedures, the Bottom-Left and the Left-Bottom, to construct a packing of the rectangles. The vector of placement procedures (VPP), supplied by the genetic

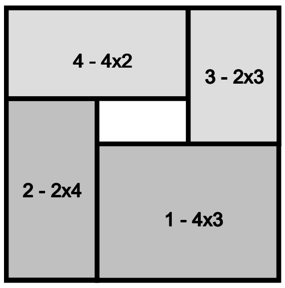


Figure 6: Example of an optimal solution that cannot be produced by the Bottom-Left (BL) placement procedure.

algorithm, indicates, for each rectangle type to be packed, whether it should be placed with the BL or LB procedure. In Figure 9 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 use a horizontal layer and when we the LB placement procedure we use a vertical layer (see Figure 11).

Let $M$ be the number of rectangles to be placed in the stock rectangle and let $r_i$ be the $i$-th rectangle to be placed as defined by the RPS. Furthermore, let $PP_i$ be the placement procedure that the genetic algorithm assigned for the placement of rectangle $r_i$. The pseudo-code of the placement procedure is presented in Figure 12.

To generate and keep track of the ERSs, we make use of the *Difference Process* (DP), developed by Lai and Chan (997a). In short, the DP can be described as follows. Suppose we have two small rectangles to be packed in a stock rectangle as depicted in Figure 13a. Furthermore, assume that we will pack rectangle 1 first and then rectangle 2. At the very beginning, there is only one ERS available (ERS-1) where we can pack rectangle 1 and its bottom left corner is precisely at the origin of the stock rectangle (see Figure 13b). After the first rectangle is packed, usually two new ERSs are generated (unless the height or the width of the first rectangle is the same as that of the stock rectangle (see ERS-2 and ERS-3 in Figure 13b). Every time a new rectangle is packed, it intersects some of the available ERSs, so we need to update the list of available ERSs before we pack the next rectangle. From each existing ERS which intersects with a rectangle nontrivially, it is easy to see that at most three new empty rectangular spaces are generated. From the newly generated ERSs, we check if any one is entirely contained in another, and, in that case, remove it. This is the so-called *elimination process* in Lai and Chan (997a). In this way, we update the list of available ERSs whenever a rectangle is packed (see ERS-2.1, ERS-3.1 and ERS-3.2 in Figure 13c).

```
procedure BL($r_i$)
1  Let $r_i$ be the $i$-th rectangle to be packed in the stock rectangle;
2  Let $n_{ERS}$ be the number of available ERSs;
3  Initialize $L \leftarrow W$;
4  Initialize $B \leftarrow H$;
5  for $k = 1, \ldots, n_{ERS}$ do
6      Let $x(ERS_k)$ be the $x$ coordinate of the the bottom left
·      corner of $ERS_k$;
7      Let $y(ERS_k)$ be the $y$ coordinate of the the bottom left
·      corner of $ERS_k$;
8      if $r_i$ fits in $ERS_k$ then
9          if $x(ERS_k) \leq L$ then
10             $L \leftarrow x(ERS_k)$;
11         else if $x(ERS_k) = L$ and $y(ERS_k) \leq B$ then
12             $B \leftarrow y(ERS_k)$ ;
13         end if
14     end if
15 end for
16 Place rectangle $r_i$ at coordinates $(L, B)$;
end BL;
```

Figure 7: Pseudo-code of the Bottom-Left (BL) placement procedure.

## 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:

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

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. Consider, for example, *Packing-1* and *Packing-2* depicted in Figure 14. Assuming that the value of each rectangular piece is equal to its area, then both packings have a *Total value* of $(3 \times 1 + 4 \times 1 + 1 \times 2 + 2 \times 1) = 11$. However, *Packing-2* has all the unused area concentrated in one contiguous space, while for *Packing-1* the unused area is made up of three non-adjacent areas. Any rectangle that can be packed in *Packing-1* can also be packed in *Packing-2*. However, the opposite is not true. For example, we cannot pack a $3 \times 1$ or a $4 \times 1$ rectangle in *Packing-1*, whereas we can in *Packing-2*. Therefore, it is easy to conclude that *Packing-2* has a higher potential than *Packing-1* to have its total value increased.

To be able to capture the improvement potential of different packings which have the

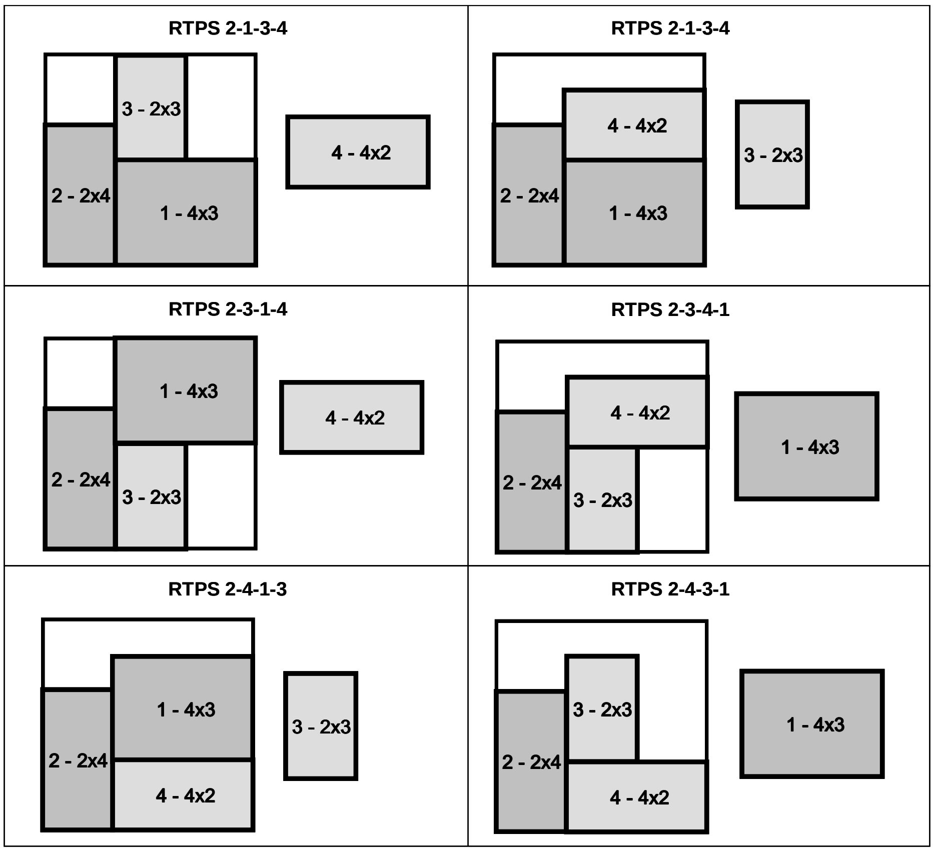


Figure 8: Solutions found by the Bottom-Left (BL) placement procedure for the problem in Figure 6 for all Rectangle Packing Sequences with rectangle 2 in the first position.

same total value, we use a fitness measure that we call *modified total value* which is an adaptation of the fitness measure propose by Gonçalves (2007). The basic idea consists in adding to the total value of a packing a small value proportional to the value of the largest empty rectangular space remaining in the packing, i.e.,

$$\begin{aligned} \textit{Modified Total Value} &= \textit{Total Value} + \\ &\qquad \beta \times \textit{Area of Largest ERS} \times \textit{Minimum Value Per Unit Area}, \end{aligned}$$

where $\beta$ is a parameter that depends on the problem data. The parameter $\beta$ should be chosen carefully because the modified total value must be able to accommodate the following two objectives:

1. For solutions with the same real total value, we want the modified total value to assign better fitness values to those solutions that have contiguous empty spaces (i.e. have a few large empty spaces rather than many small empty spaces).

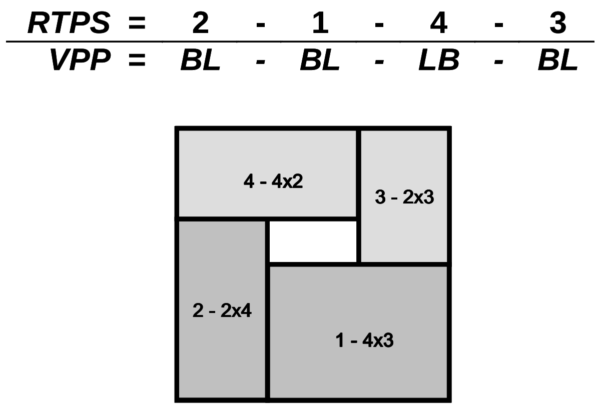


Figure 9: Optimal solution, for the problem in Figure 6, found by combining the Bottom-Left (BL) and the Left-Bottom (LB) placement procedures.

2. We do not want to have empty spaces in the final solutions obtained by the overall algorithm since the empty spaces do not increase the real total value.

Very small values of $\beta$ will not promote the solutions with large empty spaces, while large values of $\beta$ will cause the genetic algorithm to simply search for solutions with large empty spaces. To avoid having solutions with large empty spaces, we require that the increase in total value caused by the term

$$\beta \times \textit{Area of Largest ERS} \times \textit{Minimum Value Per Unit Area}$$

should be significantly less than the increase in total value caused by the packing of the smallest value rectangle, i.e.

$$\beta \times \textit{Area of Largest ERS} \times \textit{Minimum Value Per Unit Area} \ll \textit{Minimum Value Rectangle}.$$

Note that, in the worst case, we might have

$$\textit{Area of Largest ERS} = \textit{Area of Stock Rectangle}$$

which is equivalent to having

$$\beta \ll \frac{\textit{Minimum Value Rectangle}}{\textit{Area of Stock Rectangle} \times \textit{Minimum Value Per Unit Area}}.$$

We chose to compute $\beta$ using the expression

$$\beta = K \times \frac{\textit{Minimum Value Rectangle}}{\textit{Area of Stock Rectangle} \times \textit{Minimum Value Per Unit Area}},$$

where $K << 1$ is a constant. Note that this expression also satisfies the previous expression. To determine the appropriate value of $K$, we experimented with values of $K$ varying

```
procedure LB($r_i$)
1  Let $r_i$ be the $i$-th rectangle to be packed in the stock rectangle;
2  Let $n_{ERS}$ be the number of available ERSs;
3  Initialize $L \leftarrow W$;
4  Initialize $B \leftarrow H$;
5  for $k = 1, \ldots, n_{ERS}$ do
6      Let $x(ERS_k)$ be the $x$ coordinate of the the bottom left
·      corner of $ERS_k$;
7      Let $y(ERS_k)$ be the $y$ coordinate of the the bottom left
·      corner of $ERS_k$;
8      if $r_i$ fits in $ERS_k$ then
9          if $y(ERS_k) \leq B$ then
10             $B \leftarrow y(ERS_k)$;
11         else if $y(ERS_k) = B$ and $x(ERS_k) \leq L$ then
12             $L \leftarrow x(ERS_k)$ ;
13         end if
14     end if
15 end for
16 Place rectangle $r_i$ at coordinates $(L, B)$;
end LB;
```

Figure 10: Pseudo-code of the Left-Bottom (LB) placement procedure.

between 0.01 and 0.10 in steps of 0.01 and concluded that 0.03 was an appropriate value. We can now fully define the modified total value (MTV) as

$$\textit{Modified Total Value} = \textit{Total Value} + 0.03 \times \frac{\textit{Minimum Value Rectangle} \times \textit{Area of Largest ERS} \times \textit{Minimum Value Per Unit Area}}{\textit{Area of Stock Rectangle} \times \textit{Minimum Value Per Unit Area}}$$

or equivalently

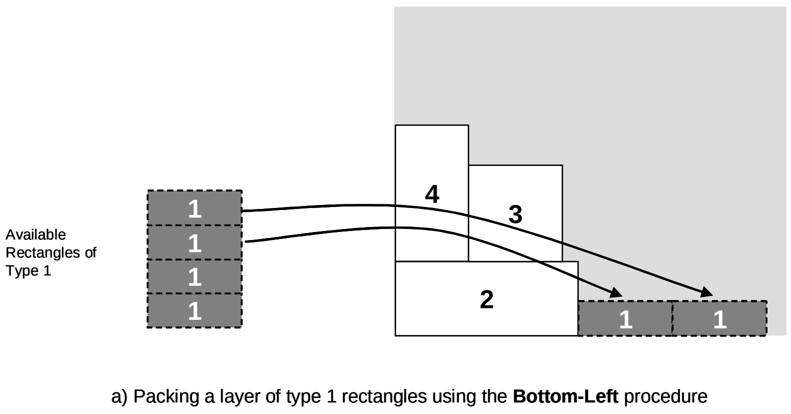
$$\textit{Modified Total Value} = \textit{Total Value} + 0.03 \times \frac{\textit{Minimum Value Rectangle} \times \textit{Area of Largest ERS}}{\textit{Area of Stock Rectangle}}.$$

The *Modified Total Value* for *Packing-1* and *Packing-2* of Figure 14 are, respectively,

$$MTV_1 = \left(11 + 0.03 \times \frac{2 \times 2 \times 1}{16}\right) = 11.0075$$

and

$$MTV_2 = \left(11 + 0.03 \times \frac{2 \times 4 \times 1}{16}\right) = 11.015.$$


Available Rectangles of Type 1
1
1
1
1
4
3
2
1
1
a) Packing a layer of type 1 rectangles using the **Bottom-Left** procedure

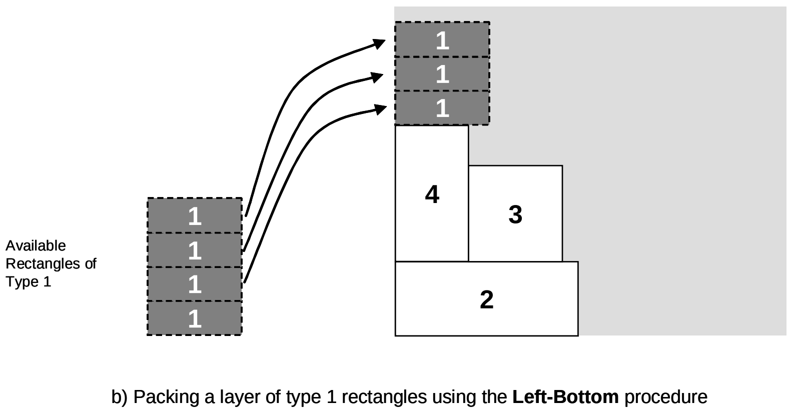

1
1
1
4
3
2
Available Rectangles of Type 1
1
1
1
1
b) Packing a layer of type 1 rectangles using the **Left-Bottom** procedure

Figure 11: Packing of a layer using a) the Bottom-Left procedure and b) the Left-Bottom procedure.

As desired, *Packing-2* has a higher MTV than *Packing-1*. The use of the modified total value as the fitness function was responsible for significantly improving the total values obtained in the computational experiments.

As 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 type of piece $i$ 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   for $i = 1, \ldots, M$ do
2       if $PP_i = BL$ then
3           Let $ERS_i^*$ be the ERS, in the list of available ERSs,
·           in which rectangle $r_i$ is placed when the Bottom-Left
·           placement heuristic is applied;
4       end if
5       else if $PP_i = LB$ then
6           Let $ERS_i^*$ be the ERS, in the list of available ERSs,
·           in which rectangle $r_i$ is placed when the Left-Bottom
·           placement heuristic is applied;
7       end if
8       if $ERS_i \neq \emptyset$ then
9           Place $r_i$ at bottom left corner of $ERS_i^*$;
10          Update list of available ERSs using the DP process
·           of Lai and Chan (997a);
11      end if
end PLACEMENT;
```

Figure 12: Pseudo-code of the placement procedure of the hybrid heuristic.

## 4 Numerical experiments

In this section we report 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 know optimal solutions.

2. A set of 630 large random problems generated by Beasley (2004), following Fekete and Schepers (2004c). All problems have a stock rectangle of size $100 \times 100$. There are three types of instances, (Type I, Type II and Type III), depending on the distribution of types of pieces of each class of rectangle. For each type, 210 instances are generated as follows. For each value $m = 40, 50, 100, 150, 250, 500, 1000$ of the number of piece types, ten problems are randomly generated with $P_i = 0$ and $Q_i = 1, 3, 4$, for all $i = 1, \ldots, n$.

3. A set of 31 zero-waste problem instances used by Leung et al. (2003). This set consists of: three instances from Lai and Chan (997a), five instances from Jakobs (1996), two from Leung et al. (2003), and 21 from Hopper and Turton (2001). In these problems, the value of each piece corresponds to its area, and the objective is to minimize the waste of the stock rectangle. The problems have been generated in such a way that the optimal solution is a cutting pattern with zero-waste. These problem instances can be considered complementary to the ones in the first two sets used by Beasley (2004). The problem instances of the second set can be considered selection problems where there are many available pieces and only a small fraction of them will form part of the solution. However, the Leung et al. (2003) and Hopper and Turton (2001) problem instances are jigsaw problems where almost all the available pieces will form part of the solution and the difficulty here is to find their correct position in the packing pattern. An algorithm considered as being general purpose should produce good solutions for both types of problems.

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. All rectangle types $i = 1, \ldots, m$ whose areas (in total) do not exceed one-third of the area of the stock rectangle had their lower bounds $P_i$ set to 1. This was done by initially setting all $P_i = 0$, for $i = 1, \ldots, m$ and, in turn, for each rectangle type $i = 1, \ldots, m$ satisfying
$$\sum\nolimits_{j=1(j \neq i)}^{m} (l_j \cdot w_j) P_j + l_i \cdot w_i \leq (H \cdot W)/3,$$
the lower bound $P_i$ is set to 1. This set of problems will be used to test the heuristic in the general case of doubly constrained problems.

Table 1: Range of Parameters in past implementations.

| Parameter | Interval |
|---|---|
| TOP | 0.10 - 0.25 |
| BOT | 0.15 - 030 |
| 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), Buriol et al. (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, we conducted a small pilot study to obtain a reasonable configuration. 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.

Table 2: Configuration used on all runs in computational experiments.

| | |
|---|---|
| Population size | $min\ \{15 \times number\, of\, input\, rectangles\,,\, 2000\}$ |
| 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 |

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.

## 4.4 Computational results

Our algorithm (MPGA) was implemented in MS Visual Basic 6.0 and the computational experiments were carried out on a computer with a 3.2 GHz Pentium CPU with the MS Windows XP operating system.

The complete computational results appear in Tables 3–6. All tests where 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 3 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 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 4. 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 4). 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

Table 3: Computational results - problems from literature.

| Source of problem | Inst. | Dim. (L×W) | m | M | Opt. | PH | GA | GRASP | TABU | MPGA | CPU Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Beasley (1985) | 1 | 10x10 | 5 | 10 | 164 | **164** | **164** | **164** | **164** | **164** | 0.01 |
| | 2 | 10x10 | 7 | 17 | 230 | **230** | **230** | **230** | **230** | **230** | 0.01 |
| | 3 | 10x10 | 10 | 21 | 247 | **247** | **247** | **247** | **247** | **247** | 0.05 |
| | 4 | 15x10 | 5 | 7 | 268 | **268** | **268** | **268** | **268** | **268** | 0.00 |
| | 5 | 15x10 | 7 | 14 | 358 | **358** | **358** | **358** | **358** | **358** | 0.01 |
| | 6 | 15x10 | 10 | 15 | 289 | **289** | **289** | **289** | **289** | **289** | 0.05 |
| | 7 | 20x20 | 5 | 8 | 430 | **430** | **430** | **430** | **430** | **430** | 0.01 |
| | 8 | 20x20 | 7 | 13 | 834 | **834** | **834** | **834** | **834** | **834** | 0.11 |
| | 9 | 20x20 | 10 | 18 | 924 | **924** | **924** | **924** | **924** | **924** | 0.17 |
| | 10 | 30x30 | 5 | 13 | 1452 | **1452** | **1452** | **1452** | **1452** | **1452** | 0.01 |
| | 11 | 30x30 | 7 | 15 | 1688 | **1688** | **1688** | **1688** | **1688** | **1688** | 0.02 |
| | 12 | 30x30 | 1 | 22 | 1865 | 1801 | **1865** | **1865** | **1865** | **1865** | 0.10 |
| Hadjiconstantinou and Christofides (1995) | 3 | 30x30 | 7 | 7 | 1178 | **1178** | **1178** | **1178** | **1178** | **1178** | 0.00 |
| | 11 | 30x30 | 15 | 15 | 1270 | **1270** | **1270** | **1270** | **1270** | **1270** | 0.02 |
| Wang (1983) | | 70x40 | 19 | 42 | 2726 | 2721 | **2726** | **2726** | **2726** | **2726** | 0.83 |
| Christofides and Whitlock (1977) | 3 | 40x70 | 20 | 62 | 1860 | 1720 | **1860** | **1860** | **1860** | **1860** | 4.51 |
| Fekete and Schepers (1997a) | 1 | 100x100 | 15 | 50 | 27718 | 27486 | **27718** | 27589 | **27718** | **27718** | 0.49 |
| | 2 | 100x100 | 30 | 30 | 22502 | 21976 | **22502** | 21976 | **22502** | **22502** | 4.96 |
| | 3 | 100x100 | 30 | 30 | 24019 | 2343 | **24019** | 23743 | **24019** | **24019** | 1.14 |
| | 4 | 100x100 | 33 | 61 | 32893 | 31269 | **32893** | **32893** | **32893** | **32893** | 1.44 |
| | 5 | 100x100 | 29 | 97 | 27923 | 26332 | **27923** | **27923** | **27923** | **27923** | 0.77 |
| Mean Percentage deviation from upper bound | | | | | | 5.49 | **0** | 0.19 | **0** | **0** | |

upper bound than both (GRASP=1.07%, TABU=0.98% and MPGA=**0.87%**) but also because it obtained better number of best results for the 21 combinations of sizes and types (GRASP=5/21, TABU=11/21 and MPGA=**14/21**).

Table 5 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=**0.24%**). In terms of the number of best results over all the instance in this set MPGA clearly outperforms the other two ( GRASP=5/31, TABU=19/31, MPGA=**30/31**).

Finally, Tables 6 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 Section 2. The problems for which the algorithms do not find solutions are

Table 4: Computational results - Large random problems.

| m | Qt per type | M | PH | GA | GRASP | TABU | MPGA | CPU Time (s) |
|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 40 | 7.77 | 6.12 | 6.97 | 6.55 | **5.93** | 6.90 |
| | 3 | 120 | 3.54 | 2.82 | 2.22 | 1.95 | **1.71** | 50.34 |
| | 4 | 160 | 3.24 | 2.40 | 1.81 | 1.65 | **1.51** | 88.75 |
| 50 | 1 | 50 | 5.48 | 4.56 | 4.80 | 4.85 | **4.29** | 5.34 |
| | 3 | 150 | 2.35 | 1.89 | 1.50 | 1.27 | **1.08** | 95.03 |
| | 4 | 200 | 2.63 | 1.86 | 1.18 | 0.96 | **0.81** | 102.38 |
| 100 | 1 | 100 | 2.26 | 1.69 | 1.51 | 1.50 | **1.22** | 52.50 |
| | 3 | 300 | 1.27 | 0.99 | 0.47 | 0.31 | **0.30** | 201.65 |
| | 4 | 400 | 1.06 | 0.85 | 0.26 | **0.18** | 0.19 | 118.01 |
| 150 | 1 | 150 | 1.31 | 1.06 | 0.89 | 0.84 | **0.68** | 104.45 |
| | 3 | 450 | 0.60 | 0.32 | 0.14 | **0.07** | 0.08 | 177.84 |
| | 4 | 600 | 0.92 | 0.60 | 0.11 | **0.05** | 0.06 | 234.83 |
| 250 | 1 | 250 | 0.88 | 0.75 | 0.51 | 0.45 | **0.33** | 114.68 |
| | 3 | 750 | 0.57 | 0.51 | 0.04 | **0.01** | 0.04 | 359.49 |
| | 4 | 1000 | 0.39 | 0.28 | 0.03 | **0.00** | 0.01 | 170.14 |
| 500 | 1 | 500 | 0.26 | 0.21 | 0.07 | **0.00** | 0.05 | 303.61 |
| | 3 | 1500 | 0.18 | 0.19 | **0.00** | **0.00** | **0.00** | 75.61 |
| | 4 | 2000 | 0.18 | 0.19 | **0.00** | **0.00** | **0.00** | 80.07 |
| 1000 | 1 | 1000 | 0.09 | 0.15 | **0.00** | **0.00** | 0.01 | 168.18 |
| | 3 | 3000 | 0.07 | 0.12 | **0.00** | **0.00** | **0.00** | 162.98 |
| | 4 | 4000 | 0.07 | 0.17 | **0.00** | **0.00** | **0.00** | 169.28 |
| Type I | | | 1.64 | 1.24 | 1.04 | 0.95 | **0.90** | 117.77 |
| Type II | | | 1.70 | 1.37 | 1.14 | 1.06 | **0.93** | 137.10 |
| Type III | | | 1.66 | 1.35 | 1.03 | 0.94 | **0.78** | 151.14 |
| All | | | 1.67 | 1.32 | 1.07 | 0.98 | **0.87** | 135.34 |

Mean percentage deviations from knapsack upper bound.

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=**6.09%**) 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**).

Table 5: Computational results - zero-waste problems.

| Source of problem | Inst. | Dim. (L x W) | m | M | Opt. | GRASP | TABU | MPGA | CPU Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| Lai and Chan (997a) | 1 | 400x200 | 9 | 10 | 80000 | **80000** | **80000** | **80000** | 0.03 |
| | 2 | 400x200 | 7 | 15 | 79000 | **79000** | **79000** | **79000** | 0.02 |
| | 3 | 400x200 | 5 | 20 | 160000 | 154600 | **160000** | **160000** | 2.30 |
| Jakobs (1996) | 1 | 70x80 | 14 | 20 | 5600 | 5447 | **5600** | **5600** | 0.57 |
| | 2 | 70x80 | 16 | 25 | 5600 | 5455 | **5540** | **5540** | 78.58 |
| | 3 | 120x45 | 22 | 25 | 5400 | 5328 | **5400** | **5400** | 3.08 |
| | 4 | 90x45 | 16 | 30 | 4050 | 3978 | **4050** | **4050** | 8.22 |
| | 5 | 65x45 | 18 | 30 | 2925 | 2871 | **2925** | **2925** | 6.13 |
| Leung et al. (2003) | 1 | 150x110 | 40 | 40 | 16500 | 15856 | **16280** | 16188 | 240.13 |
| | 2 | 160x120 | 50 | 50 | 19200 | 18628 | 19044 | **19056** | 302.11 |
| Hopper and Turton (2001) | 1-1 | 20x20 | 16 | 16 | 400 | **400** | **400** | **400** | 0.09 |
| | 1-2 | 20x20 | 17 | 17 | 400 | 386 | **400** | **400** | 5.66 |
| | 1-3 | 20x20 | 16 | 16 | 400 | **400** | **400** | **400** | 0.03 |
| | 2-1 | 40x15 | 25 | 25 | 600 | 590 | **600** | **600** | 5.05 |
| | 2-2 | 40x15 | 25 | 25 | 600 | 597 | **600** | **600** | 50.90 |
| | 2-3 | 40x15 | 25 | 25 | 600 | **600** | **600** | **600** | 0.86 |
| | 3-1 | 60x30 | 28 | 28 | 1800 | 1765 | **1800** | **1800** | 2.72 |
| | 3-2 | 60x30 | 29 | 29 | 1800 | 1755 | **1800** | **1800** | 5.88 |
| | 3-3 | 60x30 | 28 | 28 | 1800 | 1774 | **1800** | **1800** | 2.45 |
| | 4-1 | 60x60 | 49 | 49 | 3600 | 3528 | **3580** | **3580** | 323.01 |
| | 4-2 | 60x60 | 49 | 49 | 3600 | 3524 | 3564 | **3588** | 74.54 |
| | 4-3 | 60x60 | 49 | 49 | 3600 | 3544 | 3580 | **3592** | 22.51 |
| | 5-1 | 60x90 | 73 | 73 | 5400 | 5308 | 5342 | **5388** | 704.65 |
| | 5-2 | 60x90 | 73 | 73 | 5400 | 5313 | 5361 | **5400** | 770.82 |
| | 5-3 | 60x90 | 73 | 73 | 5400 | 5312 | 5375 | **5388** | 687.31 |
| | 6-1 | 80x120 | 97 | 97 | 9600 | 9470 | 9548 | **9558** | 1940.96 |
| | 6-2 | 80x120 | 97 | 97 | 9600 | 9453 | 9448 | **9588** | 1097.82 |
| | 6-3 | 80x120 | 97 | 97 | 9600 | 9450 | 9565 | **9584** | 917.56 |
| | 7-1 | 160x240 | 196 | 196 | 38400 | 37661 | 38026 | **38107** | 6579.25 |
| | 7-2 | 160x240 | 197 | 197 | 38400 | 37939 | 38145 | **38345** | 2170.29 |
| | 7-3 | 160x240 | 196 | 196 | 38400 | 37745 | 37867 | **38230** | 5356.24 |
| Mean Percentage deviation from optimum | | | | | | 1.68 % | 0.42 % | **0.24 %** | |

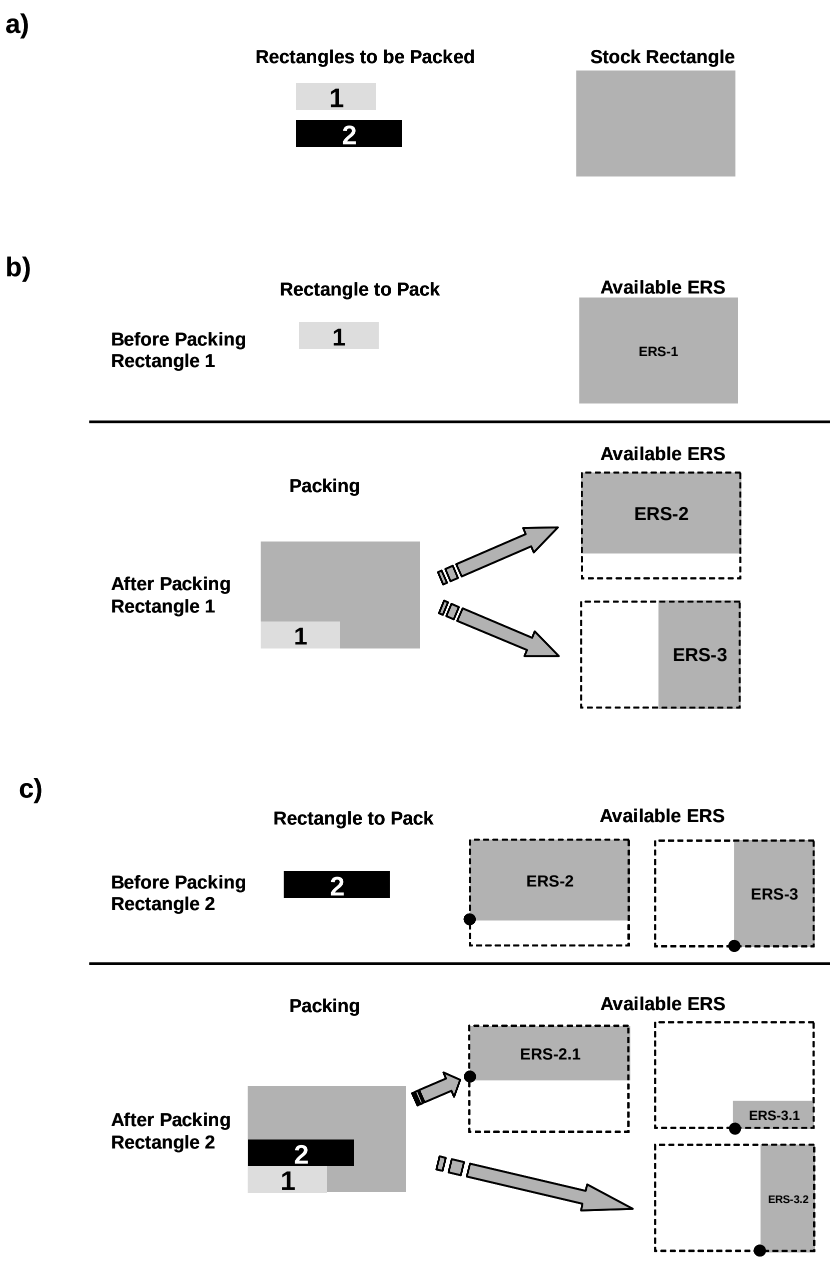


Figure 13: Example of Difference Process (DP).

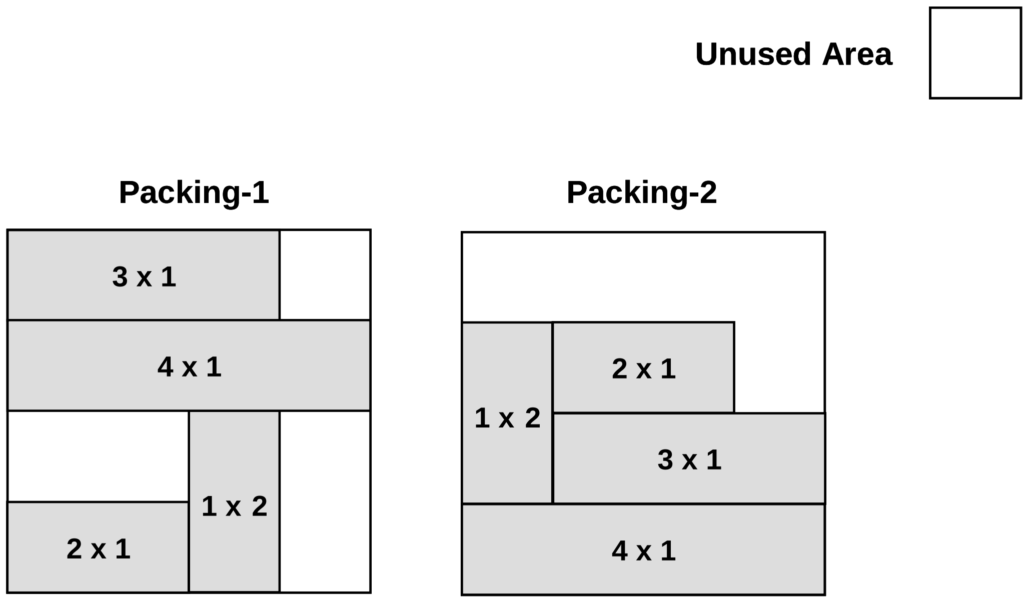


Figure 14: Two packings with different improvement potentials.

Table 6: Computational results - doubly constrained problems.

| Source of problem | Inst. | Dim. (L x W) | m | M | UB | PH | GRASP | TABU | MPGA | CPU time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| Beasley (1985) | 1 | 10x10 | 5 | 10 | 164 | **164** | **164** | **164** | **164** | 0.01 |
| | 2 | 10x10 | 7 | 17 | 230 | **225** | **225** | **225** | **225** | 0.02 |
| | 3 | 10x10 | 10 | 21 | 247 | **220** | **220** | **220** | **220** | 0.08 |
| | 4 | 15x10 | 5 | 7 | 268 | **268** | **268** | **268** | **268** | 0.00 |
| | 5 | 15x10 | 7 | 14 | 358 | **301** | **301** | **301** | **301** | 0.01 |
| | 6 | 15x10 | 10 | 15 | 289 | **265** | 252 | **265** | **265** | 0.62 |
| | 7 | 20x20 | 5 | 8 | 430 | **430** | **430** | **430** | **430** | 0.01 |
| | 8 | 20x20 | 7 | 13 | 834 | **819** | **819** | **819** | **819** | 0.21 |
| | 9 | 20x20 | 10 | 18 | 924 | **924** | **924** | **924** | **924** | 0.51 |
| | 10 | 30x30 | 5 | 13 | n/f | n/f | n/f | n/f | n/f | 9.55 |
| | 11 | 30x30 | 7 | 15 | 1688 | 1505 | **1518** | **1518** | **1518** | 3.19 |
| | 12 | 30x30 | 10 | 22 | 1865 | 1666 | 1648 | **1672** | **1672** | 0.29 |
| Hadjiconstantinou and Christofides (1995) | 3 | 30x30 | 7 | 7 | 1178 | **1178** | **1178** | **1178** | **1178** | 0.00 |
| | 11 | 30x30 | 15 | 15 | 1270 | **1216** | **1216** | **1216** | **1216** | 0.02 |
| Wang (1983) | | 70x40 | 19 | 42 | 2726 | 2499 | 2700 | **2716** | **2716** | 4.10 |
| Christofides and Whitlock (1977) | 3 | 40x70 | 20 | 62 | 1860 | 1600 | **1720** | **1720** | **1720** | 1.58 |
| Fekete and Schepers (1997a) | 1 | 100x100 | 15 | 50 | 27718 | 25373 | 24869 | **25384** | **25384** | 36.46 |
| | 2 | 100x100 | 30 | 30 | 22502 | 17789 | 19083 | 19657 | **20678** | 20.40 |
| | 3 | 100x100 | 30 | 30 | n/f | n/f | n/f | n/f | n/f | 72.35 |
| | 4 | 100x100 | 33 | 61 | 32893 | 27556 | 27898 | **28974** | **28974** | 10.70 |
| | 5 | 100x100 | 29 | 97 | 27923 | 21997 | 22011 | 22011 | **23565** | 51.16 |
| Mean Percentage deviation from upper bound | | | | | | 8.11 | 7.36 | 6.62 | **6.09** | |

## 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.

## Acknowledgments

This work has been supported by funds granted by Fundação para a Ciência e Tecnologia (FCT) project PTDC/GES/7224/2006.

## References

Alvarez-Valdes, R., F. Parreño, and J. Tamarit: 2005, ‘A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems’. *Journal of Operational Research Society* **56**, 414–425.

Alvarez-Valdes, R., F. Parreño, and J. Tamarit: 2007, ‘A tabu search algorithm for a two-dimensional non-guillotine cutting problem’. *European Journal of Operational Research* **183**, 1167–1182.

Amaral, A. and A. Letchford: 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. and R. Morabito: 1995, ‘An and/or-graph approach to the solution of two dimensional guillotine cutting problems’. *European Journal of Operational Research* **84**, 599–617.

Bean, J.: 1994, ‘Genetics and random keys for sequencing and optimization’. *ORSA Journal on Computing* **6**, 154–160.

Beasley, D., D. Bull, and R. Martin: 1993, ‘An overview of genetic algorithms: Part 1, Fundamentals’. *University Computing* **15**, 58–69.

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

Beasley, J.: 2004, ‘A population heuristic for constrained two-dimensional non-guillotine cutting’. *European Journal of Operational Research* **156**, 601–627.

Boschetti, M., E. Hadjiconstantinou, and A. Mingozzi: 2002, 'New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem'. *IMA Journal of Management Mathematics* **13**, 95–119.

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

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

Caprara, A. and M. Monaci: 2004, 'On the 2-dimensional knapsack problem'. *Operations Research Letters* **32**, 5–14.

Christofides, N. and C. Whitlock: 1977, 'An algorithm for two dimensional cutting problems'. *Operations Research* **25**, 31–44.

Darwin, C.: 1859, *On the origin of species through natural selection.* London: John Murray.

Dowsland, K. and W. Dowsland: 1992, 'Packing problems'. *European Journal of Operational Research* **56**, 2–14.

Dyckhoff, H.: 1990, 'A typology of cutting and packing problems'. *European Journal of Operational Research* **44**, 145–159.

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

Fekete, S. and J. Schepers: 1997a, 'A new exact algorithm for general orthogonal *d*-dimensional knapsack problems'. In: *Algorithms - ESA '97*, Vol. 1284 of *Springer Lecture Notes in Computer Science.* pp. 144–156.

Fekete, S. and J. Schepers: 1997b, 'On higher-dimensional packing I: Modeling'. Technical Report ZPR 97-288, Mathematisches Institut, Universitat zu Köln.

Fekete, S. and J. Schepers: 1997c, 'On higher-dimensional packing II: Bounds'. Technical Report ZPR97-289, Mathematisches Institut, Universitat zu Köln.

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

Fekete, S. and J. Schepers: 2004b, 'A general framework for bounds for higher-dimensional orthogonal packing problems'. *Mathematical Methods of Operations Research* **60**, 311–329.

Fekete, S. and J. Schepers: 2004c, 'An exact algorithm for higher-dimensional orthogonal packing'. Working paper. Available online at `http://www.math.tu-bs.de/~fekete`.

Feo, T. and M. Resende: 1989, 'A probabilistic heuristic for a computationally difficult set covering problem'. *Operations Research Letters* **8**, 67–71.

Feo, T. and M. Resende: 1995, 'Greedy randomized adaptive search procedures'. *Journal of Global Optimization* **6**, 109–133.

Garey, M. and D. Johnson: 1979, *Computers and intractability: A guide to the theory of NP-completeness.* New York: W.H. Freeman and Company.

Goldberg, D.: 1989, *Genetic algorithms in search optimization and machine learning.* Addison-Wesley.

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

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

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

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

Hadjiconstantinou, E. and N. Christofides: 1995, 'An exact algorithm for general, orthogonal, two dimensional knapsack problems'. *European Journal of Operations Research* **83**, 39–56.

Hadjiconstantinou, E. and M. Iori: 2007, 'A hybrid genetic algorithm for the two-dimensional knapsack problem'. *European Journal of Operational Research* **183**, 1150–1166.

Haessler, R. and P. Sweeney: 1991, 'Cutting stock problems and solution procedures'. *European Journal of Operational Research* **54**, 141–150.

Healy, P., M. Creavin, and A. Kuusik: 1999, 'An optimal algorithm for rectangle placement'. *Operations Research Letters* **24**, 73–80.

Hopper, E. and B. Turton: 2001, 'An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem'. *European Journal of Operational Research* **128**, 34–57.

Jakobs, S.: 1996, 'On genetic algorithms for the packing of polygons'. *European Journal of Operational Research* **88**, 165–181.

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

Lai, K. and J. Chan: 1997b, 'An evolutionary algorithm for the rectangular cutting stock problem'. *International Journal of Industrial Engineering* **4**, 130–139.

Leung, T., C. Chan, and M. Troutt: 2001, 'Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem'. *Computers and Industrial Engineering* **40**, 201–214.

Leung, T., C. Chan, and M. Troutt: 2003, 'Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem'. *European Journal of Operational Research* **141**, 241–252.

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

Scheithauer, G. and J. Terno: 1993, 'Modeling of packing problems'. *Optimization* **28**, 63–84.

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

Sweeney, P. and E. Paternoster: 1992, 'Cutting and packing problems: A categorized, application-orientated research bibliography'. *Journal of the Operational Research Society* **43**, 691–706.

Tsai, R., E. Malstrom, and H. Meeks: 1988, 'A two-dimensional palletizing procedure for warehouse loading operations'. *IIE Transactions* **20**, 418–425.

Wang, P.: 1983, 'Two algorithms for constrained two-dimensional cutting stock problems'. *Operations Research* **31**, 573–586.

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