GRASP and Path Relinking for the Two-dimensional Two-staged Cutting Stock Problem

Ramón Alvarez-Valdes[1], Rafael Martí[1], Antonio Parajón[2] and Jose M. Tamarit[1]

[1] Departamento de Estadística e I.O. Universidad de Valencia. Dr. Moliner 50, 46100 Burjassot, Valencia. Spain.

[2] Departamento de Matemáticas. Universidad Nacional Autónoma de Nicaragua, UNAN-Managua. ENEL Central 2 Km Sur, Managua. Nicaragua.

ABSTRACT

In this paper, we develop a greedy randomized adaptive search procedure (GRASP) for the constrained two-dimensional two-staged cutting stock problem. This is a special cutting problem in which the cut is performed in two phases. In the first phase, the stock rectangle is slit down its width into different vertical strips and in the second phase, each of these strips is processed to obtain the final pieces. We propose two different algorithms based on GRASP methodology. One is "piece oriented" while the other is "strip oriented". Both procedures are fast and provide solutions of different structures to this cutting problem. We also propose a path relinking algorithm, which operates on a set of elite solutions obtained with both GRASP methods, to search for improved outcomes. We perform extensive computational experiments with well-known instances which have been previously reported, first to study the effect of changes in critical search parameters and then to compare the efficiency of alternative solution procedures. The experiments establish the effectiveness of our procedure in relation to approaches previously identified as best, especially in large-scale instances.

Key Words: Cutting, Packing, Heuristics, GRASP, Path Relinking.

1. Introduction

The two-dimensional cutting problem (TDC) consists of cutting a stock rectangle S into rectangular pieces of n different types. The dimensions of S are LxW and of piece i are $l_i x w_i$. The number of appearances of piece i in the cutting is limited by its demand b_i. If $b_i = \lfloor LW/l_i w_i \rfloor$ (a trivial bound), for all i, the problem is *unconstrained*. If, for any piece i, $b_i < \lfloor LW/l_i w_i \rfloor$, the problem is *constrained*. Each piece has a value c_i and the objective is to maximize the total values of pieces cut. If $c_i = l_i w_i$, for all i, maximizing the value is equivalent to minimizing the non-used area of S. In this case, the problem is *un-weighted*. If, for any piece i, $c_i \neq l_i w_i$, reflecting some other piece characteristics besides its area, the problem is *weighted*. In this paper we develop algorithms for the more general weighted constrained TDC problem from which the other versions can be considered particular cases.

We also impose some additional conditions on the TDC problem. Specifically, we consider that pieces have a fixed orientation, that is, a piece of dimensions *lxw* is not the same as a piece of dimensions *wxl*. We only use *guillotine cuts*, that is -cuts going from one edge of the current rectangle to the opposite edge. We also limit the number of stages in the cutting process to two. In the two-staged two-dimensional cutting problem (2-TDC), the stock rectangle S is first cut into a set of horizontal (vertical) strips and then in the second stage these strips are cut vertically (horizontally) into the required pieces. We will allow the *trimming* of the strips, i.e., supplementary horizontal (vertical) cuts within each rectangle obtained in the second stage to produce the pieces. This version of the problem is called the *non-exact case*, to distinguish it from the *exact case* in which trimming is not allowed. Note that trimming introduces flexibility in the cutting of the strips (first stage) since pieces of different widths can be allocated in the same strip.

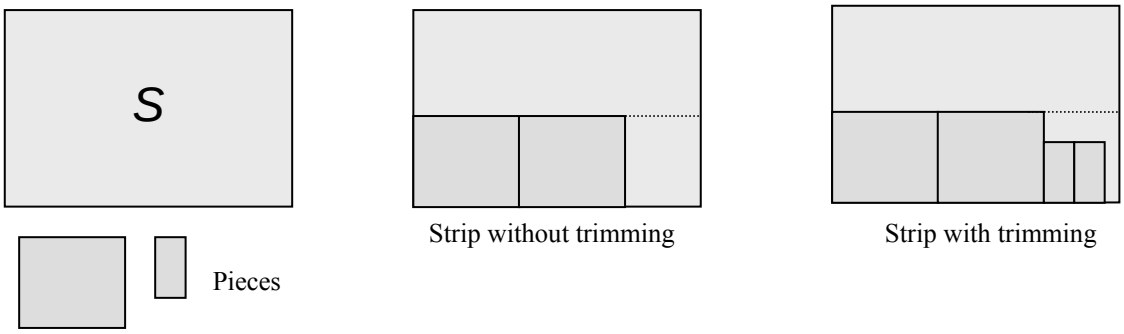


Figure 1. Example of trimming

Figure 1 shows a stock rectangle S (on the left hand side) in which we want to cut the two pieces shown below it. We show a strip without trimming (in the center of the figure) and another with trimming (on the right hand side). This figure clearly shows that if no trimming is allowed we can obtain a relatively large amount of waste in the strip; while trimming permits the fitting of smaller pieces to complete the strip in a more efficient way.

The cutting stock problem has generated a considerable amount of research interest over the years, as documented in Dyckhoff (1990) or Sweeney and Paternoster (1992). It has many practical applications in the paper, textile and other industries. In particular, 2-TDC problems are very common in some applications, as in the timber industry, in which the structure of the saws and other handling considerations impose a cutting process in two-stages (Morabito and Garcia (1998)). The 2-TDC problem had already been considered by Gilmore and Gomory (1961, 1965) in their seminal work on cutting problems, where an exact dynamic programming approach was proposed. More recently, exact and approximate solution procedures have been developed. Beasley (1985) proposes a dynamic programming procedure. Hifi (2001) and Hifi and Roucairol (2001) develop branch and bound algorithms, while Lodi and Monaci (2003) propose integer linear programming formulations. Belov and Scheithauer (2003) combine both approaches into a branch and cut and price algorithm. Heuristic procedures appear in the work by Beasley (1985), who proposes heuristically reducing the state-space of his dynamic programming procedure, and Hifi and Roucairol (2001) who solve a series of bounded knapsack problems for the strips and combine the solutions into a complete solution for the sheet by solving a packing problem. More recently, Hifi and M'Hallah (2003) have proposed some new procedures in which they improve the Hifi and Roucairol algorithm and combine it with local search methods.

In this paper we develop a heuristic solution procedure for two-staged two-dimensional cutting problems. Our solution procedure is based on constructing, improving and then combining solutions within the framework of GRASP methodology as well as the evolutionary approach known as Path Relinking. The algorithm is designed to solve the more complex, non-exact problem, but it can be easily adapted to the exact case. Section 2 presents the Bottom-Left Procedure, a constructive algorithm which will be used as a building block in the more complex algorithms proposed in the following sections. Section 3 describes two GRASP algorithms, one based on pieces and the other based on strips. Section 4 explores a combination of solutions according to the Path Relinking methodology. Finally, Section 5 contains the computational results and the paper ends with the associated conclusions.

2. Bottom-Left Procedures

The Bottom-Left procedure (BLP) is a simple and well known method for allocating a set of pieces in a stock rectangle. Basically, the method consists of allocating the pieces in the rectangle following a pre-established order. The first piece is positioned in the bottom-left corner of the rectangle. At each iteration, the next piece in the ordering is selected and moved down and left as far as possible. Then, for each initial ordering of the pieces, the algorithm produces a solution. We now propose some variants of the BLP in order to use them as a baseline for comparison in our experiments, as well as to form part of more complex methods.

In our implementation, we order the pieces by non-increasing width w_i. In the first iteration, we select the piece with largest w-value, decrease its demand b by one unit, and allocate it in the bottom-left corner of the rectangle. Note that considering the 2-stage constraint of our problem, the width of this first piece defines the width of the first strip.

In the second iteration, we select the first piece in the ordering with a positive demand, decrease its demand by one unit, and allocate it in the first strip, next to the previously allocated piece. Note that the width of the current piece is lower than or equal to the width of the previous-iteration piece; therefore it is lower than or equal to the width of the strip. We continue in this way until the length l of the selected piece exceeds the remaining length of the strip r_s (the difference between L and the sum of the lengths of the pieces added in previous iterations to this strip). The new piece is then allocated in a new strip. The algorithm terminates when all the demands have been satisfied or there is no room for a new strip.

Figure 2 shows an example in which the BLP is constructing a solution. In the first iteration, piece 3 has been selected and cut (bottom-left corner). If we consider that b_3=2, then in the second iteration, we select the second unit of piece 3 and put it in the same strip, next to the previous unit. Suppose that the next piece in the ordering is number 7 and b_7=3, so in iteration 3, a unit of this piece is allocated in strip 1. In iteration 4, the second unit of piece 7 is selected, but there is no room for it in the first strip, so we allocate this unit in a second strip. We would continue in this way with the rest of the pieces.

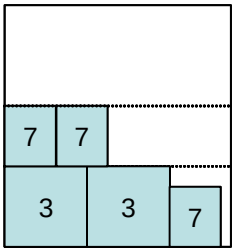


Figure 2. BLP example

It should be noted that the BLP starts a new strip each time that the next piece in the ordering does not fit in the current strip (its length is larger than the remaining strip's length r_s). However, another piece i in a posterior position in the ordering could fit in the current strip if $l_i \leq r_s$. Next, before starting a new strip, we will resort to the REMAIN procedure in order to fill this gap. This procedure searches for the first piece i in the ordering with positive demand b_i, satisfying $l_i \leq r_s$ and allocates it in the strip. Then, b_i and r_s are updated (b_i=b_i-1, r_s=r_s-l_i) and the method continues to fill the remaining gap until no piece i satisfying $l_i \leq r_s$ is found.

A pseudo-code of the BLP to produce a solution to the problem appears in Figure 3. The pseudo-code is written using general mathematical notation. However, the actual implementation takes advantage of efficient data structures and quick updating mechanisms that are hard to represent mathematically. It also implements a tie breaking rule for piece selection that is based on the cost/surface ratio. The rule is used when more than one piece has the same width and there is a tie in the ordering of the pieces. The tie is then resolved according to the ratio, where the piece with the highest ratio enters first.

Initialization
Order the pieces according to w_i (resolve ties according to $c_i / (l_i x w_i)$)
Create the first strip in the bottom of the stock rectangle
Add the first piece i in the ordering to the strip. $b_i = b_i - 1$
Let *Value* be the objective function value of the solution. $Value = c_i$
Let *Wfree* be the rectangle's width without strips: $Wfree = W - w_i$
$r_s = L - l_i$

While (A piece i with $b_i > 0$ exists)
{
  Let i be the next piece in the ordering with $b_i > 0$
  If($l_i \leq r_s$)
    Add piece i to the strip. $b_i = b_i - 1$
    $Value = Value + c_i$
    $r_s = r_s - l_i$
  Else
    Call REMAIN procedure
    Let i be the first piece in the ordering with $b_i > 0$ and $w_i \leq Wfree$
    If(Such a piece i exists)
      Create new strip
      $Wfree = Wfree - w_i$
      Add piece i to begin the strip. $b_i = b_i - 1$
      $Value = Value + c_i$
      $r_s = L - l_i$
    Else
      STOP
}

REMAIN procedure
{
  R={ pieces j / $b_j > 0$, $l_j \leq r_s$ }
  While ($R \neq \phi$)
    Select the first piece i in the ordering / $i \in R$
    Add i to the strip
    $b_i = b_i - 1$. $r_s = r_s - l_i$
    $Value = Value + c_i$
    $R = R - \{i\}$
}

Figure 3. Pseudo-code of the Bottom-Left Procedure

This BLP method is based on the fact that the initial ordering of pieces, according to w, guarantees that the piece selected in an iteration can be allocated within the strip created with a previously selected piece. However, with a minor modification, we can adapt this method to construct a solution from any initial ordering. We just need to replace the condition " ($l_i \leq r_s$) " in the outer *If* statement with the condition "($l_i \leq r_s$ and $w_i \leq Wstrip$)", where *Wstrip* is the width of the strip, defined as the width of the first piece in the strip. Now, when we select the next piece in the ordering, if it cannot be allocated in the current strip under construction (because of its length or its width), we start a new strip with this piece, but first we resort to the REMAIN routine to complete the current strip. We will refer to this general BLP method as GBLP.

The GBLP method constructs a solution from a given initial ordering. So, a simple optimization scheme consists of performing several iterations with GBLP, each one from a random initial ordering. We can improve this basic scheme using a "context-independent" optimizer to generate the initial orderings.

Specifically, we have considered the OptQuest Callable Library[1] (OCL) in order to search the space of the orderings in a more efficient way than the random sampling. OCL is a generic optimizer that overcomes the deficiency of black box systems and successfully embodies the principle of separating the solver method from the optimization model (Laguna and Martí, 2002). OCL uses the value of the GBLP output to measure the merit of the initial ordering. On the basis of both current and past evaluations, the optimization procedure within OCL generates a new permutation. The optimization procedure is designed to carry out a special "strategic search," based on the scatter search methodology, where successively generated orderings produce varying evaluations, not all of them improving, but which over time provide a highly efficient trajectory to the best solutions.

In the computational section we will show the results obtained with these three variants of bottom-left procedures; the original BLP, the GBLP from random initial orderings, and the combination of GBLP and OCL.

3. GRASP

GRASP, *greedy randomized adaptive search procedure*, is a multi-start or iterative process in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is explored until a local optimum is found after the application of the local search phase. The best local optimum is reported as the best overall solution found. Performing multiple GRASP iterations may be interpreted as a means of strategically sampling the solution space. Based on empirical observations, it has been found that the sampling distribution generally has a mean value that is inferior to the one obtained by a deterministic construction, but the best solution over all trials dominates the deterministic solution with a high probability. Resende and Ribeiro (2001) present a comprehensive review of GRASP and an extensive survey of GRASP literature can be found in Festa and Resende (2001).

The construction phase plays a critical role with respect to providing high-quality starting solutions for the local search. In this section, we propose two different iterative methods to construct solutions. In each iteration, the first method adds a piece to the solution, while the second one adds an entire strip. As will be shown in the computational section, these two methods provide solutions with different structures.

At each iteration of the construction phase, GRASP maintains a set of candidate elements that can be feasibly added to the partial solution under construction. All candidate elements are evaluated according to a greedy function in order to select the next element to be added to the construction. The greedy function represents the marginal increase in the cost function from adding the element to the partial solution. The evaluation of the elements is used to create a restricted candidate list (RCL). RCL consists of the best elements, i.e. those with the smallest incremental cost. This is the greedy aspect of the method. The element to be added into the partial solution is randomly selected from those in the RCL. This is the probabilistic aspect of the heuristic. Once the selected element is added to the partial solution, the candidate list is updated and the incremental costs are recalculated. This is the adaptive aspect of the heuristic.

The solutions generated by a greedy randomized construction are not necessarily optimal, even with respect to simple neighborhoods. The local search phase usually improves upon the constructed solution and terminates when no better solution is found in the neighborhood. Since the constructive methods obtain solutions of different characteristics, we propose a different local search method for each constructive procedure. The first one is based on a piece replacement while the other tries to replace a strip. The following sub-sections describe in detail our GRASP implementations for the TDC problem.

[1] OptQuest is a registered trademark of OptTek Systems, Inc. (www.opttek.com).

3.1 A GRASP based on Pieces

Construction Phase

In the construction phase, we distinguish two stages. The first one selects a suitable width to create a new strip, while the second one fills an existing strip with pieces. Each time a strip is completed, we resort to the first stage to select the width of a new strip.

In the first stage, we consider the values w of the widths of all pieces. Let *Wfree* be the rectangle's width with no strips. Initially, *Wfree* is equal to W, but when a strip of width w is created, then *Wfree* = *Wfree* – w. Let CL_1 be the candidate list of widths in the first stage, which includes all the widths $w \leq$ *Wfree*.

For each width $w \in CL_1$, we define the set of pieces P_w with a width lower than or equal to w. We calculate *value(w)* as a measure of the attractiveness of this width to create a new strip (of w width).

$$value(w) = \frac{\sum_{i \in P_w} c_i b_i}{\sum_{i \in P_w} b_i}$$

The largest attractiveness value of all the widths in CL_1 is multiplied by the α parameter. This final value represents a threshold that is used to build the RCL_1 of the first stage. In particular, RCL_1 consists of all the widths in CL_1 whose attractiveness measure is at least as large as the threshold value. The procedure randomly selects the next width to construct a strip from the RCL_1. Let w^* be the selected width. We fill the strip now, in stage two, with pieces from P_{w^*}.

As in the BLP method, we define the remaining length r_s of the strip as the difference between L and the sum of the lengths of the pieces added in previous iterations to this strip. When we initiate stage two, r_s is equal to L. Then the candidate list in stage two, CL_2, consists of all the pieces in P_{w^*} with a positive demand and a length lower than or equal to r_s.

For each piece $i \in CL_2$, we consider its cost c_i as a measure of its attractiveness. The largest attractiveness value of all the pieces in CL_2 is multiplied by the α parameter. RCL_2 consists of all pieces in CL_2 whose cost is at least as large as the threshold value. The procedure randomly selects the next piece i to be allocated in the current strip from the RCL_2. After this assignment, the demand of i and the remaining length of the strip s are updated (b_i=b_i-1, r_s=r_s-l_i). Then CL_2 and RCL_2 are computed and another piece is selected for assignment. Stage-two continues to fill the remaining strip until no piece i satisfying $l_i \leq r_s$ is found. Then we create a new strip with the stage one method described above. The procedure terminates when no new strip can be created.

Improvement Phase

Each step of the improvement phase consists of selecting each piece to be considered for a move. We scan the pieces in a solution in the order given by the strips. We start with the first strip (the bottom one) and continue up to the last one (the "top" strip). In each strip, we select each piece and try to replace it with another piece with positive demand. The move value is the difference between the cost of the removed piece and the new one. We perform the move with maximum positive value. If no improvement is possible, then the piece is not moved. Note that when piece i is removed from the strip, the remaining length becomes r_s+l_i. We then find the piece for insertion in the move by scanning all the pieces with a positive demand and a length lower than or equal to r_s+l_i.

An improvement step terminates when all pieces have been considered for replacement, with the exception of the last strip, which is re-computed from scratch in a greedy fashion (according to the cost values). More steps are performed as long as at least one piece is replaced (i.e., as long as the current solution keeps improving).

We have considered an extended version of the improvement phase. In this variant, when a move fails to find a piece for replacement, we try to replace one piece with two pieces. Although this move is more time consuming than the simple one, we only resort to this "one-to-two" move if the "one-to-one" move

fails. Note that piece *i* can be replaced with two pieces if the sum of their lengths is lower than or equal to r_s+l_i . As in the simple move, we only perform positive moves.

As mentioned earlier, GRASP consists of a construction phase and an improvement phase. The method alternates both phases until a maximum number of iterations is reached. The value of α is a search parameter. In Section 5, we perform a set of experiments to test the effect of different α values on solution quality, speed, and the number of iterations. We will refer to this method as GRASP_Piece for the initial improvement phase, and as GRASP_PieceExt for the extension with the "one-to-two" movements.

3.2 A GRASP based on Strips

Construction Phase

We can build efficient strips by solving a series of bounded knapsack problems. For a given width *w*, the best strip of length *L* and width *w* is given by the optimal solution of the following problem:

$$KP_{(L,w)} = \begin{cases} z_w^* = \max \sum_{i \in P_w} c_i x_i \\ s.t. \quad \sum_{i \in P_w} l_i x_i \le L \\ \qquad x_i \le b_i, \;\; x_i \text{ integer}, i \in P_w \end{cases}$$

where $P_w = \left\{ j \middle| w_j \le w \right\}$

This idea is by no means new. It has been used by several authors on several cutting problems. In particular, for the problem considered here, Hifi and Roucairol (2001) propose algorithms in which the first phase consists of solving a series of knapsack problems, one for each different piece width. The strips obtained in the first phase are then combined into a complete solution by solving a packing problem. We propose a different strategy. Instead of solving the knapsack problems just once, taking the original demands as upper bounds, we develop an iterative procedure in which at each iteration we solve one knapsack problem for each width with positive residual demand of its associated pieces. The best strip obtained is added to the partial solution and the residual demands are updated. The process finishes when all the pieces have been cut or no new strip can be fitted into the sheet. A pseudo-code of the procedure appears in Figure 3. We assume that among the *n* piece types there are *r* different widths, $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_r$.

Initialization
Let *Value* be the objective function value of the solution: *Value*=0
Let *Wfree* be the rectangle's width without strips: *Wfree = W*
Let M = $\{ \overline{w}_k \mid$ a piece *i* with $w_i = \overline{w}_k$ and $b_i>0$ exists$\}$
Order M by decreasing $\overline{w}_k$
While(M $\neq \varnothing$) {
  Let $z_{\max}^* = 0;\; x_{\max}^* = 0;\; w_{\max}^* = 0.$
  For each $\overline{w}_k \in$ M {
    Solve the problem $KP_{(L,\overline{w}_k)}$. Let $z_{\overline{w}_k}^*$ the optimal value and $x_{\overline{w}_k}^*$ the optimal solution
    If $z_{\max}^* < z_{\overline{w}_k}^*$, then $z_{\max}^* = z_{\overline{w}_k}^*;\; x_{\max}^* = x_{\overline{w}_k}^*;\; w_{\max}^* = \overline{w}_k.$
  }
  If $(z_{\max}^* \neq 0)$
    $Value = Value + z_{\max}^*$, $b = b - x_{\max}^*$, $Wfree = Wfree - w_{\max}^*$
    If, for some $\overline{w}_k$, all pieces *i* with $w_i = \overline{w}_k$ have $b_i = 0$, then M = M - $\{\overline{w}_k\}$
    If, for some $\overline{w}_k$, $\overline{w}_k > Wfree$, then M = M - $\{\overline{w}_k\}$
  Else
    STOP
}

Figure 4. Pseudo-code of the Constructive Phase of the GRASP based on Strips

Note that at each iteration the procedure selects the strip with maximum absolute value of the knapsack problems. An alternative way of selecting the strip to be added to the partial solution could be based on the maximum relative value: $z^*_{\bar{w}_k} / \bar{w}_k$. The solution obtained by this procedure consists of a list of strips, in the order of inclusion in the sheet plus, maybe, an empty strip of waste at the end of the list.

The basic procedure described in Figure 4 can be enhanced in two ways. First, the number of knapsack problems required to be solved can be reduced. Consider, for instance, the first iteration in which *r* problems, one for each $\bar{w}_k$ are solved, but only one strip is added to the solution. Therefore, updating the demands does not modify some of these demands substantially. At the second iteration, when we consider the problem associated to a width $\bar{w}_k$, if the optimal solution of that problem in the previous iteration $x^*_{\bar{w}_k}$ is lower than or equal to the residual demand *b*, the problem does not need to be solved. A second way of improving the procedure is based on the following argument. If we are considering the problem associated to a width $\bar{w}_k$, but $\bar{w}_k > Wfree - \min\{\bar{w}_i \mid \bar{w}_i \in \mathsf{M}\}$, no other strip would fit into the stock rectangle if a strip of width $\bar{w}_k$ is added. Therefore, when updating M, all widths $\bar{w}_k$ such that $\bar{w}_k + \min\{\bar{w}_i \mid \bar{w}_i \in \mathsf{M}\} > Wfree$ can be replaced in M by a unique value *Wfree.*

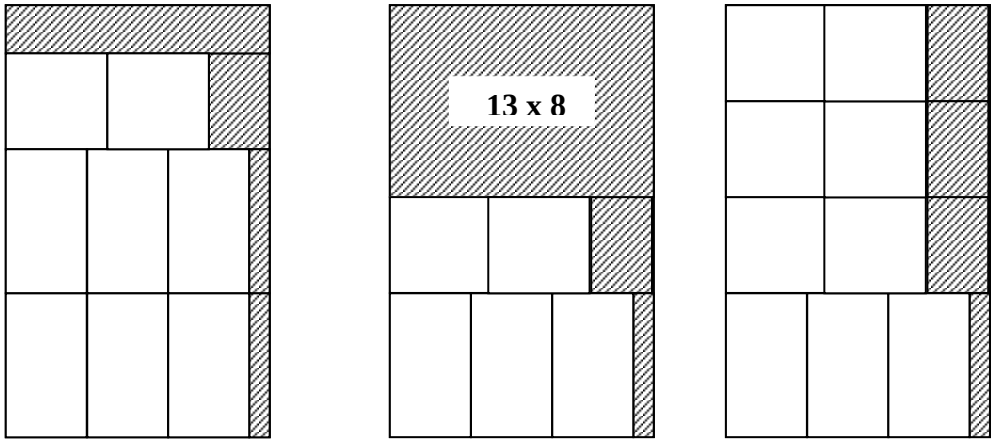
This deterministic procedure is used in the GRASP algorithm in a similar way to that described in Section 3.1. The candidate list at each iteration *k*, CL_k, contains all the widths currently in M. The restricted candidate list $RCL_k = \left\{\bar{w}_k \in \mathsf{M} \mid z^*_{\bar{w}_k} \geq \alpha z^*_{\max}\right\}$. From RCL_k a width is selected at random and the corresponding strip is added to the partial solution.

Improvement Phase

At each step of an improvement move we select a strip to be removed from the current solution. The emptied space, merged with the strip of waste if it exists, is filled again by applying the deterministic constructive algorithm (BLP) on this new strip with the residual demands updated. Note that the removed strip can be replaced by one or more new strips.

This improvement move is used in a standard local search procedure. Given a solution, we consider every one of its strips, one at a time, and study the associated move. The solution obtained by the move producing the largest improvement is taken as the new solution and the procedure starts from it again. If no improvement is found, the local search stops. We will refer to this method in which both phases are repeated until a stopping criteria is met as GRASP_Strip.

In Figure 5 we show an example of the construction and improvement of a solution. Let us suppose we are given a stock sheet (13 x 18) from which we have to cut two types of pieces A (4 x 6) and B (5 x 4) with demands 6 for both types. In the constructive phase for selecting the first strip we solve two knapsack problems $KP_{(13,6)}$ with solution value 72 and relative solution value 12, and $KP_{(13,4)}$ with solution value 40 and relative solution value 10. We choose the first strip of width 6, update *Wfree*=12 and $d_A = 3$. For the second strip we would have to solve the same two knapsacks again, but the solutions obtained in the previous iteration are still valid and we use them again to choose the strip one, updating *Wfree*=6 and $d_A = 0$. In the third iteration there is just one possible strip of width 4 and the complete solution with value 184 appears in Figure 5 (a). In the improvement phase, we eliminate the first strip, obtaining the situation in Figure 5 (b) in which a region of (13 x 8) has to be filled with demands d_A=3 and d_B=4, but the strip recently eliminated is not considered for inclusion again. Therefore, we add a strip of width 4 twice, producing the improved solution of value 192 which appears in Figure 5 (c).



(a) (b) (c)

Figure 5. Improvement move

4. Path Relinking

Path relinking (PR) was originally suggested as an approach to integrate intensification and diversification strategies in the context of tabu search (Glover, 1994; Glover and Laguna, 1997). This approach generates new solutions by exploring trajectories that connect high-quality solutions, by starting from one of these solutions, called an *initiating solution*, and generating a path in the neighbourhood space that leads toward the other solutions, called *guiding solutions*. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions.

Path relinking can be considered an extension of the Combination Method of scatter search. Instead of directly producing a new solution when combining two or more original solutions, PR generates paths between and beyond the selected solutions in the neighborhood space. The character of such paths is easily specified by reference to solution attributes that are added, dropped or otherwise modified by the moves executed. Examples of such attributes include edges and nodes of a graph, sequence positions in a schedule, vectors contained in linear programming basic solutions, and values of variables and functions of variables.

The approach may be viewed as an extreme (highly focused) instance of a strategy that seeks to incorporate attributes of high quality solutions, by creating inducements to favor these attributes in the moves selected. However, instead of using an inducement that merely encourages the inclusion of such attributes, the path relinking approach subordinates other considerations to the goal of choosing moves that introduce the attributes of the guiding solutions, in order to create a "good attribute composition" in the current solution. The composition at each step is determined by choosing the best move, using customary choice criteria from a restricted set — the set of those moves currently available that incorporate a maximum number (or a maximum weighted value) of the attributes of the guiding solutions. A survey of this methodology can be found in Laguna and Martí (2003). Path relinking in the context of GRASP was first introduced by Laguna and Martí (1999) as a form of intensification.

We combine solutions obtained from the GRASP_PieceExt and the GRASP_Strip. As will be shown, these procedures produce solutions with different structures. While solutions from GRASP_Strip tend to be of higher quality, solutions from GRASP_PieceExt exhibit more diversity. The combination of these two characteristics, quality and diversity, can be very fruitful in the search for solutions of even higher quality. We build a set of elite solutions by taking the *m* best solutions from each procedure. This set of *2m* solutions is ordered by non-increasing value, and each one is used as a *guiding solution* for all the *initiating solutions* behind it in the ordered set. We take each strip from the guiding solution and add it at the beginning of the strip list of the initiating solution. In order to accommodate the enlarged solution into the stock sheet, one or more strips from the end of the list are eliminated. Moreover, this double move inclusion-elimination may produce a solution in which some of the demands are exceeded. In this case, the strips of the original initiating solution containing pieces that produce an excess of the demands are also eliminated. All the empty space produced in these moves is merged into a waste strip which is filled

again by using the BLP method. At the end of this process, the strip list of the guiding solution will have been imposed on the initiating solution, but the intermediate steps, in which the solutions will have a blend of strips from both solutions, can be better than any of them.

5. Computational Experiments

We use two sets of test problems in our experimentation. On the one hand, we have the set composed of 38 medium size instances which have appeared in literature and have been collected and used by Hifi and Roucairol (2001). The optimal solutions are known and can be used as a reference for the quality of heuristic procedures. On the other hand, we have the set composed by the 20 large constrained test problems generated by Alvarez-Valdes et al (2002). Most of the optimal solutions for these problems have been recently obtained by Belov and Scheithauer (2003). The study of alternatives and parameter tuning in Tables 1 to 5 is performed on the first set of 38 instances for the case in which the first cut is horizontal. The final algorithm will be tested on both test sets and in both cases of horizontal and vertical first cutting stage. All the algorithms are coded in C++ and run on a Pentium IV at 3 Ghz.

5.1. Bottom-Left procedures

Table 1 shows the comparison on the first set of test problems between the standard Bottom-Left Procedure (BLP), the Generalized BLP, in which the pieces are randomly ordered before going into BLP (GBLP), and the OptQuest Callable Library (OCL). We report the results of the GBLP from 1000 and 100000 random initial orderings. The first row contains the average solution value, the second is the average deviation from optimum (in percentage), the third is the number of optimal solutions, out of 38 instances, and the fourth row is the average computing time.

	BLP	GBLP (1000)	GBLP (100000)	OCL
Value	8373.39	9370.86	9578.55	9618.81
Deviation	15.69%	2.37%	0.80%	0.49%
Num. of Opt.	1	13	16	18
CPU seconds	<0.01	<0.01	0.016	0.35

Table 1. Comparison of Bottom-Left Procedures

Table 1 shows that the OCL implementation provides the best solutions in terms of quality. However, it takes much more running time than the other methods. The BLP procedure achieves relatively good results considering its simplicity. GBLP can be considered as a good bottom-left procedure since it presents a good trade-off between solution quality and computing time.

5.2. GRASP_Piece algorithm

In our next experiment we consider the influence of the parameter α used to define the Restricted Candidate List in the constructive phase of the GRASP_Piece algorithm. Specifically, Table 2 compares the values 0.25, 0.50, 0.75 and 0.90 for this parameter, when running the construction phase for 2000 iterations (no improvement phase is applied in this experiment).

	α			
	0.25	**0.50**	**0.75**	**0.90**
Value	9136.32	9298.61	9353.68	9345.18
Deviation	4.35%	3.34%	3.13%	3.32%
Num. of Opt.	7	6	8	8
CPU seconds	0.028	0.028	0,030	0.027

Table 2. GRASP_Piece Constructions (2000 iterations)

Table 2 shows that α=0.75 provides the best solutions. A tendency is observed in which the greater the α–value, the better the results, up to one point (found at α=0.75) in which no further improvement is

possible (it seems that the Restricted Candidate List becomes too restrictive with higher values). Therefore, α is set to 0.75 in the following experiments.

Table 3 shows the contribution to the quality of solutions of the two versions of the improvement phase of the GRASP_Piece algorithm. Specifically, this Table shows the results obtained with three variants: the GRASP construction phase without improvement (GRASP_Piece Construction), the GRASP method considering construction and improvement as described in Section 3.1 (GRASP_Piece) and the GRASP method with the extended improvement method also described in Section 3.1 (GRASP_PieceExt). These three methods are run for 2000 iterations for each of the 38 problems considered in these preliminary experiments.

	GRASP_Piece Construction	**GRASP_Piece**	**GRASP_PieceExt**
Value	9353.68	9445.08	9462.11
Deviation	3.13%	2.24%	1.97%
Num. of Opt.	8	10	14
CPU seconds	0,030	0,037	0,048

Table 3. Comparison of GRASP variants (2000 iterations)

Table 3 shows that the extended improvement procedure significantly decreases the average deviation from optimum (and increases the number of optima) with respect to the construction phase, with a modest increment in running times. If the iteration limit is increased to 200,000 iterations, the average deviation of the GRASP with the GRASP_PieceExt improvement procedure achieves an average deviation from optimum of 0.59% and is able to match 23 optimum solutions out of 38.

5.3. GRASP_Strip algorithm

Table 4 shows the performance of several alternatives for the constructive procedures of the GRASP_Strip algorithm. We have studied two possibilities for selecting the best strip, according to the maximum absolute value and the maximum relative value. We have also considered the possibility of solving the knapsack problems heuristically, using the greedy algorithm by Martello and Toth (1990).

	Heuristic		**Exact**	
	Absolute	**Relative**	**Absolute**	**Relative**
Value	7674,47	8410,89	8818,05	9315,53
Deviation	21.67%	16.68%	10,69%	3.44%
Num. of Opt.	0	0	2	13
CPU seconds	< 0.01	< 0.01	< 0.01	< 0.01

Table 4. Comparison of deterministic constructive procedures for GRASP_Strip

Table 4 shows that solving the knapsack problem exactly and selecting the best strip according to the maximum relative value criterion provides much better solutions than the other alternatives considered. Since the exact method takes extra running time (not reflected in Table 4 because it shows a single run of the methods), we set the number of iterations to 200 in the GRASP_Strip algorithm.

	α (Constructive phase)				GRASP_Strip
	0.25	0.50	0.75	0.90	
Value	9625.42	9625.55	9625.92	9651.71	9659.45
Deviation	0.33%	0.33%	0.35%	0.25%	0.22%
Optima	31	31	30	34	35
CPU time	0.09	0.09	0.10	0.10	0.18

Table 5. GRASP_Strip results (200 iterations)

Table 5 compares different values of the parameter α used to define the Restricted Candidate List in the constructive phase of the GRASP_Strip algorithm and shows the results of the complete GRASP_Strip in

the last column. It is clearly shown that the GRASP_Strip method outperforms the GRASP_Piece algorithm although it consumes longer running times.

5.4. Combining GRASP_Piece and GRASP_Strip

In the following experiment, we have compared both GRASP methods. In order to apply the path relinking strategy we need a set of elite solutions with both quality and diversity. Therefore, we compare both characteristics in GRASP_Piece and GRASP_Strip. We have created a measure of diversity for a set of solutions P, which is calculated as follows:

1. Calculate the average frequency *afreq(i)* of each piece *i* in the solutions in *P*.
2. Calculate the distance of each solution in P with respect to the "average solution" in which each piece *i* appears *afreq(i)* times. (Note that this can be an unfeasible solution). This distance is the Euclidean distance between the frequency vectors of both solutions.

We have generated a set of 1000 solutions with each GRASP method, and computed the objective function value and the distance value (w.r.t. the "average solution") for each one. This experiment confirms that GRASP_Strip produces solutions of better quality than GRASP_Piece. However, the latter method presents a wider range of diverse solutions. Moreover, we have found that the solutions of both methods are of different structure. Therefore, we will consider the solutions obtained with both methods in the elite set of the path relinking phase. We do not reproduce tables for this experiment here; we only show one example in Figure 6.

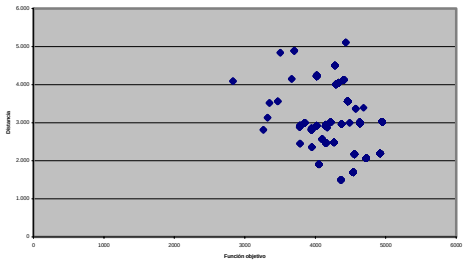


Figure 6.1 GRASP_Strip (α=0.25)

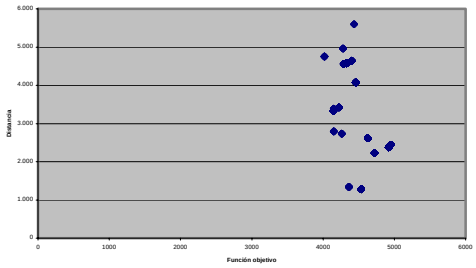


Figure 6.2 GRASP_Strip (α=0.75)

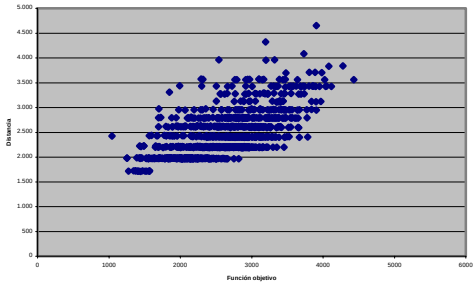


Figure 6.3 GRASP_Piece (α=0.25)

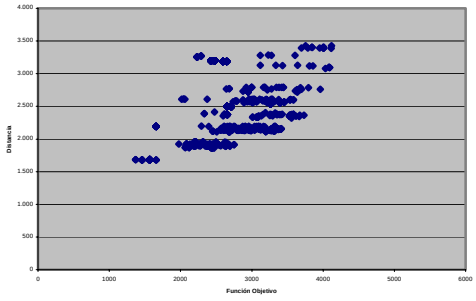


Figure 6.4 GRASP_Piece (α=0.75)

Figure 6 shows the diagram of the population obtained with both GRASP methods (and two values of the parameter α) in example CW3. Specifically, it depicts for each solution the objective function value (x-axis) and the distance value (y-axis). It shows that both methods provide solutions of different structure and how the procedure depends on the α−value. The average objective function of the 1000 solutions is 2611.26 and 2746.97 for the GRASP_Piece (α=0.25 and α=0.75), and 4539.54 and 4695.72 for the GRASP_Strip (α=0.25 and α=0.75). On the other hand, the average distance is 2.46 and 2.42 for the GRASP_Piece (α=0.25 and α=0.75) and 2.94 and 2.79 for the GRASP_Strip (α=0.25 and α=0.75).

5.5. Path Relinking

Finally we present the results of the complete algorithm. First, algorithms GRASP_Piece and GRASP_Strip run separately with iteration limits of 2000 and 200 respectively. Then, the best 10 solutions provided by each algorithm form the set of elite solutions upon which the Path Relinking procedure acts. The results on the first test set of 38 problems appear in Table 6. We will refer to this complete procedure in which the three methods are run as PathRelink. This table compares this procedure with the best results published up to now for these problems: H&R (Hifi & Roucairol, 2001) for both types of first cut, and ESGA (Hifi & M'Hallah, 2003) for the case in which the first cut is horizontal. On this set of problems all the methods require very low computing times (below 0.5 seconds).

	First cut horizontal				First cut vertical		
Instance	Optimum	PathRelink	H&R	ESGA	Optimum	PathRelink	H&R
HH	10689	*10689*	10545	*10689*	9246	9141	8298
2	2535	*2535*	2375	*2535*	2444	*2444*	2305
3	1720	*1720*	1660	1700	1740	*1740*	1440
A1	1820	*1820*	*1820*	*1820*	1820	*1820*	1640
A2	2315	*2315*	1940	2295	2310	*2310*	2240
STS2	4450	*4450*	4280	4420	4620	*4620*	*4620*
STS4	9409	*9409*	9263	*9409*	9468	*9468*	8531
CHL1	8360	*8360*	7421	8347	8208	*8208*	7597
CHL2	2235	*2235*	2076	*2235*	2086	*2086*	*2086*
CW1	6402	*6402*	*6402*	*6402*	6402	*6402*	*6402*
CW2	5354	*5354*	*5354*	*5354*	5159	*5159*	5032
CW3	5287	4947	4434	*4947*	5689	*5689*	5026
Hchl2	9630	*9630*	9079	9616	9528	*9528*	9274
Hchl9	5100	*5100*	4610	5000	5060	*5060*	4680
2s	2430	*2430*	2375	*2430*	2450	*2450*	2188
3s	2599	*2599*	2470	*2599*	2623	*2623*	2470
A1s	2950	*2950*	*2950*	*2950*	2910	*2910*	2812
A2s	3423	*3423*	*3423*	*3423*	3451	*3451*	3445
STS2s	4569	*4569*	4342	*4569*	4625	*4625*	4342
STS4s	9481	*9481*	9258	9409	9481	*9481*	8563
OF1	2713	*2713*	2437	*2713*	2660	*2660*	*2660*
OF2	2515	*2515*	2307	*2515*	2522	*2522*	2442
W	2623	*2623*	2470	*2623*	2599	*2599*	2432
CHL1s	13036	*13036*	12276	13014	12602	*12602*	12314
CHL2s	3162	*3162*	*3162*	*3162*	3198	*3198*	*3198*
A3	5380	*5380*	5348	*5380*	5403	*5403*	5082
A4	5885	*5885*	*5885*	*5885*	5905	*5905*	5705
A5	12553	*12553*	12276	12276	12449	*12449*	12276
CHL5	363	*363*	330	*363*	344	*344*	*344*
CHL6	16572	*16572*	16157	16402	16281	*16281*	15862
CHL7	16728	*16728*	16037	16632	16602	*16602*	16111
CU1	12312	*12312*	12243	*12312*	12200	*12200*	*12200*
CU2	26100	*26100*	*26100*	*26100*	25260	*25260*	24750
Hchl3s	11961	*11961*	11410	11691	11829	*11829*	10836
Hchl4s	11408	*11408*	10545	11165	11258	*11258*	9573
Hchl6s	60170	*60170*	56105	*60170*	59853	*59853*	58121
Hchl7s	62459	*62459*	60384	61660	62845	*62845*	60683
Hchl8s	729	715	662	727	791	*791*	715
Deviation		0,22%	4.68%	0.58%		0,03%	5.07%
#Optima		36	8	22		37	7

Table 6. Comparison of Path Relinking and Hifi &Roucairol algorithm

Table 6 clearly shows that the PathRelinking method outperforms previous approaches, since it is able to obtain 36 optimal solutions with first cut horizontal and 37 optimal solutions with first cut vertical. This compares favorably with the other existing methods.

Tables 7 and 8 show the results of our three procedures on a set of large scale cutting problems. Instances from APT 30 to APT39 are unweighted, while those from APT40 to APT 49 are weighted. The second column, in which the optimal values are shown, has been taken from Belov and Scheithauer (2003). Their results do not include the optimal solution for instances APT33 with first cut horizontal and APT30 and APT38 with first cut vertical. In these cases, they give the best solution found when running their exact procedures with a time limit of 600 seconds (on an AMD K7 Athlon XP 1000Mhz). Our algorithms are compared with HESGA (Hifi and M'Hallah, 2003), which seems to be the best published heuristic algorithm. The results of HESGA have been taken from Hifi & M'Hallah (2003) for the case in which the first cut is horizontal and from Belov and Scheithauer (2003) for the case in which the first cut is vertical.

	First cut horizontal				
Instance	Optimum	GRASP_Piece	GRASP_Strip	Path Relink	HESGA
APT30	140168	134531	140168	140168	140168
APT31*	820260	813752	808597	813935	818512
APT32	37880	36303	37799	37799	37880
APT33	235580	230631	235580	235580	234564
APT34	356159	353063	354219	356159	356159
APT35	614429	595423	607999	612337	613784
APT36	129262	125570	129262	129262	129262
APT37	384478	376052	384478	384478	382910
APT38	259070	247753	258134	258382	258221
APT39	266135	261552	265853	265853	265621
APT40	63945	61539	63945	63945	63945
APT41	202305	192657	202305	202305	202305
APT42	32589	31767	32487	32589	32589
APT43	208998	204869	208998	208998	208571
APT44	70940	63160	70901	70901	70678
APT45	74205	69716	74205	74205	74205
APT46	146402	143305	146402	146402	146402
APT47	144317	136270	144317	144317	143458
APT48	165428	150765	165428	165428	162032
APT49	206965	189471	205946	206965	204574
Deviation		4,06	0,23	0,09	0,30
Optima		0	11	14	9
CPU time		0.17	0.74	1.18	13.53[#]

* Optimal solution of APT31 is not known

[#] Running times on a UltraSparc10 (250Mhz, 128 Mb of RAM)

Table 7. Comparison of algorithms on large problems (First cut horizontal)

The results show that even for these large problems, the Path Relinking method obtains high quality solutions in very short computing times and is able to outperform the best previous approach. Note that for APT30 and APT38 with first cut vertical the Path Relinking method is able to achieve better solutions than the exact procedure of Belov and Scheithauer, where 600 seconds was the maximum time deployed.

	First cut vertical				
Instance	Optimum	GRASP_Piece	GRASP_Strip	Path Relink	HESGA
APT30*	140067	136735	139774	140197	140007
APT31	821073	795354	809109	814451	818296
APT32	37973	36402	37973	37973	37744
APT33	234670	229482	234670	234670	234538
APT34	357741	347548	354675	354675	353590
APT35	614336	601412	601492	607352	614132
APT36	128814	126687	128814	128814	128814
APT37	385811	370484	383406	385164	385811
APT38*	258812	253414	259028	259028	258040
APT39	266378	256778	265062	265062	265330
APT40	65584	62647	65584	65584	65044
APT41	196559	194551	196559	196559	195453
APT42	33012	31347	33012	33012	32937
APT43	212062	206185	212062	212062	212062
APT44	69784	69571	69474	69784	69732
APT45	69988	69162	69988	69988	69857
APT46	147021	141464	147021	147021	147021
APT47	142935	142199	142935	142935	142935
APT48	162458	153117	162458	162458	160318
APT49	211784	202531	211784	211784	210169
Deviation		2,87	0,30	0,16	0,34
Optima		0	13	15	5
CPU time		0,17	0,89	1,34	14.42[#]

* Optimal solutions of APT30 and APT38 are not known

[#] Running times on a UltraSparc10 (250Mhz, 128 Mb of RAM)

Table 8. Comparison of algorithms on large problems (First cut vertical)

6. Conclusions

We have presented two GRASP methods for the constrained two-dimensional two-staged cutting stock problem. A set of elite solutions is constructed with the best solutions of both methods and a Path Relinking algorithm is applied to this elite set. The combination of these three methods has been shown to be remarkably efficient in solving a set of well known instances compared with the best previous approach for this problem. Finally, our experience with the path relinking approach shows that, although computationally more expensive, this strategy was able to improve the performance of our basic GRASP implementations.

Acknowledgments

Antonio Parajon's research is partially supported by the visiting professor fellowship program of the *Agencia Valenciana de Ciencia y Tecnología* (Grant Ref. CTESIN/2003/022). Ramon Alvarez-Valdes, Rafael Marti and Jose Manuel Tamarit are partially supported by the *Ministerio de Ciencia y Tecnología* (Grant Refs. TIC2003-C05-01, TIC2002-10886-E, DPI2002-02553) and by the *Agencia Valenciana de Ciencia y Tecnologia* (GRUPOS2003/174-189).

References

Alvarez-Valdes, R., Parajon, A. and Tamarit, J.M. (2002) "A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems", *Computers and Operations Research*, vol. 29, pp. 925-947.

Beasley, J.E. (1985) "Algorithms for unconstrained two-dimensional guillotine cutting", *Journal of the Operational Research Society*, vol. 36, pp. 297-306.

Belov, G. and Scheithauer, G. (2003) "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting", *Technical Report MATH-NM-03*, Dresden University.

Belov, G. and Scheithauer, G. (2003) "Models with variable strip widths for two-dimensional two-stage cutting", *Technical Report*, Dresden University.

Dyckhoff, H. (1990) "A typology of cutting and packing problems", *European Journal of Operational Research*, vol 44, pp. 145-159.

Festa, P. and Resende, M.G.C. (2001) "GRASP: An annotated bibliography" in *Essays and Surveys in Metaheuristics,* M.G.C. Resende and P. Hansen (Eds.), Kluwer Academic Publishers, Boston, pp. 325-367.

Gilmore, P.C. and Gomory, R.E. (1961) "A linear programming approach to the cutting stock problem", *Operations Research*, vol. 9, pp. 849-859.

Gilmore, P.C. and Gomory, R.E. (1965) "Multistage cutting problems of two and more dimensions", *Operations Research*, vol 13, pp. 94-119.

Glover, F. (1994) "Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms)," *Discrete Applied Mathematics*, vol. 49, pp. 231-255.

Glover, F. and M. Laguna (1997) *Tabu Search,* Kluwer Academic Publishers, Boston.

Hifi, M. (2001) "Exact algorithms for large-scale unconstrained two and three staged cutting problems", *Computational Optimization and Applications*, vol. 18, pp. 63-88.

Hifi, M and M'Hallah, R.(2003) "Strip generation algorithms for constrained two-dimensional two-staged cutting problems", *Working paper*.

Hifi, M. and Roucairol, C. (2001) "Approximate and exact algorithms for constrained (un)weighted two-dimensional two-staged cutting stock problems", *Journal of Combinatorial Optimization*, vol. 5, pp. 465-494.

Laguna, M. and R. Martí (1999) "GRASP and Path Relinking for 2-Layer Straight Line Crossing Minimization," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 44-52.

Laguna, M. and R. Martí (2002) "The OptQuest Callable Library," *Optimization Software Class Libraries*, S. Voss and D. L. Woodruff (Eds.), Kluwer Academic Publishers, Boston, pp. 193-218.

Laguna, M. and R. Martí (2003) *Scatter Search – Methodology and Implementations in C,* Kluwer Academic Publishers, Boston.

Lodi, A. and Monaci, M. (2003) "Integer linear programming models for 2-staged two-dimensional knapsack problems", *Mathematical Programming*, Series B, vol. 94, pp. 257-278.

Martello, S. and Toth, P. (1990) *Knapsack problems. Algorithms and computer implementations*, Wiley, Chichester.

Morabito, R. and Garcia, V. (1998) "The cutting stock problem in the hardboard industry: A case study" *Computers and Operations Research*, vol. 25, pp. 469-485.

Resende, M.G.C. and Ribeiro, C.C. (2001) "Greedy Randomized Adaptive Search Procedures" in *State-of-the-art Handbook in Metaheuristics,* F. Glover and G. Kochenberger (Eds.), Kluwer Academic Publishers, Boston, pp. 219-250.

Sweeney, P.E. and Paternoster, E.R. (1992) "Cutting and packing problems: A categorized applications-oriented research bibliography" *Journal of the Operational Research Society*, vol. 43, pp. 691-706.