This article was downloaded by: [144.214.89.16] On: 18 September 2017, At: 07:51
Publisher: Institute for Operations Research and the Management Sciences (INFORMS)
INFORMS is located in Maryland, USA

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

GRASP and Path Relinking for the Two-Dimensional Two-Stage Cutting-Stock Problem

Ramón Alvarez-Valdes, Rafael Martí, Jose M. Tamarit, Antonio Parajón,

To cite this article:
Ramón Alvarez-Valdes, Rafael Martí, Jose M. Tamarit, Antonio Parajón, (2007) GRASP and Path Relinking for the Two-Dimensional Two-Stage Cutting-Stock Problem. INFORMS Journal on Computing 19(2):261-272. https://doi.org/10.1287/ijoc.1050.0169

Full terms and conditions of use: http://pubsonline.informs.org/page/terms-and-conditions

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2007, INFORMS

Please scroll down for article—it is on subsequent pages

INFORMS is the largest professional society in the world for professionals in the fields of operations research, management science, and analytics.
For more information on INFORMS, its publications, membership, or meetings visit http://www.informs.org

RIGHTSLINK

INFORMS Journal on Computing
Vol. 19, No. 2, Spring 2007, pp. 261–272
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1902 | 0261

DOI 10.1287/ijoc.1050.0169
© 2007 INFORMS

GRASP and Path Relinking for the Two-Dimensional Two-Stage Cutting-Stock Problem

Ramón Alvarez-Valdes, Rafael Martí, Jose M. Tamarit
Departamento de Estadística e I.O., Universidad de Valencia, Dr. Moliner, 50, 46100 Burjassot, Valencia, Spain
{ramon.alvarez@uv.es, rafael.marti@uv.es, jose.tamarit@uv.es}

Antonio Parajón
Departamento de Matemáticas, Universidad Nacional Autónoma de Nicaragua, UNAN-Managua,
ENEL Central 2 Km Sur, Managua, Nicaragua, ramon.a.parajon@uv.es

We develop a greedy randomized adaptive search procedure (GRASP) for the constrained two-dimensional two-stage 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 a wide range of instances, 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
History: Accepted by Michel Gendreau, Area Editor for Heuristic Search and Learning; received June 2004; revised March 2005, July 2005; accepted November 2005.

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 $L \times W$ and of piece i are $l_i \times w_i$. 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 nonused 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*. 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 $l \times w$ is not the same as a piece of dimensions $w \times l$. We use only *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-stage 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. (Other authors call the strips *levels* or *shelves* in the context of packing problems; see, for instance, Lodi and Monaci 2003.) We will allow *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 *nonexact 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 shows a stock rectangle S (on the left) in which we want to cut the two pieces shown below it. We show a strip without trimming (in the center) and another with trimming (on the right). 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 considerable research over the years, as documented in

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

RIGHTSLINK

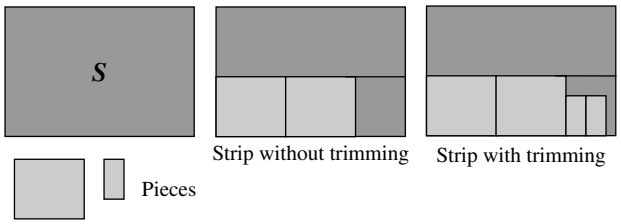

Figure 1 Example of Trimming

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 Beasley (1985), who proposes heuristically reducing the state space of his dynamic-programming procedure, and in 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 (2006) proposed new procedures in which they improve the Hifi and Roucairol algorithm and combine it with local search methods.

We develop a heuristic solution procedure for two-stage 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, nonexact problem, but it can be easily adapted to the exact case. Section 2 presents the bottom-left procedure, a constructive algorithm that 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 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 nonincreasing 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 two-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 no more than or equal to the width of the previous-iteration piece; therefore it is no more than the width of the strip. We continue in this way until the length l of the selected piece exceeds the remaining length r_s of the strip (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.

The BLP starts a new strip each time 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 $l_i \le r_s$. 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 $l_i \le r_s$ 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 $l_i \le r_s$ is found.

A pseudo-code of the BLP to produce a solution to the problem is in Figure 2, 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

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

```
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 ≤ 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 ≤ 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 ≤ r_s}
  While (R ≠ ∅)
    Select the first piece i from R respecting the fixed order
    Add i to the strip
    b_i = b_i - 1. r_s = r_s - l_i
    Value = Value + c_i
    R = R\{i}
}
```

Figure 2 Pseudo-Code of the Bottom-Left Procedure

ordering of the pieces. The tie is then resolved according to the ratio, where the piece with the highest ratio enters first.

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 "$(l_i \le r_s)$" in the outer *If* statement with the condition "$(l_i \le r_s$ and $w_i \le Wstrip)$," 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.

In Section 5 we will show the results obtained with both variants of the bottom-left procedures; the original BLP and the GBLP. The BLP method also appears as BFD (best fit decreasing height) or FFD (fit decreasing height) in related papers.

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 application of the local search phase. The best local optimum is reported as the best overall solution found. Resende and Ribeiro (2001) present a comprehensive review of GRASP, and an extensive survey of the 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 method adds an entire strip. As will be shown in Section 5, 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 evaluation of the elements is used to create a restricted candidate list (RCL). The RCL consists of the best elements, i.e., those with the smallest incremental cost. The element to be added into the partial solution is randomly selected from those in the RCL. Once the selected element is added to the partial solution, the candidate list is updated and the incremental costs are recalculated.

The solutions generated by a greedy randomized construction are not necessarily locally 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. Sections 3.1 and 3.2 describe in detail our GRASP implementations for the TDC problem.

3.1. A GRASP Based on Strips

3.1.1. 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

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

and width w is given by the optimal solution of the following problem:

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

where $P_w = \{j \mid w_j \le w\}$.

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. In addition, Hifi and M'Hallah (2006) combined the solution obtained through a series of knapsack problems with a filling procedure to complete some free areas. Their heuristic procedure is coupled with a method based on linear combinations. We propose a new approach to this problem, based partially on these previous strategies. 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, $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_r$.

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 $z^*_{\overline{w}_k}/\overline{w}_k$. 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 3 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 $\overline{w}_k$, 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 $\overline{w}_k$, if the optimal solution of that problem in the previous iteration $x^*_{\overline{w}_k}$ is less 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 $\overline{w}_k$, but $\overline{w}_k > Wfree - \min\{\overline{w}_i \mid \overline{w}_i \in \mathfrak{W}\}$, no other strip would fit into the stock rectangle if a strip of width $\overline{w}_k$ is added. Therefore, when updating $\mathfrak{W}$, all widths $\overline{w}_k$ such that $\overline{w}_k + \min\{\overline{w}_i \mid \overline{w}_i \in \mathfrak{W}\} > Wfree$ can be replaced in $\mathfrak{W}$ by a unique value $Wfree$.

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 $\mathfrak{W} = \{\overline{w}_k \mid$ a piece i with $w_i = \overline{w}_k$ and $b_i > 0$ exists$\}$
Order $\mathfrak{W}$ by decreasing $\overline{w}_k$
While $(\mathfrak{W} \neq \varnothing)$ {
  Let $z^*_{\max} = 0$; $x^*_{\max} = 0$; $w^*_{\max} = 0$.
  For each $\overline{w}_k \in \mathfrak{W}$ {
    Solve the problem $KP_{(L,\,\overline{w}_k)}$. Let $z^*_{\overline{w}_k}$ be the optimal value and let $x^*_{\overline{w}_k}$ be 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 $\mathfrak{W} = \mathfrak{W} \backslash \{\overline{w}_k\}$
    If, for some $\overline{w}_k$, $\overline{w}_k > Wfree$, then $\mathfrak{W} = \mathfrak{W} \backslash \{\overline{w}_k\}$
  Else
    STOP
}

Figure 3 Pseudo-Code of the Constructive Phase of the GRASP Based on Strips

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 $\mathfrak{W}$. The restricted candidate list $\mathrm{RCL}_k = \{\overline{w}_k \in \mathfrak{W} \mid z^*_{\overline{w}_k} \ge \alpha z^*_{\max}\}$. From RCL_k a width is selected at random and the corresponding strip is added to the partial solution.

3.1.2. 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 mechanism is an adaptation of the strip-generation procedure introduced in Hifi and M'Hallah (2006).

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

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

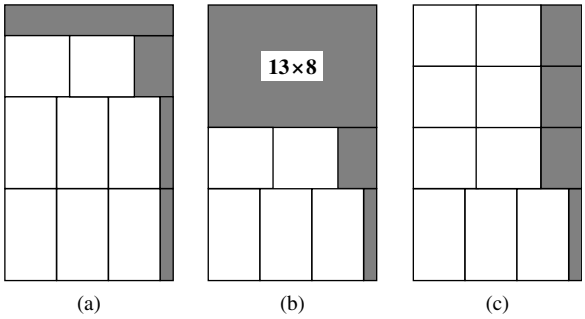

Figure 4 Improvement Move

repeated until a stopping criterion is met, as GRASP_Strip.

In Figure 4 we show an example of the construction and improvement of a solution. Suppose we are given a stock sheet (13×18) from which we have to cut two types of pieces, A (4×6) and B (5×4), 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, and 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 4(a). In the improvement phase, we eliminate the first strip, obtaining the situation in Figure 4(b) in which a region of 13×8 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 in Figure 4(c).

3.2. A GRASP Based on Pieces

3.2.1. 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 $w \le Wfree$.

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

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

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 $l_i \le r_s$ 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.

3.2.2. 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 less 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

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

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 resort to this "one-to-two" move only if the "one-to-one" move fails. Note that piece i can be replaced with two pieces if the sum of their lengths is less 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.

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 neighborhood space that leads toward the other solutions, called *guiding solutions*. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions.

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. 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 method is 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 GRASP_PieceExt and 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 nonincreasing 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 that 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 three sets of test problems in our experimentation. First, we have the set composed of 38 medium-sized instances that 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). We have also considered a third set of ten instances with a wide range of sizes and characteristics, introduced by Wang and Valenzuela (2001) in the context of the strip-packing problem. The study of alternatives and parameter tuning in Tables 1–5 was 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 the three test sets and in both cases of horizontal and vertical first cutting stage. All the algorithms were coded in C++ and run on a Pentium IV at 3 GHz.

Table 1 Comparison of Bottom-Left Procedures

	BLP	GBLP (1,000)	GBLP (100,000)
Value	8,373.39	9,370.86	9,578.55
Deviation (%)	15.69	2.37	0.80
No. of optima	1	13	16
CPU seconds	<0.01	<0.01	0.016

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

Table 2 Comparison of Deterministic Constructive Procedures for GRASP_Strip

	Heuristic		Exact	
	Absolute	Relative	Absolute	Relative
Value	7,674.47	8,410.89	8,818.05	9,315.53
Deviation (%)	21.67	16.68	10.69	3.44
No. of optima	0	0	2	13
CPU seconds	<0.01	<0.01	<0.01	<0.01

5.1. Bottom-Left Procedures

Table 1 shows the comparison on the first set of test problems between the standard BLP and the GBLP, in which the pieces are randomly ordered before going into BLP (GBLP). We report the results of the GBLP from 1,000 and 100,000 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.

Table 1 shows that, as expected, the GBLP implementation applied from 100,000 random initial orderings provides the best solutions in terms of quality. However, it takes more time than do the other methods. Nonetheless, they are all quite fast and this experiment provides a baseline for future comparisons. 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_Strip Algorithm

Table 2 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).

Table 2 shows that solving the knapsack problem exactly and selecting the best strip according to the maximum-relative-value criterion provides much better solutions than do the other alternatives considered. Since the exact method takes extra time (not reflected in Table 2 because it shows a single run of the methods), we set the number of iterations to 200 in the GRASP_Strip algorithm.

Table 3 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 more time.

Table 3 GRASP_Strip Results (200 Iterations)

	α (Constructive phase)				
	0.25	0.50	0.75	0.90	GRASP_Strip
Value	9,625.42	9,625.55	9,625.92	9,651.71	9,659.45
Deviation (%)	0.33	0.33	0.35	0.25	0.22
No. of optima	31	31	30	34	35
CPU time	0.09	0.09	0.10	0.10	0.18

Table 4 GRASP_Piece Constructions (2,000 Iterations)

	α			
	0.25	0.50	0.75	0.90
Value	9,136.32	9,298.61	9,353.68	9,345.18
Deviation (%)	4.35	3.34	3.13	3.32
No. of optima	7	6	8	8
CPU seconds	0.028	0.028	0.030	0.027

5.3. 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 4 compares the values 0.25, 0.50, 0.75, and 0.90 for this parameter, when running the construction phase for 2,000 iterations (no improvement phase is applied in this experiment).

Table 4 shows that $\alpha = 0.75$ provides the best solutions. A tendency is observed in which the greater the α value, the better the results, up to a point (found at $\alpha = 0.75$) beyond which no further improvement seems possible (it appears that the restricted candidate list becomes too restrictive with higher values). Therefore, α is set to 0.75 in the following experiments.

Table 5 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 were run for 2,000 iterations for each

Table 5 Comparison of GRASP Variants (2,000 Iterations)

	GRASP_Piece construction	GRASP_Piece	GRASP_PieceExt
Value	9,353.68	9,445.08	9,462.11
Deviation (%)	3.13	2.24	1.97
No. of optima	8	10	14
CPU seconds	0.030	0.037	0.048

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

of the 38 problems considered in these preliminary experiments.

Table 5 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 increase 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.

5.4. Path Relinking

Our path-relinking implementation combines solutions obtained with both GRASP methods. It seems that the solutions obtained with GRASP_Piece have different structures from those obtained with GRASP_Strip, and therefore the path-relinking algorithm will combine both sets of solutions (as it will with the solutions within each set). Our next experiment measures the relative contribution of each type of combination to achieve the best solution found. We have run two versions of our complete algorithm. In the first one, called PR(200, 20), algorithms GRASP_Piece and GRASP_Strip run separately with iteration limits of 200 and 20 respectively. Then the best five solutions provided by each algorithm form the set of elite solutions upon which the path relinking procedure acts. In the second version, called PR(2,000, 200), the iteration limits of both GRASP methods are 2,000 and 200 respectively.

Table 6 shows the number of times each type of combination is able to obtain the best solution found with the PR method. Column PP counts, for each variant considered, the number of times that an application of the path relinking, combining two solutions of the GRASP_Piece method, is able to produce the best solution found. Similarly, PS (and SS) records the number of times a combination of GRASP_Piece with GRASP_Strip (GRASP_Strip with GRASP_Strip) is able to produce the best solution found with our algorithm. Table 6 summarizes these values for the 38 medium-sized and 20 large problems.

Table 6 Contribution of Type of Combinations

Instances	First cut	PR version	PP	PS	SS
38 medium problems	Horizontal	PR(200, 20)	14	18	13
	Vertical	PR(200, 20)	20	33	26
	Horizontal	PR(2,000, 200)	4	10	4
	Vertical	PR(2,000, 200)	5	10	7
20 large problems	Horizontal	PR(200, 20)	11	34	23
	Vertical	PR(200, 20)	8	25	21
	Horizontal	PR(2,000, 200)	8	28	13
	Vertical	PR(2,000, 200)	6	15	12
Total			76	173	119

Table 6 clearly shows that many of the best solutions result from application of path-relinking to a solution generated with GRASP_Piece and a solution generated with GRASP_Strip. This type of combination provides 173 best solutions when running the procedure over the 58 problems of medium and large size. However, the combination of solutions generated with the same GRASP variant is able to obtain the best solution found with the method in a relatively large number of cases (119 for the GRASP_Strip and 76 for the GRASP_Piece). Therefore, we will apply our path-relinking algorithm to all the pairs of solutions in the elite set. This table also shows that when the GRASP methods run longer, the path-relinking phase obtains a lower number of best solutions (compare PR(2,000, 200) with PR(200, 20)). This is to be expected since shorter GRASP executions provide medium-quality solutions in which improvements are more likely than in longer executions, which produce high-quality solutions by themselves.

In the following experiment we compare our path-relinking procedure PR(2,000,200) with the best results published up to now for the 38 medium-size problems. Specifically, we target the H&R method (Hifi and Roucairol 2001) for both types of first cut, and ESGA (Hifi and M'Hallah 2006) for the case in which the first cut is horizontal. Table 7 shows the results of this comparison in which all methods require very low computing times (below 0.5 second).

Table 7 clearly shows that the path-relinking method PR(2,000, 200) outperforms previous approaches, since it is able to obtain 38 optimal solutions with first cut horizontal and 37 optimal solutions with first cut vertical. This compares favorably with the other existing methods.

Tables 8 and 9 show the results of our three procedures on a set of large cutting problems. Instances from APT30 to APT39 are unweighted, while those from APT40 to APT49 are weighted. The second column, in which the optimal values are shown, is from Belov and Scheithauer (2003). Their results do not include the optimal solution for instances APT31 with first cut horizontal and APT30 and APT38 with first cut vertical. With CPLEX and the formulation given in Lodi and Monaci (2003) we obtained the optimal solutions of these three instances with extremely long running times (17 hours in APT38). Our algorithms are compared with HESGA (Hifi and M'Hallah 2006), which seems to be the best published heuristic algorithm. The results of HESGA have been taken from Hifi and M'Hallah (2006) 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.

RIGHTSLINK

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

Table 7 Comparison of Path Relinking and Hifi and Roucairol Algorithms

	First cut horizontal				First cut vertical		
Instance	Optimum	PR	H&R	ESGA	Optimum	PR	H&R
HH	10,689	*10,689*	10,545	*10,689*	9,246	*9,249*	8,298
2	2,535	*2,535*	2,375	*2,535*	2,444	*2,444*	2,305
3	1,720	*1,720*	1,660	1,700	1,740	*1,740*	1,440
A1	1,820	*1,820*	*1,820*	*1,820*	1,820	*1,820*	1,640
A2	2,315	*2,315*	1,940	2,295	2,310	*2,310*	2,240
STS2	4,450	*4,450*	4,280	4,420	4,620	*4,620*	*4,620*
STS4	9,409	*9,409*	9,263	*9,409*	9,468	*9,468*	8,531
CHL1	8,360	*8,360*	7,421	8,347	8,208	*8,208*	7,597
CHL2	2,235	*2,235*	2,076	*2,235*	2,086	*2,086*	*2,086*
CW1	6,402	*6,402*	*6,402*	*6,402*	6,402	*6,402*	*6,402*
CW2	5,354	*5,354*	*5,354*	*5,354*	5,159	*5,159*	5,032
CW3	5,287	*5,287*	4,434	*4,947*	5,689	*5,689*	5,026
Hchl2	9,630	*9,630*	9,079	9,616	9,528	*9,528*	9,274
Hchl9	5,100	*5,100*	4,610	5,000	5,060	*5,060*	4,680
2s	2,430	*2,430*	2,375	*2,430*	2,450	*2,450*	2,188
3s	2,599	*2,599*	2,470	*2,599*	2,623	*2,623*	2,470
A1s	2,950	*2,950*	*2,950*	*2,950*	2,910	*2,910*	2,812
A2s	3,423	*3,423*	*3,423*	*3,423*	3,451	*3,451*	3,445
STS2s	4,569	*4,569*	4,342	*4,569*	4,625	*4,625*	4,342
STS4s	9,481	*9,481*	9,258	9,409	9,481	*9,481*	8,563
OF1	2,713	*2,713*	2,437	*2,713*	2,660	*2,660*	*2,660*
OF2	2,515	*2,515*	2,307	*2,515*	2,522	*2,522*	2,442
W	2,623	*2,623*	2,470	*2,623*	2,599	*2,599*	2,432
CHL1s	13,036	*13,036*	12,276	13,014	12,602	*12,602*	12,314
CHL2s	3,162	*3,162*	*3,162*	*3,162*	3,198	*3,198*	*3,198*
A3	5,380	*5,380*	5,348	*5,380*	5,403	*5,403*	5,082
A4	5,885	*5,885*	*5,885*	*5,885*	5,905	*5,905*	5,705
A5	12,553	*12,553*	12,276	12,276	12,449	*12,449*	12,276
CHL5	363	*363*	330	*363*	344	*344*	*344*
CHL6	16,572	*16,572*	16,157	16,402	16,281	*16,281*	15,862
CHL7	16,728	*16,728*	16,037	16,632	16,602	*16,602*	16,111
CU1	12,312	*12,312*	12,243	*12,312*	12,200	12,183	*12,200*
CU2	26,100	*26,100*	*26,100*	*26,100*	25,260	*25,260*	24,750
Hchl3s	11,961	*11,961*	11,410	11,691	11,829	*11,829*	10,836
Hchl4s	11,408	*11,408*	10,545	11,165	11,258	*11,258*	9,573
Hchl6s	60,170	*60,170*	56,105	*60,170*	59,853	*59,853*	58,121
Hchl7s	62,459	*62,459*	60,384	61,660	62,845	*62,845*	60,683
Hchl8s	729	*729*	662	727	791	*791*	715
Deviation (%)		0.00	4.68	0.58		0.01	5.07
No. of optima		38	8	22		37	7

Note. Italic denotes that the solution value matches the optimal value.

The results in Tables 8 and 9 show that even for these large problems, the path-relinking method obtains high-quality solutions in very short computing times. It is able to outperform the best previous approach since it obtains 13 optimal solutions for both horizontal and vertical first cut problems, while HESGA obtains nine and five respectively. It is difficult to compare the running times of the two approaches since they were run on different machines. However, HESGA requires a computation time of one order of magnitude larger than the PR method.

Given that our GRASP with path relinking incorporates random elements, we measure its robustness across different executions. Specifically, we ran the method four extra times and report in Table 10 the number of optima achieved (Optima), and the average percent deviation in each run (Average Deviation). We consider PR(2,000,200) and PR(200,20) variants of our method and both sets of problems previously reported. Table 10 also includes the number of optima obtained with the five runs and the average deviation to optima with the best solution across the five runs (Overall). Comparing a single run with these latter values, we can measure the benefit in solution quality when running the method several times.

We see small variations across the five different executions reported in Table 10, thus assessing the robustness of our solution procedure. This is especially true when we ran the method for a large number of iterations (see PR version 2,000, 200). Note that as reported above, the average running time for

RIGHTSLINK

Table 8 Comparison of Algorithms on Large Problems (First Cut Horizontal)

	First cut horizontal				
Instance	Optimum	GRASP_Piece	GRASP_Strip	PR	HESGA
APT30	140,168	134,531	140,168	140,168	140,168
APT31	820,260	813,752	808,597	813,935	818,512
APT32	37,880	36,303	37,799	37,793	37,880
APT33	235,580	230,631	235,580	235,580	234,564
APT34	356,159	353,063	354,219	355,660	356,159
APT35	614,429	595,423	607,999	612,082	613,784
APT36	129,262	125,570	129,262	129,262	129,262
APT37	384,478	376,052	384,478	384,478	382,910
APT38	259,070	247,753	258,134	258,819	258,221
APT39	266,135	261,552	265,853	266,135	265,621
APT40	63,945	61,539	63,945	63,945	63,945
APT41	202,305	192,657	202,305	202,305	202,305
APT42	32,589	31,767	32,487	32,589	32,589
APT43	208,998	204,869	208,998	208,998	208,571
APT44	70,940	63,160	70,901	70,901	70,678
APT45	74,205	69,716	74,205	74,205	74,205
APT46	146,402	143,305	146,402	146,402	146,402
APT47	144,317	136,270	144,317	144,317	143,458
APT48	165,428	150,765	165,428	165,428	162,032
APT49	206,965	189,471	205,946	206,965	204,574
Deviation (%)		4.06	0.23	0.08	0.30
No. of optima		0	11	13	9
CPU time		0.17	0.74	1.18	13.53[#]

[#]Running times on a UltraSparc10 (250 MHz, 128 MB of RAM).

Table 9 Comparison of Algorithms on Large Problems (First Cut Vertical)

	First cut vertical				
Instance	Optimum	GRASP_Piece	GRASP_Strip	PR	HESGA
APT30	140,197	136,735	139,774	140,067	140,007
APT31	821,073	795,354	809,109	812,553	818,296
APT32	37,973	36,402	37,973	37,973	37,744
APT33	234,670	229,482	234,670	234,670	234,538
APT34	357,741	347,548	354,675	355,571	353,590
APT35	614,336	601,412	601,492	605,959	614,132
APT36	128,814	126,687	128,814	128,362	128,814
APT37	385,811	370,484	383,406	385,164	385,811
APT38	259,137	253,414	259,028	259,137	258,040
APT39	266,378	256,778	265,062	265,507	265,330
APT40	65,584	62,647	65,584	65,584	65,044
APT41	196,559	194,551	196,559	196,559	195,453
APT42	33,012	31,347	33,012	33,012	32,937
APT43	212,062	206,185	212,062	212,062	212,062
APT44	69,784	69,571	69,474	69,784	69,732
APT45	69,988	69,162	69,988	69,988	69,857
APT46	147,021	141,464	147,021	147,021	147,021
APT47	142,935	142,199	142,935	142,935	142,935
APT48	162,458	153,117	162,458	162,458	160,318
APT49	211,784	202,531	211,784	211,784	210,169
Deviation (%)		2.87	0.30	0.20	0.34
No. of optima		0	13	13	5
CPU time		0.17	0.89	1.34	14.42[#]

[#]Running times on a UltraSparc10 (250 MHz, 128 MB of RAM).

the PR(2,000, 200) version is 1 second, and for the PR(200, 20) 0.7 second.

In our last experiment we consider the ten instances by Wang and Valenzuela (2001) which can be obtained by request from the authors. They range from 25 to 500 pieces and are created to guarantee zero waste in the strip-packing problem. The authors generated two types of problems: *Nice*, with pieces similar in shape and size, and *Path*, with extreme (pathological) variations of shape and size among pieces. We have slightly modified the original data in order to have integer dimensions, multiplying by ten all lengths and widths and rounding piece sizes to the nearest integer. According to these modifications the initial stock rectangles are now of dimensions (1,000, 1,000). Table 11 shows the result of our path-relinking method when solving these instances five times. As in previous experiments, we consider both types of first cut, horizontal and vertical, and two versions of our method, PR(2,000, 200) and PR(200, 20). We have also run the CPLEX 9.0 method with the formulation given in Lodi and Monaci (2003) to obtain, if possible, the optimal solution of these instances.

Table 11 shows that the larger instances of this set are very hard. On the one hand, CPLEX solves to optimality only the smallest 25-piece instances of the *Nice* class, and instances of up to 100 pieces of the *Path* class, within the time limit of 1,800 seconds of CPU time. On the other hand, except for the smallest instance of each subgroup, the path-relinking

Table 10 Comparison of Five Different Executions of PR

			PR executions (No. of optima/average deviation)					
Instances	First cut	PR version	1	2	3	4	5	Overall
38 medium problems	H	200, 20	34/0.11	34/0.11	34/0.20	34/0.18	34/0.20	37/0.08
	V	200, 20	36/0.11	36/0.11	37/0.11	36/0.11	36/0.11	37/0.01
	H	2,000, 200	38/0.00	37/0.08	37/0.01	35/0.10	37/0.08	38/0.00
	V	2,000, 200	37/0.01	37/0.01	37/0.01	37/0.01	38/0.00	38/0.00
20 large problems	H	200, 20	10/0.25	12/0.21	8/0.41	11/0.28	10/0.26	15/0.10
	V	200, 20	12/0.25	12/0.25	11/0.24	9/0.43	12/0.21	13/0.19
	H	2,000, 200	14/0.08	13/0.11	11/0.10	14/0.08	13/0.08	15/0.05
	V	2,000, 200	13/0.20	13/0.15	14/0.17	13/0.18	12/0.18	15/0.11

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

RIGHTSLINK

Table 11 Comparison of Five Different Executions of PR and CPLEX on Wang and Valenzuela Instances

		Path relinking executions					CPLEX	
Instances	First cut	PR version	Average value	No. of different solutions	Average time	Best solution	Solution value	Time
nice_25	H	2,000, 200	860,829	1	0.9	860,829	860,829	0.9
		200, 20	859,305	2	0.8	860,829		
nice_50	H	2,000, 200	902,581	3	3.1	904,955	912,395	>1,800
		200, 20	896,748	3	1.7	903,163		
nice_100	H	2,000, 200	906,040	4	15.0	908,734	906,476	>1,800
		200, 20	901,391	5	7.9	904,875		
nice_200	H	2,000, 200	931,428	5	67.6	932,686	924,576	>1,800
		200, 20	927,983	5	30.2	931,598		
nice_500	H	2,000, 200	951,228	5	464.0	952,015	942,548	>1,800
		200, 20	950,455	5	132.6	953,197		
nice_25	V	2,000, 200	809,532	4	1.2	820,363	834,300	11
		200, 20	806,237	4	1.0	813,613		
nice_50	V	2,000, 200	838,229	4	3.3	842,696	854,425	>1,800
		200, 20	827,384	3	2.0	836,844		
nice_100	V	2,000, 200	910,888	2	12.7	911,958	904,360	>1,800
		200, 20	906,447	4	6.2	910,174		
nice_200	V	2,000, 200	933,488	5	62.5	935,290	919,253	>1,800
		200, 20	927,337	5	26.9	931,249		
nice_500	V	2,000, 200	958,691	5	492.2	960,767	935,255	>1,800
		200, 20	955,526	4	140.3	956,532		
path_25	H	2,000, 200	892,765	1	1.1	892,765	892,765	0.1
		200, 20	892,765	1	0.6	892,765		
path_50	H	2,000, 200	746,401	4	2.9	748,655	750,215	645
		200, 20	736,842	5	1.5	746,767		
path_100	H	2,000, 200	868,883	5	14.3	875,592	888,578	>1,800
		200, 20	866,149	5	4.3	876,366		
path_200	H	2,000, 200	827,282	5	45.4	835,895	888,109	>1,800
		200, 20	814,199	5	15.7	833,343		
path_500	H	2,000, 200	879,392	2	420.0	879,465	877,327	>1,800
		200, 20	877,523	5	125.4	881,801		
path_25	V	2,000, 200	699,707	1	0.8	699,707	699,707	0.4
		200, 20	699,707	1	0.6	699,707		
path_50	V	2,000, 200	912,790	3	2.7	915,745	924,908	13.4
		200, 20	913,552	3	1.1	915,745		
path_100	V	2,000, 200	659,332	5	5.3	665,339	754,753	138.6
		200, 20	656,188	5	1.6	667,404		
path_200	V	2,000, 200	786,479	5	37.2	803,521	834,637	>1,800
		200, 20	761,357	4	14.1	783,279		
path_500	V	2,000, 200	806,181	3	434.0	808,376	769,278	>1,800
		200, 20	805,155	2	136.8	805,264		

algorithm produces a different solution at virtually each run. If we compare CPLEX with the path-relinking algorithm we see that, for the *Nice* class, CPLEX obtains better solutions for only the 25 and 50 pieces and it is clearly outperformed by path-relinking on larger instances. On the *Path* class, CPLEX obtains better results, especially for 200-piece instances, though the heuristic algorithm is better at 500-piece instances.

As expected, in small problems CPLEX is able to obtain the optimal solutions, thus improving or matching the best solution obtained by the heuristic method. However, in large problems, the size of the corresponding integer linear problem is so large that CPLEX explores only a small fraction of the branch-and-bound tree within the time limit imposed. In those cases, the heuristic solution clearly improves the best solution found with CPLEX.

6. Conclusions

We have presented two GRASP methods for the constrained two-dimensional two-stage 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

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

RIGHTSLINK

instances compared with the best previous approach for this problem and with the latest version of the CPLEX solver. 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.

A possible way to adapt the proposed algorithm within an evolutionary method, such as scatter search or genetic algorithms, would be to construct the initial population with the GRASP methods and then either apply the path-relinking or develop other combination methods. This straightforward implementation could be considered a starting point for future developments.

Acknowledgments
The last author's research is partially supported by the visiting professor fellowship program of the *Agencia Valenciana de Ciencia y Tecnología* (Grant Ref. CTESIN/2003/022). The first three authors are partially supported by the *Ministerio de Ciencia y Tecnología* (Grant Refs. DPI2005-04796 and TIN2006-02696) and by the *Agencia Valenciana de Ciencia y Tecnologia* (GRUPOS2003/174-189).

References

Alvarez-Valdes, R., A. Parajon, J. M. Tamarit. 2002. A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Comput. Oper. Res.* **29** 925–947.

Beasley, J. E. 1985. Algorithms for unconstrained two-dimensional guillotine cutting. *J. Oper. Res. Soc.* **36** 297–306.

Belov, G., G. Scheithauer. 2003. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. Technical Report MATH-NM-03, Institute of Numerical Mathematics, Dresden University, Dresden, Germany.

Dyckhoff, H. 1990. A typology of cutting and packing problems. *Eur. J. Oper. Res.* **44** 145–159.

Festa, P., M. G. C. Resende. 2001. GRASP: An annotated bibliography. M. G. C. Resende, P. Hansen, eds. *Essays and Surveys in Metaheuristics*. Kluwer Academic Publishers, Boston, MA, 325–367.

Gilmore, P. C., R. E. Gomory. 1961. A linear programming approach to the cutting stock problem. *Oper. Res.* **9** 849–859.

Gilmore, P. C., R. E. Gomory. 1965. Multistage cutting problems of two and more dimensions. *Oper. Res.* **13** 94–119.

Glover, F. 1994. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Appl. Math.* **49** 231–255.

Glover, F., M. Laguna. 1997. *Tabu Search*. Kluwer Academic Publishers, Boston, MA.

Hifi, M. 2001. Exact algorithms for large-scale unconstrained two and three staged cutting problems. *Comput. Optim. Appl.* **18** 63–88.

Hifi, M., R. M'Hallah. 2006. Strip generation algorithms for constrained two-dimensional two-staged cutting problems. *Eur. J. Oper. Res.* **172** 515–527.

Hifi, M., C. Roucairol. 2001. Approximate and exact algorithms for constrained (un)weighted two-dimensional two-staged cutting stock problems. *J. Combin. Optim.* **5** 465–494.

Laguna, M., R. Martí. 1999. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. Comput.* **11** 44–52.

Laguna, M., R. Martí. 2003. *Scatter Search—Methodology and Implementations in C*. Kluwer Academic Publishers, Boston, MA.

Lodi, A., M. Monaci. 2003. Integer linear programming models for 2-staged two-dimensional knapsack problems. *Math. Programming Ser. B* **94** 257–278.

Martello, S., P. Toth. 1990. *Knapsack Problems. Algorithms and Computer Implementations*. Wiley, Chichester, UK.

Morabito, R., V. Garcia. 1998. The cutting stock problem in the hardboard industry: A case study. *Comput. Oper. Res.* **25** 469–485.

Resende, M. G. C., C. C. Ribeiro. 2001. Greedy randomized adaptive search procedures. F. Glover, G. Kochenberger, eds. *State-of-the-Art Handbook in Metaheuristics*. Kluwer Academic Publishers, Boston, MA, 219–250.

Sweeney, P. E., E. R. Paternoster. 1992. Cutting and packing problems: A categorized applications-oriented research bibliography. *J. Oper. Res. Soc.* **43** 691–706.

Wang, P. Y., C. L. Valenzuela. 2001. Data set generation for rectangular placement problems. *Eur. J. Oper. Res.* **134** 378–391.

Downloaded from informs.org by [144.214.89.16] on 18 September 2017, at 07:51 . For personal use only, all rights reserved.

RIGHTSLINK