GRASP/VNS hybrid for the Strip Packing Problem

Jesús David Beltrán, Jose Eduardo Calderón, Rayco Jorge Cabrera, José A. Moreno Pérez and J. Marcos Moreno-Vega
Departamento de E.I.O. y Computación
Escuela Técnica Superior de Ingeniería Informática,
Universidad de La Laguna. 38271 La Laguna SPAIN.
e-mail: jamoreno@ull.es, jmmoreno@ull.es

April 23, 2004

Abstract

In this work we consider and hybrid metaheuristc approach to solve the Strip Packing Problem. The hybrid combines features of the GRASP and the VNS heuristics. We experimentally compare two kind of stopping rules: dependent and independent of the problem, and compare them.

Keywords: Metaheuristics, GRASP, VNS, Strip Packing Problem

1 Introduction

Metaheuristic searches are general solution procedures for solving problems. They are characterized by their wide applicability and by their good performance in many cases. Thus, ideally, they provide high quality solutions consuming moderate resource (time).

In this work we describe ...... We formulate ...... for the Strip Packing Problem (SPP) and compare them. Consider a strip of fixed width w and infinite height, and a finite set of rectangles, with at least one of their sides less than w. The SPP consists of packing the rectangles in the strip minimizing the height of the packing (see figure 1). In this problem the rectangles can be rotate at 90 degrees and the cuttings can be non guillotine. A cutting is guillotine if it goes from a side of the object to the front side. In a non-guillotine cut this can not be true.

The rest of the paper is structured as follows. In the section 3 we formulate the Strip Packing Problem ...... In the next section we describe the Metaheuristics ...... In section 5 we do an experimental comparative study of the metaheuristic searches to solve the problem. Finally, in the section 6 we enumerate the conclusions of our computational experience.

2 The Strip Packing Problem

The *Strip Packing Problem* (SPP) [8] is formulated as follows. Let w be the width of the strip with infinite height and let $\mathcal{R} = \{R(w_i, h_i) : i = 1, \ldots, n\}$ be a set of n rectangles. Each rectangle $R(w_i, h_i)$ has width w_i and height h_i verifying $\min\{w_i, h_i\} \leq w$, for each $i = 1, \ldots, n$.

The problem is to establish the optimal placing (r_i, a_i, b_i) of the rectangles $R(w_i, h_i)$, $i = 1, \ldots, n$. Here r_i is a binary variable that represents if the i-th rectangle is rotated or not, and (a_i, b_i) are the coordinates of the position of the bottom-left corner of the rectangle $R(w_i, h_i)$ with respect to the origin of coordinates (this origin is fixed at the bottom-left corner of the strip). If $r_i = 1$ then the i-th rectangle is rotated at 90 degrees; otherwise $r_i = 0$ and it is not rotated. The placing of each rectangle is feasible if the following two conditions are verified: the rectangle is completely included in the strip and it does not overlap with any other rectangle already placed. Therefore,

Figure 1: Solution of a Strip Packing Problem, SPP).

- If $r_i = 0$ then $0 \leq a_i \leq w - w_i$, $0 \leq b_i$ and the position of no other rectangle $R(w_j, h_j)$ verifies $a_i < a_j < a_i + w_i$ and $b_i < b_j < b_i + h_i$.
- If $r_i = 1$ then $0 \leq a_i \leq w - h_i$, $0 \leq b_i$ and the position of no other rectangle $R(w_j, h_j)$ verifies $a_i < a_j < a_i + h_i$ and $b_i < b_j < b_i + w_i$.

The objective function to be minimized is the maximum height h reached by the rectangles that is computed by:

$$h = \max\{\max_{r_i=0}(b_i + h_i), \max_{r_i=1}(b_i + w_i)\}$$

Equivalently, the objective is to minimize the area of the strip used that is the product of its width w by the maximum height h; $h \cdot w$.

Some infinite set of equivalent feasible solutions could be obtained by shifting horizontally or vertically some rectangles when it is possible. To avoid it, only solutions obtained by introducing successively all the rectangles following a given placement strategy are considered. We use the following usual *bottom-left* strategy: each rectangle is placed at the deepest location and, within it, in the most left possible location. So, each feasible solution of the problem is determined by the order in which the rectangles are introduced in the strip. Therefore, the space of solutions consists of the permutations of the numbers $[1, \ldots, n]$ with a binary vector $[r_1, \ldots, r_n]$ that represents if each rectangle is rotated or not.

The solution represented in figure 3 is obtained by introducing the five rectangles labelled with numbers 1 to 5 in this order, following the above strategy. In this solution, it is possible that some rectangles have been rotated at 90 degrees. The figure shown may be the configuration corresponding to a partial solution where some rectangles are not placed yet. There we can see two important elements: the space closed by the rectangles 2, 3 and 4, that is an area wasted, and the shape of the upper contour of the set of rectangles, that constraints the possible location of the next rectangles. Observe that, if it is a complete solution, the area between the upper contour and the maximum height reached by a rectangle is also wasted see figure 3.

3 Greedy Randomized Adaptive Search Procedure

In a constructive method an element is iteratively added to an initially empty structure until a solution of the problem is obtained. The choice of the item to be included in the partial solution is based on one or several heuristic evaluations that measure the convenience of considering the item as belonging to the solution. The heuristic functions depend on the problem and also on the knowledge of the decision maker about the problem. If the evaluation of an element depends on the items already in the solution, the function and the method are adaptive.

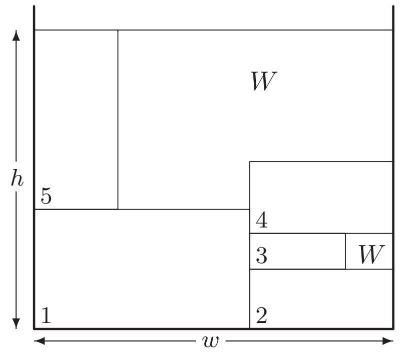


Figure 2: Total Waste of a solution with 5 packed rectangles

In addition to the heuristic function, it is necessary a strategy to select the elements. One of the most known strategy is the *greedy* rule: select the element that optimizes the heuristic function. However, this strategy shows poor performance in most cases. An alternative strategy is to randomly select one of the best elements. The set of best elements is called *Restricted Candidate List* (*RCL*).

GRASP (Greedy Randomized Adaptive Search Procedure) [?] is a constructive heuristic consisting of two phases. In the *constructive phase*, a solution is iteratively constructed randomly selecting one element of the restricted candidate list. Then, in the *post-processing phase*, it is attempted to improve this solution using a improved method (generally, a descent local search). These steps are repeated until a stopping rule is satisfied. The best overall solution is kept as the result.

The elements which completely determine a GRASP are: the heuristic function, the way in which restricted candidate list is built, the improved method and the stopping rule. In our proposal, the search terminates when a maximum number of local searches, n_{iter}, is reached. In the post-processing phase, the best placement of the k last rectangles into the solution is determined. For this, we use the VNS procedure.

3.1 Contour

La inclusión de un rectángulo cualquiera en el objeto, determina un contorno superior rectangular como el que se muestra en la figura ??. Además, es posible que se obtengan áreas no aprovechables, llamadas desperdicios, como el que se obtiene al incluir el rectángulo 4 en el objeto de la figura ??. El contorno, C, puede representarse por medio del conjunto de segmentos horizontales (tomados de izquierda a derecha) que lo forman. Es decir:

$$C = \{(y^1, x_1^1, x_2^1), (y^2, x_1^2, x_2^2), \ldots, (y^c, x_1^c, x_2^c)\}$$

con

$$\begin{array}{rcl} y^i & \equiv & \text{altura del } i\text{-ésimo segmento} \\ x_1^i & \equiv & \text{punto inicial del } i\text{-ésimo segmento} \\ x_2^i & \equiv & \text{punto final del } i\text{-ésimo segmento} \end{array} .$$

Además, $x_1^1 = 0$ y $x_2^c = w$. Nótese que, intuitivamente, es preferible un contorno formado por pocos niveles a otro con muchos niveles. Esto es así, ya que, en general, la posibilidad de obtener desperdicios aumenta con el número de niveles.

3.2 Restricted candidate list

Let t be the current iteration of the constructive phase and we suppose that $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$, where $\mathcal{R}_1$ is the set of rectangles into the partial solution and $\mathcal{R}_2 = \mathcal{R} \setminus \mathcal{R}_1$. Let $C(t)$ be the contour determined by $\mathcal{R}_1$. Evaluaremos la conveniencia de incluir un rectángulo de $\mathcal{R}_2$ en el objeto por la forma que tendrá el contorno $C(t)$ tras su inclusión. Las diferentes evaluaciones que proponemos pretenden aprovechar mejor el espacio disponible y suavizar el contorno.

- *Restricted candidate list:* let (y^i, x_1^i, x_2^i) be the lowest segment of the contour. Given $\alpha_1 \in [0,1]$, the restricted candidate list is built as:

$$\begin{aligned} RCL \;=\; & \{R(w_j,h_j) \in \mathcal{R}_2 : \\ & (0 \leq x_2^i - x_1^i - w_j \leq \alpha_1) \vee \\ & (0 \leq x_2^i - x_1^i - h_j \leq \alpha_1)\}. \end{aligned}$$

  The list is formed by the rectangles that better fit to the width of the lowest segment of the contour. The adjustment is given by the value of the parameter α_1.

If the list $\mathcal{RCL}$ is empty, we take from $\mathcal{R}_2$ the rectangle that better fits to (y^i, x_1^i, x_2^i). If such rectangle does not exist, we rebuild the contour $C(t)$ replacing the segments (y^i, x_1^i, x_2^i) and $(y^{i+1}, x_1^{i+1}, x_2^{i+1})$ by a new segment $(y^{i+1}, x_1^i, x_2^{i+1})$. The area between x_1^i, x_2^i, y^i and y^{i+1} is wasted.

4 Variable Neighbourhood Search

Variable neighborhood search (VNS) [6], [7] is a recent metaheuristic based on systematic change of the neighbourhood in an heuristic search. Usual heuristic searches are based on transformations of solutions that determine a neighborhood structure on the solution space. First VNS algorithms where based upon a simple idea: change the neighborhood structure when a local search is trapped on a local minimum.

An VNS can be implemented by the combination of series of random and improving (local) searches. The random and the improving searches are usually based on neighbourhood structures. When the improving local search stops at a local minimum, a shake procedure performs a random search for a new starting point for a new local search. The improving local search and the random shake procedure are usually based on a standard moves that determines the neighbourhood structures. A basic local search consists of applying an improving move until no such move exists. A simple shake procedure consists of applying a number of random moves. Many extensions have been made, mainly to allow solution of large problem instances. In most of them, an effort has been made to keep the simplicity of the basic scheme.

A *neighborhood structure* on the space solution X is function $\mathcal{N} : X \to 2^X$ that associates, to each solution $x \in X$, a neighborhood of solutions $\mathcal{N}(x) \subset X$, that are named *neighbors* of X. Let us denote with $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, a finite set of neighborhood structures on the solution space X. Neighborhoods $\mathcal{N}_k$ may be induced from one or more metric functions introduced into the solution space X. An improving local search changes the current solution by another better solution in its neighborhood, therefore it is trapped in a local minimum.

A local search iteratively seeks for a better solution in the neighborhood of the current solution. The classic greedy descent search consists in replacing the current solution by the best solution of its neighborhood, while an improvement is obtained. The *Variable Neighborhood Descent* (VND) method is obtained if change of neighborhoods is performed each time a local optimum (minimum) is reached. The steps of the VND with the greedy strategy are presented on next Figure.

The final solution provided by the algorithm should be a local minimum with respect to all k_{max} neighborhoods, and thus chances to reach a global one are larger than by using a single structure.

The *Reduced Variable Neighborhood Search* (RVNS) method is obtained if random points are selected from $\mathcal{N}_k(x)$, without being followed by descent, and its steps are presented on the next figure

The *Basic Variable Neighborhood Search* (BVNS) method combines random moves and improving (local) searches. Its steps are given on the next figure.

The Local search step (2b) of the BVNS may be replaced by VND. The resulting search method applies two (possibly different) series of neighborhoods; one for the shaking and one for the descent. The steps of this *General Variable Neighborhood Search* (GVNS) method are shown in next figure.

```
Initialization
    Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \dots, k_{max}$, that will be used in the descent;
    Find an initial solution $x$;
Iterations
    Repeat the following sequence until no improvement is obtained:
    (1) Set $k \leftarrow 1$;
    (2) Repeat the following steps until $k = k_{max}$:
        (a) Exploration of neighborhood
            Find the best solution $x'$ in the $k$-th neighborhood of $x$ ($x' \in \mathcal{N}_k(x)$);
        (b) Move or not
            If the solution thus obtained $x'$ is better than $x$, set $x \leftarrow x'$ and $k \leftarrow 1$; otherwise, set
            $k \leftarrow k + 1$.
```

Figure 3: Steps of the basic VND

The stopping condition may be e.g. maximum CPU time allowed, maximum number of iterations, or maximum number of iterations between two improvements.

The GVNS led to the most successful applications of VNS metaheuristic recently reported. Several of those applications use the same set of neighborhood structures for shaking and descent. The neighbourhoods are often nested and based on a single standard move. Given a base neighborhood structure $\mathcal{N}$ the series of nested neighborhood structure is defined as follows. The first neighbourhoods are the base ones, $\mathcal{N}_1(x) = \mathcal{N}(x)$ and the next neighborhoods are defined recursively by:

$$\mathcal{N}_k(x) = \bigcup_{y \in N_1(x)} \mathcal{N}_{k-1}(y).$$

In other words $\mathcal{N}_k(x)$ consists of the solutions that can be obtained from x by a series of k base moves.

Then the VNS comprises the following steps:

VNS algorithm

1. Initialization:

   Find an initial solution x. Set $x^* = x$ and $k = 1$.

2. Repeat the following until the stopping condition is met:

   (a) Shake:
   Generate at random a solution x' in $\mathcal{N}_k(x)$.

   (b) Local Search:
   Apply a local search in $\mathcal{N}(x')$ until a local minimum x'' is found.

   (c) Improve or not:
   If x'' is better than x^*, do $x^* = x''$ and $k = 1$. Otherwise do $k = k + 1$. Set $x = x^*$.

If the local search uses the greedy strategy then at step 2b an iterative procedure tests all the base moves, and that providing the best solution is made until no improving move exists The shake consists of applying a number k of random base moves. This number of random moves is the size of the shake. The base move for problems where the solutions are represented by permutations is the interchange move that consists in the interchange of the position of two elements of the permutation.

The random generation of solutions is performed by successively selecting one of the remainder elements with the same probability.

```
Initialization
    Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \dots, k_{max}$, that will be used in the search;
    Find an initial solution $x$;
    Choose a stopping condition;
Iterations
    Repeat the following sequence until no improvement is obtained:
    (1) Set $k \leftarrow 1$;
    (2) Repeat the following steps until $k = k_{max}$:
        (a) Shaking
            Generate a point $x'$ at random from the $k$-th neighborhood of $x$ ($x' \in \mathcal{N}_k(x)$);
        (b) Move or not
            If this point $x'$ is better than the incumbent $x$, move there $x \leftarrow x'$ and continue the search
            with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$.
```

Figure 4: Steps of the Reduced VNS

5 The Hybrid Approach. Variable Neighborhood Search as Post-processing phase in GRASP

In figure **??** is shown one of the anomalies which can occurs when the previous constructive method is applied. Let us consider the position of the rectangle 6. The method pack it as figure **??** indicates. The goodness of this new situation depends on instant in which it occurs. In the first iterations, it is recommendable. However, in the last iteration, it is preferable packing it as figure **??** shows. Thus, we propose the following improvement method.

Improvement method: extract the last k (parameter) rectangles of the solution. Let us assume, by simplicity, that are $\{R_1, R_2, \dots, R_k\}$. Apply VNS on the solution space consisting of all the permutations of the rectangles $\{R_1, R_2, \dots, R_k\}$. Return the best obtained solution.

6 Computational Experience

To analyze the performance of the proposed hybrid procedure we solve different instances of the *Strip Packing Problem.* The instances used are those corresponding to the categories $C_1, C_2, \dots, C_7$ of Hopper and Turton [8] (available at: `mscmga.ms.ic.ac.uk/jeb/orlib/stripinfo.html`). Each category consists of 3 instances with the following characteristics. The instances in C_1 have $n = 16$ or 17 rectangles, the strip has width $w = 20$ and the optimal height is $h_{opt} = 20$. The category C_2 consists of 3 instances with $n = 25$ rectangles, $w = 40$ and $h_{opt} = 20$. The instances of C_3 have 28 or 29 rectangles, width $w = 20$ and optimal height $h_{opt} = 20$. The three instances of C_4 have 49 rectangles and width and the optimal height are equal to 60. Category C_5 consists of 3 instances with $n = 72$ or 73, $w = 60$ and $h_{opt} = 90$. Category C_6 consists of 3 instances with $n = 97$ rectangles, the strip has width $w = 40$ and the height of the optimal solution is $h_{opt} = 20$. Finally, the category C_7 has 3 instances with $n = 196$ or 197 rectangles, $w = 160$ and $h_{opt} = 240$.

Tables show the results obtained with the VNS and GRASP metaheuristics and with the hybrid. In the first column is the category of instance in the Hopper and Turton repository. We consider the stopping rule by fixing four values for the percentage of total waste $W(X)$: 0.20%, 0.15%, 0.10% and 0.05%. The next four pairs of columns of the table show the corresponding counter and the objective

```
Initialization
    Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \dots, k_{max}$, that will be used in the search;
    Find an initial solution $x$;
    Choose a stopping condition;
Iterations
    Repeat the following sequence until no improvement is obtained:
     (1) Set $k \leftarrow 1$;
     (2) Repeat the following steps until $k = k_{max}$:
         (a) Shaking
             Generate a point $x'$ at random from the $k$-th neighborhood of $x$ ($x' \in \mathcal{N}_k(x)$);
         (b) Local Search
             Apply some local search method with $x'$ as initial solution; denote with $x''$ the so obtained
             local optimum;
         (c) Move or not
             If this local optimum $x''$ is better than the incumbent $x$, move there $x \leftarrow x''$ and continue
             the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$.
```

Figure 5: Steps of the Basic VNS

value when the solution obtained reach each one of these percentages. They are average values on the 3 instances in each category that have been solved 10 times with each metaheuritic. However, we stop the search when the counter is 1000 even if it has not reach the smallest percentage. Therefore, in several cases, the average values are obtained for less than 30 executions (this number is indicated between round brackets). The last column indicates the average objective function obtained with the independent of the problem stopping rule that uses the corresponding counter.

7 Conclusiones

We obtain the following conclusions from the analysis of the computational result:

1. The problem dependent and problem independent stopping rules are adequately formalized using fuzzy sets.

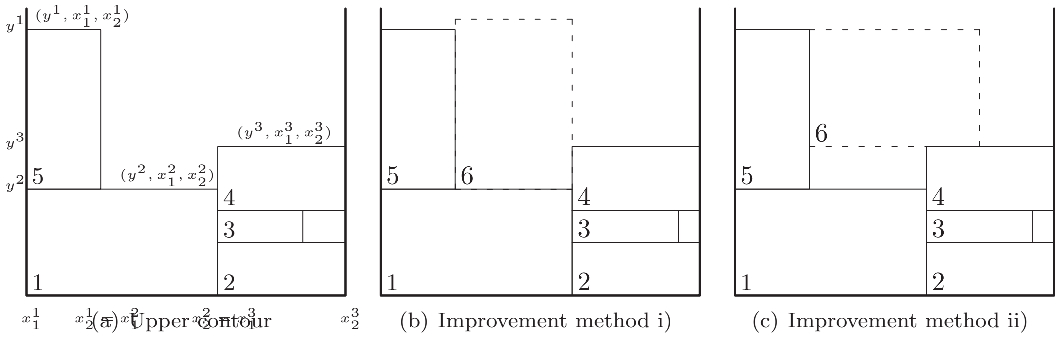
2. The problem dependent fuzzy stopping rule was more effective than the problem independent stopping rule for the same problem. With the problem independent stopping rules the executions of the same metaheuristics for the problems in the same category obtain very different objective values, even for the same problem, but with similar computational time. However, with the problem dependent stopping rules, the executions of each metaheuristics for the problems in each category obtain similar objective values but with different computational time.

3. As better is the metaheuristic more recommendable is the use of the problem dependent fuzzy stopping rule. In heuristics that provide low quality solutions, like the Pure Random Search, is unlike to find a solution that fulfill the stopping condition of the problem dependent fuzzy stopping rule. The opposite occurs for the GRASP, that can find solutions that fulfills the stopping condition of the problem dependent fuzzy stopping rule in very few iterations.

¿From the trade-off between the wasted areas among the rectangles and the wasted area between the upper contour and the maximum height it seems that for the high quality solutions, the interior waste is more important than the superior waste. This would be analyzed in a forthcoming research.

```
Initialization
    Select the set of neighborhood structures $\mathcal{N}_k$, for $k = 1, \dots, k_{max}$, that will be used in the shaking;
    Select the set of neighborhood structures $\mathcal{N}'_j$, for $j = 1, \dots, j_{max}$, that will be used in the descent;
    Find an initial solution $x$;
    Choose a stopping condition;
Iterations
    Repeat the following sequence until no improvement is obtained:
     (1) Set $k \leftarrow 1$;
     (2) Repeat the following steps until $k = k_{max}$:
         (a) Shaking
             Generate a point $x'$ at random from the neighborhood $\mathcal{N}_k(x)$;
         (b) Descent
             Apply the VND method with the neighborhood structures $\mathcal{N}'_j$, for $j = 1, \dots, j_{max}$; denote
             with $x''$ the so obtained local optimum;
         (c) Move or not
             If this local optimum $x''$ is better than the incumbent $x$, move there $x \leftarrow x''$ and continue
             the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$.
```

Figure 6: Steps of the General VNS



(a) Upper contour (b) Improvement method i) (c) Improvement method ii)

Figure 7: Upper contour and improvement method