Hindawi Publishing Corporation
Advances in Operations Research
Volume 2009, Article ID 732010, 20 pages
doi:10.1155/2009/732010

Research Article

An Exact Method for the 2D Guillotine Strip Packing Problem

Abdelghani Bekrar[1] and Imed Kacem[2]

[1] *Research and Development Department in Algorithmic, DynaSys S.A., Allee de Stockholm, 67300 Schiltigheim, France*
[2] *LITA Laboratory, University of Paul Verlaine Metz, Ile du Saulcy, 57000 Metz, France*

Correspondence should be addressed to Imed Kacem, kacem@univ-metz.fr

Received 24 August 2008; Revised 13 January 2009; Accepted 16 April 2009

Recommended by Mhand Hifi

We consider the two-dimensional strip packing problem with guillotine cuts. The problem consists in packing a set of rectangular items on one strip of width W and infinite height. The items packed without overlapping must be extracted by a series of cuts that go from one edge to the opposite edge (guillotine constraint). To solve this problem, we use a dichotomic algorithm that uses a lower bound, an upper bound, and a feasibility test algorithm. The lower bound is based on solving a linear program by introducing new valid inequalities. A new heuristic is used to compute the upper bound. Computational results show that the dichotomic algorithm, using the new bounds, gives good results compared to existing methods.

1. Introduction

The two-dimensional strip packing problem (2SP) is a well-known combinatorial optimization problem. It has several industrial applications like the cutting of rolls of paper or textile material. Moreover, some approximation algorithms for bin packing problems are in two phases where the first phase aims to solve a strip packing problem [1, 2]. Consider a set J of n rectangular items. Each item i has a width w_i and a height h_i ($i \in \{1, 2, \ldots, n\}$). The 2SP consists in packing all the items in a strip of width W and infinite height. The dimensions of the items and the strip are integers. The objective is to minimize the total height used to pack the items without overlapping. The orientation of items is fixed, that is, they cannot be rotated.

This problem is NP-hard in the strong sense since the special case where all items have the same height is equivalent to the one-dimensional bin packing [3, 4].

An additional constraint considered in this paper is the guillotine cut: All items must be extracted by cuts that go from one edge to the opposite edge. Figure 1(a) shows a guillotine pattern where all items can be extracted by guillotine cuts. When items cannot be extracted by guillotine cuts, the pattern is called nonguillotine as shown in Figure 1(b).

Most of papers considering the 2SP problem are approximation algorithms. Fernandez de la Vega and Zissimopoulos [5], Lesh et al. [6] Kenyon and Rémila [7], and Zhang et al. [8] presented approximation algorithms for the strip packing problem. Bortfeldt [9], and Beltran et al. [10] used metaheuristics. Hopper and Turton [11] provided an overview of metaheuristic algorithms applied to 2D strip packing problem.

To the best of our knowledge, there are only few papers which used exact algorithms to solve 2SP problem. Hifi [12] introduced the cutting/packing problem with guillotine cut, and he proposed two algorithms based on branch and bound. Martello et al. [4] proposed a new lower bound and used a branch and bound to solve the problem without guillotine constraint.

Recently, three papers introduced the strip packing problem with guillotine constraint. Cui et al. [13] proposed a recursive branch and bound to obtain an approximate solution. A new lower bound and a Branch and Bound were proposed by Bekrar et al. [14]. Finally, Cintra et al. [15] used the column generation method and dynamic programming to solve another variant of the problem (a bound on the number of each small rectangle to be packed is imposed).

In this paper, we propose an exact algorithm based on dichotomic search to solve the two-dimensional strip packing problem with guillotine cut. In Section 2 we present some lower bounds proposed in literature. In Sections 3 and 4, we, respectively, present the lower bound and the upper bound used in the algorithm. In Section 5, we explain the dichotomic algorithm. We provide computational results in Section 6. Finally, we discuss some perspectives of our work.

2. Lower Bounds for the 2SP Problem

In this section, we first recall *existing* lower bounds proposed for the strip packing problem: the continuous lower bound L_c, the lower bounds of Martello et al. [4], L_{mmv1} and L_{mmv2}, and our lower bound proposed in Bekrar et al. [16], L_{BKCS}. Note that, in all lower bounds the guillotine constraint is relaxed.

2.1. The Continuous Lower Bound L_c

By splitting each item into unit squares, we obtain a lower bound L_c which is called the *continuous lower bound*:

$$L_c = \left\lceil \frac{\sum_{i=1}^{n} h_i w_i}{W} \right\rceil \tag{2.1}$$

Let $L_0 = \max\{L_c, \max_{i=1,\dots,n} h_i\}$. Martello et al. [4] proved that the absolute worst case performance ratio is equal to $1/2$.

2.2. First Lower Bound L_{mmv1} (Martello et al. [4])

This first lower bound presented by Martello et al. [4] is an extension of the one presented in Martello and Toth [17] proposed for the one-dimensional bin packing problem. The idea is to decompose the set of items (J) into three subsets according to their dimensions: the subset of the largest items (J_1), the subset of medium items (J_2), and the subset of the smallest items (J_3).

Let $\alpha \in [1, W/2]$,
$J_1 = \{j \in J : w_j > W - \alpha\}$,
$J_2 = \{j \in J : W - \alpha \geq w_j > W/2\}$,
$J_3 = \{j \in J : W/2 \geq w_j \geq \alpha\}$.
Observe that

(i) beside a large item we cannot pack any item of the other subsets,

(ii) two items of the subset of medium items cannot be packed one beside the other, and

(iii) beside items of medium items only items of the last class can be packed.

The sum of heights of items in the two first classes is a valid lower bound for the 2SP problem, L_{J1J2}. Some items of J_3 cannot be packed beside the first class or the second class. A continuous lower bound on those items is computed and added to L_{J1J2}.

2.3. Second Lower Bound L_{mmv2} (Martello et al. [4])

The second lower bound proposed by Martello et al. [4] is based on a relaxation of the 2SP problem by cutting each item $j \in J$ into h_j unit-height *slices* of width w_j. The authors consider the *one-dimensional contiguous bin packing problem (1CBP)*, where all slices must be packed in bins of size W. The h_j slices derived from the item j must be packed into h_j contiguous bins. The optimal solution value of (1CBP) is a valid lower bound for the 2SP problem (denoted L_{mmv2}). This solution is computed by a Branch and Bound. The authors proved that L_{mmv2} dominates the previous bounds (L_c and L_{mmv1}).

When the Branch and Bound fails in determining the optimal solution within a fixed time, a new instance of (1CBP) is created from the 2SP instance by cutting each item $j \in J$ into $\lfloor h_j/2 \rfloor$ slices of height 2 or into $\lfloor h_j/3 \rfloor$ slices of height 3, and so on, until a solvable (1CBP) instance is produced. L_{mmv2} is improved by using a special binary search procedure (for more details see [4]).

This lower bound gives the best results on the tested instances, but as we can remark it is complicated and it takes considerable computation time.

2.4. Bekrar et al. [14] Lower Bound (L_{BKCS})

This lower bound is based on the same decomposition presented in Martello et al. [4]. The idea is to calculate the maximum number of items from the last class J_3 that can be packed beside items of the second class J_2. This allowed us to compute a best bound on the items of J_3 that cannot be packed with J_2.

This lower bound gives good results on the same instances of Martello et al. [4] with the advantage of reducing the computation time.

3. MIP Formulation and Introduction of Valid Inequalities

This method is based on an adapted formulation of the problem in a mixed integer programming. It is impossible to solve effectively this problem by a commercial solver because of the large number of constraints and variables. *In addition*, the linear relaxation (relaxation of the integrity constraint) is very poor. This is why we add a family of valid inequalities to the model to improve the lower bound associated to its linear relaxation.

3.1. The MIP Formulation for the Two-Dimensional Bin Packing Problem (Pisinger and Sigurd [18])

The model which we used is adapted from the one presented by Pisinger and Sigurd [18] for the two-dimensional bin packing problem. The latter is based on the modeling technique proposed by Onodera et al. [19] and Chen et al. [20] described as follows.

Let J denote the set of items. (W, H) are, respectively, the width and the length of the bin.

Decision Variables

Consider the following variables.

(i) (x_i, y_i), $forall\ i \in J$: the lower-left coordinates of item i.

(ii) m_i, $forall\ i \in J$: the index of the bin containing rectangle i.

(iii) $l_{ij} \in \{0,1\}$, $forall\ i, j \in J$ is equal to 1 if rectangle i is located left of rectangle j, that is, $l_{ij} = 1$ if $x_i + w_i \leq x_j$.

(iv) $b_{ij} \in \{0,1\}$, $forall\ i, j \in J$ is equal to 1 if rectangle i is located below rectangle j, that is, $b_{ij} = 1$ if $y_i + h_i \leq y_j$.

$$
(\text{MIP1}) \begin{cases}
\text{Min} & v & \\
s.\ t. & & \\
l_{ij} + l_{ji} + b_{ij} + b_{ji} + p_{ij} + p_{ji} \geq 1, & i, j \in J,\ i < j, & (1) \\
x_i - x_j + W l_{ij} \leq W - w_i, & i, j \in J, & (2) \\
y_i - y_j + H b_{ij} \leq H - h_i, & i, j \in J, & (3) \\
m_i - m_j + n p_{ij} \leq n - 1, & i, j \in J, & (4) \\
0 \leq x_i \leq W - w_i, & i \in J, & (5) \\
0 \leq y_i \leq H - h_i, & i \in J, & (6) \\
1 \leq m_i \leq v, & i \in J, & (7) \\
l_{ij}, b_{ij}, p_{ij} \in \{0,1\}, & i, j \in J, & (8) \\
x_i, y_i \in R, & i \in J, & (9) \\
m_i, v \in N, & i \in J, & (10)
\end{cases} \tag{3.1}
$$

(v) $p_{ij} \in \{0,1\}$, $forall\ i, j \in J$: $p_{ij} = 1$ if $m_i + 1 \leq m_j$, that is, $p_{ij} = 1$ if rectangle i is packed in a bin before the bin where rectangle j is packed (the index of the bin containing i is less than the bin containing j).

(vi) v is the number of used bins.

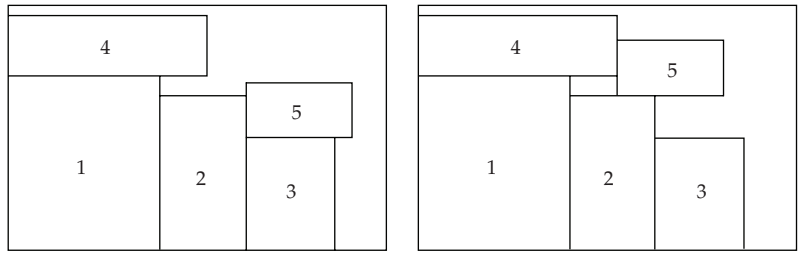


Figure 1: (a) Guillotine pattern, (b) nonguillotine pattern.

Constraints (1) avoids the overlapping between items, so item i should be in the left or the right side of item j, and, in the same time, it should be located above or below item j. In constraint (2), if $l_{ij} = 1$, we have: $x_j \geq x_i + w_i$, which means that item i is located left to item j. If $b_{ij} = 1$ in constraint (3), we deduce that item i is located below item j. In constraint (4), we determine if item i is packed "before" item j, $p_{ij} = 1$ (the index of the bin containing i is less than the index of the bin containing j). These three constraints determine the position of item i relative to the position of item j. Constraints (5) and (6) ensure that the items do not exceed the edges of the bins.

3.2. MIP Formulation for the Two-Dimensional Strip Packing Problem 2SP

We adapted the model of Pisinger and Sigurd [18] for the 2SP problem. Constraints (4), (7), and (10) are eliminated since they concern the bin packing problem. The length of the bin H is replaced by an upper bound H_h corresponding to the length of the strip.

The 2SP problem can be formulated in the following MIP model, denoted (MIP2):

$$
(\text{MIP2}) \begin{cases}
\text{Min} & HH & & \\
s.\,t. & & & \\
l_{ij} + l_{ji} + b_{ij} + b_{ji} \geq 1, & i, j \in J,\ i < j, & (1) & \\
x_i - x_j + W l_{ij} \leq W - w_i, & i, j \in J, & (2) & \\
y_i - y_j + H_h b_{ij} \leq H_h - h_i, & i, j \in J, & (3) & \\
0 \leq x_i \leq W - w_i, & i \in J & (4) & \\
0 \leq y_i \leq H_h - h_i, & i \in J & (5) & \\
HH \geq y_i + h_i, & i \in J & (6) & \\
l_{ij}, b_{ij} \in \{0, 1\}, & i, j \in J, & (7) & \\
HH, x_i, y_i \in \mathbf{R}, & i \in J. & (8) &
\end{cases} \tag{3.2}
$$

The valid inequalities are added to MIP2 in which we relax the integrity constraint (constraints (7) are replaced by the constraints $0 \leq l_{i,j} \leq 1$ and $0 \leq b_{i,j} \leq 1$, resp.).

In the following, we propose a series of valid inequalities to improve the lower bound of the 2SP problem.

3.3. Inequalities Related to the Parallel-Machine Scheduling Problem and y_i

This bound is inspired from the principle proposed in Kacem [21] applied to the tardiness minimization on a single machine. Kacem [21] associates fictitious weights to jobs to be scheduled and computes a lower bound on the weighted flow-time of the optimal solution. Therefore, for each vector of fictitious weights, he obtained a valid inequality. For the studied problem, we consider the items as a set of jobs (i.e., each item i is a set of w_i jobs of duration h_i) to be scheduled on W identical parallel machines. Hence, the variables $(y_i + h_i)$ represents the completion time of the item i (all the jobs corresponding to this item). Clearly, feasible packing can be viewed as a schedule of these fictitious jobs. To each item i, we associate a fictitious weight γ_i. We use an iterative method to calculate different values of weights. For each vector of fictitious weights, we obtain a valid inequality. We deduce the following valid inequality based on the lower bound of Eastman et al. [22] for the *problem of minimizing the weighted flow-time on identical parallel machines*$(Pm \parallel \sum w_i C_i)$:

$$\sum_{i=1}^{n} \gamma_i w_i (y_i + h_i) \geq \frac{1}{W} \sum_{i=1}^{n} \gamma_{[i]} \left(w_{[i]} \sum_{j=1}^{i-1} w_{[j]} h_{[j]} + \sum_{j=1}^{w_{[i]}} j h_{[i]} \right) + \frac{W-1}{2W} \sum_{i=1}^{n} \gamma_i w_i h_i, \tag{3.3}$$

where $[i]$ is the item in position i when the set J is sorted in non-decreasing order of h_i/γ_i. This constraint is valid for any positive vector of fictitious weight $\gamma = (\gamma_i)_{1 \leq i \leq n}$. We generated several constraints of this type.

Recall that the bound of Eastman et al. [22] for the $Pm \parallel \sum w_i C_i$ problem is given by the following equation:

$$LBM = \frac{LB1}{M} + \frac{M-1}{2M} \sum_{j=1}^{N} w_j p_j, \tag{3.4}$$

where $LB1$ is the optimal solution of the problem on a single machine. M is the number of machines, and N is the number of jobs. w_j and p_j are, respectively, the weight and the duration of the job j.

3.4. Inequalities Related to the Parallel-Machine Scheduling Problem and x_i

This inequality consists in the same principle. We consider items as jobs; each item i is considered as a set of h_i jobs of duration w_i. These jobs are to be scheduled on H_h identical machines. Therefore, the variables $(x_i + w_i)$ represent the completion time of the item i. It is obvious that any feasible packing can be viewed as a scheduling of fictitious jobs.

To each job i, a fictitious weight γ_i is associated. Hence, we deduce the following valid inequalities that used the bound of Eastman et al. [22] for the problem $Pm \parallel \sum w_i C_i$:

$$\sum_{i=1}^{n} \gamma_i h_i (x_i + w_i) \geq \frac{1}{H_h} \sum_{i=1}^{n} \gamma_{[i]} \left(h_{[i]} \sum_{j=1}^{i-1} w_{[j]} h_{[j]} + \sum_{j=1}^{h_{[i]}} j w_{[i]} \right) + \frac{H_h - 1}{2H_h} \sum_{i=1}^{n} \gamma_i w_i h_i, \tag{3.5}$$

where $[i]$ is the ith item when the set J is sorted in nondecreasing order of w_i/γ_i.

This constraint is valid for any vector $\gamma = (\gamma_i)_{1\leq i\leq n}$ of fictitious weights. For this reason, we generate several constraints of this family.

3.5. Inequalities Linking y_i and HH

By solving the model with the previous cuts (Sections 3.3 and 3.4), we noticed that in the obtained solutions items are in the bottom and in the right of the strip to respect these inequalities. To avoid this problem, we adopted the solution which consists in bounding the weighted sum of y_i.

We apply the same reasoning used in the previous inequality, and then we can obtain a valid inequality able to link y_i and HH. Indeed, by replacing ($y'_i = HH - y_i$, i.e., $y_i = HH - y'_i$) and using a similar notation as in the first family (Section 3.3), we can introduce the following inequality:

$$\sum_{i=1}^{n}\gamma_i w_i y'_i \geq \frac{1}{W}\sum_{i=1}^{n}\gamma_{[i]}\left(w_{[i]}\sum_{j=1}^{i-1}w_{[j]}h_{[j]} + \sum_{j=1}^{w_{[i]}}jh_{[i]}\right) + \frac{W-1}{2W}\sum_{i=1}^{n}\gamma_i w_i h_i, \tag{3.6}$$

or

$$H\sum_{i=1}^{n}\gamma_i w_i \geq \sum_{i=1}^{n}\gamma_i w_i y_i + \frac{1}{W}\sum_{i=1}^{n}\gamma_{[i]}\left(w_{[i]}\sum_{j=1}^{i-1}w_{[j]}h_{[j]} + \sum_{j=1}^{w_{[i]}}jh_{[i]}\right) + \frac{W-1}{2W}\sum_{i=1}^{n}\gamma_i w_i h_i. \tag{3.7}$$

3.6. Inequalities Bounding the Weighted Sum on x_i

This inequality is similar to the previous one. It is based on the following transformation of variables: $x'_i = W - x_i$. It is described as follows:

$$W\sum_{i=1}^{n}\gamma_i h_i \geq \sum_{i=1}^{n}\gamma_i h_i x_i + \frac{1}{H_h}\sum_{i=1}^{n}\gamma_{[i]}\left(h_{[i]}\sum_{j=1}^{i-1}w_{[j]}h_{[j]} + \sum_{j=1}^{h_{[i]}}jw_{[i]}\right) + \frac{H_h-1}{2H_h}\sum_{i=1}^{n}\gamma_i w_i h_i. \tag{3.8}$$

3.7. Inequalities for Large Items and High Items

For the set of large items such that each pair cannot be packed side by side, we can consider it as items of width W. Indeed, let $\mathcal{G} = \{i \mid forall\ j \in \mathcal{G}$ and $j \neq i$, we have $w_i + w_j > W\}$. The following constraints are valid inequalities:

$$\begin{aligned} &l_{i,j} + l_{j,i} = 0 \quad \forall\ i, j \in \mathcal{G},\ j \neq i, \\ &b_{i,j} + b_{j,i} = 1 \quad \forall\ i, j \in \mathcal{G},\ j \neq i \end{aligned} \tag{3.9}$$

Let $(\gamma_i)_{1\le i\le|\mathcal{G}|}$ be a vector of fictitious weights. $[i]_{1\le i\le|\mathcal{G}|}$ represents the ith item of the set $\mathcal{G}$ by sorting the items in nondecreasing order of h_i/γ_i. The following constraint defines a valid inequality:

$$\sum_{i\in\mathcal{G}}\gamma_i(y_i+h_i)\ge\sum_{i=1}^{|\mathcal{G}|}\gamma_{[i]}\left(\sum_{j=1}^{i}h_{[j]}\right). \tag{3.10}$$

Obviously, the following constraint is also a valid inequality:

$$HH\sum_{i\in\mathcal{G}}\gamma_i\ge\sum_{i\in\mathcal{G}}\gamma_i y_i+\sum_{i=1}^{|\mathcal{G}|}\gamma_{[i]}\left(\sum_{j=1}^{i}h_{[j]}\right) \tag{3.11}$$

We use the same reasoning in the previous cut. Let $\mathcal{G}' = \{i \mid forall\ j \in \mathcal{G}'$and $j\neq i,$ we have $h_i+h_j>H_h\}$. The following constraints are valid inequalities:

$$\begin{aligned} l_{i,j}+l_{j,i}&=1 \quad \forall\ i,j\in\mathcal{G}',\ j\neq i,\\ b_{i,j}+b_{j,i}&=0 \quad \forall\ i,j\in\mathcal{G}',\ j\neq i. \end{aligned} \tag{3.12}$$

Let $(\gamma_i)_{1\le i\le|\mathcal{G}'|}$ be a vector of fictitious weights. $[i]_{1\le i\le|\mathcal{G}'|}$ represents the ith item in the set $\mathcal{G}'$ when it is sorted in the nondecreasing order of w_i/γ_i. The following constraint is a valid inequality:

$$\sum_{i\in\mathcal{G}'}\gamma_i(x_i+w_i)\ge\sum_{i=1}^{|\mathcal{G}'|}\gamma_{[i]}\left(\sum_{j=1}^{i}w_{[j]}\right). \tag{3.13}$$

The following constraint is also a valid inequality:

$$W\sum_{i\in\mathcal{G}'}\gamma_i\ge\sum_{i\in\mathcal{G}'}\gamma_i x_i+\sum_{i=1}^{|\mathcal{G}'|}\gamma_{[i]}\left(\sum_{j=1}^{i}w_{[j]}\right). \tag{3.14}$$

3.8. Inequalities with Identical Fictitious Weights

It is obvious that all the previous cuts remain valid for the special case when the fictitious weights are equal to 1. However, it is more advantageous in this case to apply the *Shortest Processing Time (SPT)* rule instead of the Eastman bound.

We established a simple procedure to apply this rule to our problem in which we consider items as jobs.

3.9. Valid Constraints Avoiding the Overlapping of Items in the Bottom of the Strip

When we tested the previous cuts, we noticed in some obtained solutions that items overlap in the bottom of the strip ($y_i = 0$) while the sum of their widths exceeds the width of the strip W. Therefore, we introduced a special cut to avoid this situation for which a constraint of the initial problem is violated.

Let $\mathcal{S}$ be the real solution violated and $\mathcal{U} = \{i \mid y_i = 0 \text{ in } \mathcal{S}\}$. We have $\sum_{i\in\mathcal{U}} w_i > W$. We apply for the items of this set $\mathcal{U}$ a cut of the same type of the first one with fictitious weights equal to 1 (Section 3.3). Obviously, we do not use the bound of Eastman but the generalized SPT mentioned previously. Let $SPT(\mathcal{U})$ be the value of this bound calculated for the set $\mathcal{U}$. The following constraint is a valid cut:

$$\sum_{i\in\mathcal{U}} w_i(y_i + h_i) \geq SPT(\mathcal{U}). \tag{3.15}$$

We observed that, in some solutions, items overlap on the other sides of the strip. Using the same approach, we can see that this cut can be immediately generalized for the items on the top, on the right edge, and the left edge of the strip.

3.10. Valid Constraints Based on Precedence Considerations

For all $j \leq n$, the following relations hold due to the precedence considerations:

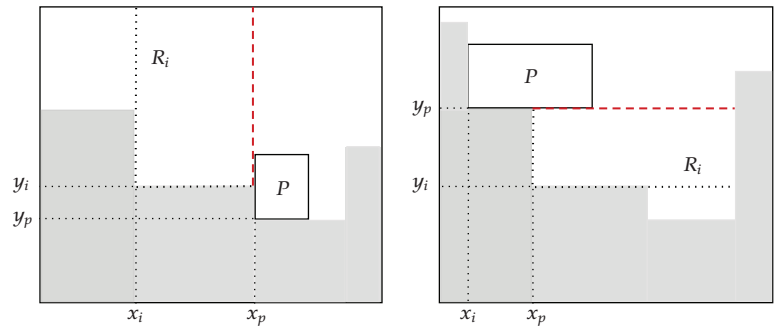
$$\begin{aligned}
y_j &\geq \sum_{i=1\cap i\neq j}^{i=n} \frac{h_i w_i b_{i,j}}{W},\\
x_j &\geq \sum_{i=1\cap i\neq j}^{i=n} \frac{h_i w_i l_{i,j}}{H_h},\\
y_j + h_j &\leq HH - \sum_{i=1\cap i\neq j}^{i=n} \frac{h_i w_i b_{j,i}}{W},\\
x_j + w_j &\leq W - \sum_{i=1\cap i\neq j}^{i=n} \frac{h_i w_i l_{j,i}}{H_h}.
\end{aligned} \tag{3.16}$$

4. Upper Bounds for the 2SP Problem

4.1. The Two-Dimensional Level Algorithms

Many papers proposed approximation algorithms for 2D Strip Packing Problem, but few of them consider the guillotine constraint, citing from this category: Fernandez de la Vega and Zissimopoulos [5], Lesh et al. [6], Kenyon and Rémila [7], Zhang et al. [8], and Cui et al. [13]

To maintain this constraint, most of the proposed approaches are level algorithms, that is, the strip packing is obtained by placing the items, from left to right, in rows forming levels. The set of items is sorted in decreasing order of their heights. The first item packed in the bottom-left of the strip initializes the first level. The remaining items are packed in the



(a) The item is located in the right side of R_i

(b) The item is located in the left side of R_i

Figure 2: Updating the set of available rectangles in $NSHF$.

initialized levels according to the used strategy (Best Fit (BFDH), First Fit (FFDH), Next Fit (NFDH), etc.). If no initialized level can contain the current item, a new level is initialized.

4.2. The Best Shelf Heuristic Filling (BSHF)

The Shelf Heuristic Filling (SHF) algorithm was proposed previously by Ben Messaoud et al. [23]. It is a generalization of the two-dimensional level algorithms (for more details about level algorithms like Floor Ceiling, see Lodi et al. [24]).

The main idea of SHF algorithm is to exploit the nonused area in each shelf. It makes it possible to pack items one over the other in the same shelf, which is not permitted in the level algorithms. This packing is achieved by respecting the guillotine constraint. For this purpose, SHF uses the definition of the available rectangle: the rectangle which has its bottom left corner as an *available point*. The available point can be the down-right or the top-left corner of an item already packed. Items are sorted in nonincreasing order of heights. The first item (the tallest one) is packed into the first available rectangle (the lowest). The leftmost item initializes a shelf with a height equal to the height of this item. After every item-packing, the set of available rectangles is updated in order to maintain the guillotine constraint. The updating consists in reducing the dimensions of available rectangles which overlap with the packed items. The item-packing creates two new available rectangles. This procedure is repeated until the last item is packed.

In Bekrar et al. [14], we proposed some strategies to improve the SHF algorithm. We called this new heuristic BSHF (Best SHF). We tested different rules of sorting items. We used the Best Fit rule to pack items, and we proposed a new way to update the list of available rectangles. The Best Fit rule consists in packing items in the available rectangle that minimizes the unused surface. This algorithm gives good results on large instances in few seconds. The average waste rate is estimated at 9%.

4.3. A New Heuristic for Solving the 2D Strip Packing Problem

In this subsection we present a new heuristic that is able to solve approximately the 2SP problem with guillotine cuts. Unlike previous heuristics (BSHF and BFDH), the items are not packed according to levels.

The items are sorted in different orders (decreasing heights, decreasing widths, decreasing areas).

The items are then packed in available rectangle as it is done in BSHF. However, the set of available rectangles is not updated in the same manner. When the current item can fill in an available rectangle, we check if the obtained pattern is guillotine using the procedure proposed by Ben Messaoud et al. [23]. If this pattern is guillotine, then the packing is validated and the next item is treated; otherwise, we try with another available rectangle.

During the packing, no shelf is created, hence the name of the algorithm: Nonshelf Heuristic Filling. The heuristic is adapted to the two-dimensional bin packing problem; when the current item cannot be packed in the open bins, a new bin is initialized.

The updating procedure in the BSHF heuristic aims to maintain the guillotine property of the patterns. In the NSHF heuristic, the updating procedure aims to *correct* the dimension of the available rectangles that are overlapping with the packed item.

Let $R_i(x_i, y_i, w_i, h_i)$ be an available rectangle that has a width w_i and a height h_i and position (x_i, y_i). The bottom-left point coordinates of R_i are x_i and y_i. Let an item p with a width w_p and a height h_p be packed in the position (x_p, y_p). We distinguish two possibilities.

(1) *Case 1*. $x_p \geq x_i$, the item p is located in the right side of R_i. If p overlaps with R_i, (i.e., $y_p + h_p > y_i$) then the width of R_i is reduced (Figure 2(a)): $R_i(x_i, y_i, x_p - x_i, h_i)$.

(2) *Case 2*. $x_p \leq x_i$, the item p is located in the left side of R_i. If p overlaps with R_i (i.e., $x_p + w_p > x_i$), then the height of R_i is reduced (Figure 2(b)): $R_i(x_i, y_i, w_i, y_p - y_i)$.

5. The Dichotomic Algorithm

In this section we present an exact method to solve the problem to optimality. The principle of this method was introduced first by Hifi [12]. Its effectiveness has been shown on other packing problems.

The algorithm starts by computing the lower bound LB using the cutting plane method described in Section 3. An upper bound, UB, is then computed by the heuristic *BSHF* and *NSHF* described in Section 4. If the upper bound is equal to the lower bound, then this is the optimal solution. If it is not the case, we search the optimal length included in the interval $[LB, UB]$ for which the items can be packed.

To reduce the search space, we use a dichotomic search. The Dichotomic Algorithm is sketched in Algorithm 1. The principle of this method was exploited in a previous work of Bekrar et al. [14].

When a height S is chosen from the interval $[LB, UB]$, a decision problem is generated: could the set of items be packed in the bin of width W and height S? The problem of determining if a set of rectangles can be packed in a larger rectangle of fixed size is well known as the two-dimensional orthogonal packing problem (2OPP).

Several papers were interested in this problem. Fekete et al. [25] proposed an exact algorithm based on the graph theory. Clautiaux et al. [26, 27] proposed two approaches to solve 2OPP to optimality. The first one is a two-step algorithm where they determined the x-coordinates of items in the first step, then they checked the feasibility of the obtained configurations in the second step Clautiaux et al. [26]. In Clautiaux et al. [27], they used another approach based on constraint programming. The constraint programming approach was also used by Pisinger and Sigurd [18].

```
   input: n: number of items;
   listP: list of items with their dimensions;
   W: the width of the strip;
   TL: the limit of the algorithm running time;
   output: H, the optimal height to pack all items
 1//Functions;
 2UBF(n, listP, W) : the function computing the upper bounds;
 3LBF(n, listP, W) : the function computing the lower bounds;
 4OPP(n, S, W1, H1) : this procedure returns TRUE if the set of items S can be packed in
   a bin of width W1 and height H1;
 5//initialization;
 6LB := LBF(n, listP, W);
 7UB := UBF(n, listP, W);
 8if(LB == UB)then
 9    Return LB; STOP;
10else
11    if OPP (n, listP, W, LB) ==TRUEthen
12        Return LB; STOP;
13    else
14        while((LB < UB) and (Time < TL))do
15            if(UB − LB == 1)then
16                Return UB; STOP;
17            else
18                S := ⌊(UB + LB)/2⌋;
19                if OPP (n, listP, W, S) == TRUEthen
20                    UB := S;
21                else
22                    LB := S;
```

Algorithm 1: Dichotomic Algorithm (Bekrar et al. [14]).

We choose to use the last approach in our algorithm, because it maintains the guillotine constraint, and we have experimentally checked that it has less computational time.

The (2OPP) is solved by a constraint satisfaction program (CSP). Constraint Satisfaction Programming (CSP) is a field of Artificial Intelligence that looks to solve problems modeled by a set of constraints imposed on a finite set of variables. The set of variables is defined in a domain: a finite set of values for each variable. A solution to CSP is a complete assignment of variables satisfying all the constraints.

For each pair of items i, j the domain M_{ij} is associated. $M_{ij} = \{\text{left}, \text{right}, \text{below}, \text{above}\}$ determines the possible relative placements among which we should choose at least one. r_{ij} is the variable that determines the position of item i according to the position of item j. The different relations between items can be defined as

$r_{ij} \in M_{ij},\ i, j \in \{1, \ldots, n\}, i \neq j,$
$r_{ij} = \text{left} \Rightarrow x_i + w_i \leq x_j,\ i, j \in \{1, \ldots, n\}, i \neq j,$
$r_{ij} = \text{right} \Rightarrow x_j + w_j \leq x_i,\ i, j \in \{1, \ldots, n\}, i \neq j,$
$r_{ij} = \text{below} \Rightarrow y_i + h_i \leq y_j,\ i, j \in \{1, \ldots, n\}, i \neq j,$
$r_{ij} = \text{above} \Rightarrow y_j + h_j \leq y_i,\ i, j \in \{1, \ldots, n\}, i \neq j,$
$0 \leq x_i \leq W - w_i,\ i \in \{1, \ldots, n\},$
$0 \leq y_i \leq H - h_i,\ i \in \{1, \ldots, n\}.$
Initially all relations r_{ij} are set to *undefined*.

Table 1: Comparison of *BSHF* and *NSHF* heuristics and literature heuristics.

Name	n	W	LB	$Best_H$	$\frac{Best_H}{LB}$	$BFDH$	$\frac{BFDH}{LB}$	$XSHF$	$\frac{XSHF}{LB}$
SCPL1	110	425	49	57	1.16	62	1.26	54	**1.10**
SCPL2	120	127	127	165	1.29	170	1.33	154	**1.21**
SCPL3	84	225	156	181	1.16	182	1.16	178	**1.14**
SCPL4	102	356	59	67	**1.13**	77	1.30	67	**1.13**
SCPL5	102	165	129	147	1.14	148	1.14	140	**1.08**
SCPL6	56	657	19	24	1.26	28	1.47	22	**1.15**
SCPL7	139	357	121	126	**1.04**	197	1.62	135	1.11
SCPL8	156	475	64	77	1.20	78	1.21	71	**1.10**
SCPL9	117	175	90	102	1.13	99	**1.10**	99	**1.10**
Average			90.44	105.11	1.17	115.66	1.29	102.22	1.12

In each iteration of the algorithm, two rectangles i and j are considered, and a value is assigned to r_{ij} from M_{ij}. It is then checked whether a feasible assignment of coordinates exists by respecting the current relations r_{ij}. If the coordinate check fails, the algorithm backtracks. Otherwise a recursive call is done.

6. Computational Results

To evaluate the proposed algorithms, we compare them to some literature algorithms. The heuristics *NSHF* and *BSHF* are compared to the best of the algorithm proposed by Hifi [28] and *BFDH* heuristic (Chung et al. [1]). The dichotomic algorithm is compared with the branch and bound of Bekrar et al. [16] and the algorithm of Hifi [12].

All our algorithms were coded in C++ and tested on a Pentium Xeon with 2.7 GHz of RAM.

6.1. Computational Results for Heuristics

We test our heuristics *BSHF* and *NSHF* to literature heuristics. Hifi [12] proposed four heuristics: FIA, SIA, HC/FIA, and HC/SIA. We compare the best values of those algorithms to our heuristics on the large instances studied by Hifi [12].

In Table 1, we present the results. The following information is given:

(i) n and W: respectively, the number of items and the strip width,

(ii) LB: the value of the linear lower bound,

(iii) $Best_H$: the best value obtained by Hifi's heuristics (*FIA, SIA, HC/FIA, HC/SIA*),

(iv) *XSHF*: the best value obtained by our heuristics *NSHF* and *BSHF*

(v) *BFDH*: results of *BFDH* heuristic, and

(vi) A/LB: the ratio of each heuristic by the lower bound (A represents the value of solution obtained by the heuristic).

As we can see in Table 1, our algorithms allow to obtain satisfactory solutions on the tested instances. It improves generally the existing solutions. Note that Hifi's algorithm remains the best for instance (SCPL7). This confirms the interest of our heuristics.

6.2. Computational Results on Hifi's Instances

The first tests are carried out on the instances of Hifi [12]. The dichotomic algorithm is compared to the branch and bound of Bekrar et al. [14] and to the best-first search MVB algorithm of Hifi [12]. The 25 instances are of various sizes and are available at http://www.laria.u-picardie.fr/hifi/OR-Benchmark/Strip-cutting/Strip-cutting.html.

In Table 2, we give for each problem instance the following:

(i) the width W of the strip and the number of items n,

(ii) values of upper bound UB and lower bound LB computed by the methods described previously,

(iii) values of lower bound $L_{\min}$ computed by Hifi [12],

(iv) values of the optimal solution OPT, and

(v) the computational time of Hifi [12] T_h, that of our branch and bound T_b, and that of the dichotomic algorithm T_{Dich} (in seconds).

Note that we carry out our algorithms for one hour, and if the optimal solution was not found (marked by "—"), we take the best solution found. Table 2 shows that the dichotomic algorithm could optimally solve all the tested instances of Hifi [12] and outperform our previous branch-and-bound algorithm in computational time, and on only few instances from 25 the dichotomic algorithm was not better. Note that Hifi used a Sparc-Server20 (module 712 MP) to test his algorithm. For this reason, it is not possible to make a reliable comparison in terms of the computation time. However, we prefer to report these results for information completeness.

6.3. Computational Results on Martello's Instances

The second tests were carried out on a series of instances from literature. All instances are available at http://www.or.deis.unibo.it/research.html.

We compare the dichotomic algorithm to the branch and bound of Bekrar et al. [14]. For each problem, Table 3 gives the following information:

(i) problem name and values of n and W,

(ii) values of the our lower bound LB and values of the upper bound UB,

(iii) values of lower bound L_{mmv2} computed by Martello et al. [4],

(iv) values of the solutions: branch and bound ($S_{B\&B}$) and dichotomic algorithm (S_{Dich}), and

(v) the computation time of the branch and bound $T_{B\&B}$ and that of the dichotomic algorithm T_{Dich} (in seconds).

As we can see in Table 3, the dichotomic algorithm often outperforms the Branch and Bound both in the number of solved instances and in the computation time.

6.4. Computational Results on Randomly Instances

To further analyze the performance of the algorithms, we compare the branch-and-bound method and the dichotomic algorithm on instances randomly generated. We adapt the classes

Table 2: Results of exact algorithms on Hifi's instances.

Name	n	W	$L_{\min}$	LB	UB	OPT	T_{Dich}	T_h	T_b
SCP1	10	5	13	13	13	13	<0.01	<0.1	<0.01
SCP2	11	4	40	40	40	40	<0.01	1.2	<0.01
SCP3	15	6	14	14	19	14	403.83	19.4	47.07
SCP4	11	6	19	20	22	20	41.22	4.1	8.75
SCP5	8	20	20	20	20	20	<0.01	0.1	<0.01
SCP6	7	30	32	38	38	38	<0.01	3.4	<0.01
SCP7	8	15	12	13	15	14	0.03	1.4	0.23
SCP8	12	15	17	17	20	17	10.45	2.3	9.57
SCP9	12	27	68	68	69	68	0.01	0.1	0.14
SCP10	8	50	80	79	80	80	0.01	0.5	0.12
SCP11	10	27	47	47	49	48	0.12	0.7	14.89
SCP12	18	81	34	34	38	34	0.01	2.7	—
SCP13	7	70	42	50	56	50	0.1	16.8	2.38
SCP14	10	100	63	63	83	69	126.65	48.7	27.57
SCP15	14	45	34	34	35	34	445.94	165.7	—
SCP16	14	6	32	33	35	33	201.83	181.4	—
SCP17	9	42	34	39	39	39	<0.01	323.6	<0.01
SCP18	10	70	90	91	104	101	14.74	327.8	36.43
SCP19	12	5	25	26	26	26	<0.01	473.0	<0.01
SCP20	10	15	19	20	22	21	0.69	673.9	42.73
SCP21	11	30	135	135	145	145	12.34	2001.8	654.31
SCP22	22	90	34	34	43	34	1597.91	757.8	—
SCP23	12	15	28	33	35	35	10.03	1031.9	401.97
SCP24	10	50	105	105	123	114	5.26	5585.7	329.34
SCP25	15	25	36	36	41	36	116.11	2662.9	—

of instances proposed by Berkey and Wang [2] and Martello and Vigo [24] for the two-dimensional bin packing problem to the strip packing problem. These instances consist of ten classes of problems. For each class, there are 40 instances: 10 with 10 rectangles, 10 with 15 rectangles, 10 with 20 rectangles, and 10 with 25 rectangles. The first six classes have been proposed by Berkey and Wang [2]:

$j = 1, \ldots, n.\ n = 10, 15, 20, 25$,

Class I: w_j *and* h_j*uniformly random in* $[1, 10]$, $W = 10$;

Class II: w_j *and* h_j*uniformly random in* $[1, 10]$, $W = 30$;

Class III: w_j *and* h_j*uniformly random in* $[1, 35]$, $W = 40$;

Class IV: w_j *and* h_j*uniformly random in* $[1, 35]$, $W = 100$;

Class V: w_j *and* h_j*uniformly random in* $[1, 100]$, $W = 100$;

Class VI: w_j *and* h_j*uniformly random in* $[1, 100]$, $W = 300$.

The remainder four classes were inspired from Martello and Vigo [24]. The items are classified into four types:

Type 1: *wj uniformly random in* $[2/3W, W]$, *hj uniformly random in* $[1, W/2]$;

Type 2: *wj uniformly random in* $[1, W/2]$, *hj uniformly random in* $[2/3W, W]$;

Type 3: *wj uniformly random in* $[W/2, W]$, *hj uniformly random in* $[W/2, W]$;

Type 4: *wj uniformly random in* $[1, W/2]$, *hj uniformly random in* $[1, W/2]$.

Table 3: Results of exact algorithms on literature instances.

Name	n	W	L_{mmv2}	LB	UB	S_{Dich}	$S_{B\&B}$	T_{Dich}	$T_{B\&B}$
ht1	16	20	20	20	20	20	20	0.01	<0.01
ht2	17	20	20	20	23	20	20	67.45	TL
ht3	16	20	20	20	25	20	20	197.53	353.10
ht4	25	40	15	15	17	15	15	874.05	TL
ht5	25	40	15	15	16	15	15	571.65	TL
ht6	25	40	15	15	15	15	15	0.00	<0.01
ht7	28	60	30	30	33	31	33	2045.34	TL
ht8	29	60	30	31	36	32	34	2101.33	TL
ht9	28	60	30	30	30	30	30	0.00	<0.00
cgcut1	16	10	23	23	25	23	23	323.54	TL
cgcut2	23	70	63	64	82	67	72	2012.24	TL
cgcut3	62	70	636	637	714	647	676	TL	TL
gcut1	10	250	1016	1016	1016	1016	1016	0.00	<0.00
gcut2	20	250	1133	1133	1349	1284	1349	TL	TL
gcut3	30	250	1803	1803	1810	1810	1810	TL	TL
gcut4	50	250	2934	2934	3214	2956	3214	TL	TL
ngcut1	10	10	23	21	23	23	23	16.73	2.58
ngcut2	17	10	30	30	31	30	30	1112.61	TL
ngcut3	21	10	28	28	33	29	29	724.31	TL
ngcut4	7	10	20	17	20	20	20	0.10	0.01
ngcut5	14	10	36	36	37	36	36	120.12	0.01
ngcut6	15	10	31	30	36	31	31	1091.33	TL
ngcut7	8	20	20	20	20	20	20	0.00	<0.00
ngcut8	13	20	33	32	38	33	33	188.64	TL
ngcut9	18	20	49	49	59	51	53	1531.20	TL
ngcut10	13	30	80	58	80	80	80	970.02	TL
ngcut11	15	30	52	50	60	52	52	55.84	TL
ngcut12	22	30	87	87	96	87	87	21.04	TL
beng1	20	25	30	30	35	30	30	608.41	TL
beng2	40	25	57	58	60	59	60	TL	TL
beng3	60	25	84	85	89	88	89	TL	TL
beng4	80	25	107	108	112	111	112	TL	TL
beng5	100	25	134	134	138	138	138	TL	TL
beng6	40	40	36	37	39	38	39	3152.27	TL
beng7	80	40	67	67	72	70	72	TL	TL
beng8	120	40	101	101	108	108	108	TL	TL
beng9	160	40	126	126	130	130	130	TL	TL
beng10	200	40	156	156	158	158	158	TL	TL

The strip widths are equal to 100 for all these classes, while the items are as follows:
Class VII: *type 1with probability of* 70%, *type* 2, 3, 4 *with probability of* 10% *each*;
Class VIII: *type 2with probability of* 70%, *type* 1, 3, 4 *with probability of* 10% *each*;
Class IX: *type 3with probability of* 70%, *type* 1, 2, 4 *with probability of* 10% *each*;
Class X: *type 4with probability of* 70%, *type* 1, 2, 3 *with probability of* 10% *each*.

Table 4: Results obtained by dichotomic and branch-and-bound algorithms on random instances.

Problem			Bound		Dichotomic			B&B		
Class	W	n	LB	UB	S_{Dich}	$\#Opt$	T(sec)	S_{bb}	$\#Opt$	T (sec)
		10	35.50	36	35.50	10	0.01	35.50	10	112.32
		15	45.90	49.10	46.90	10	221.60	46.70	9	999.67
I	10	20	63.50	66.70	65	10	1112.25	65.70	6	2151.81
		25	82.10	84.80	84	10	1758.08	85	3	1854.33
		Avg	56.86	59.15	57.88	10.00	772.99	58.23	7	1279.53
		10	10.20	11.30	10.70	10	0.01	10.70	10	301.21
		15	16.40	19.40	17.10	10	137.78	18.30	5	1267.74
II	30	20	19.60	21.70	20.80	10	612.82	20.90	3	2921.28
		25	26.90	29.40	28.20	10	955.14	29.30	3	2663.98
		Avg	18.28	20.45	19.20	10.00	426.44	19.80	5.25	1788.55
		10	31.90	34.90	33.80	10	0.07	34.40	10	516.54
		15	147	153.60	147.50	10	171.80	149.30	5	1592.42
III	40	20	169	176.20	172.80	10	810.37	175.40	4	3004.14
		25	220.20	232.60	227.20	8	1008.53	232	2	1518.21
		Avg	142.03	149.33	145.33	9.50	497.69	147.78	5.25	1657.83
		10	37	41.70	40.80	10	1.01	40.80	10	960.26
		15	49.40	59.90	54.50	10	465.60	55.50	4	968.76
IV	100	20	60.80	67.60	63.60	10	854.63	68.20	4	1975.87
		25	84.60	98.80	90.60	6	937.73	97.20	4	2980.43
		Avg	57.95	67.00	62.38	9.00	564.74	65.43	3600 5.50	2381.27
		10	320.92	330	321.40	10	1.98	321.40	10	111.04
		15	403.50	434.40	419.50	10	319.54	421.50	5	812.22
V	100	20	569.90	593.00	581.10	8	1213.28	583	5	722.19
		25	624.17	694.67	665	1	3489.31	689.17	0	3600
		Avg	478.37	513.02	496.75	7.25	1256.03	503.77	5	1311.36
		10	108.10	122.70	118.30	10	0.17	118.30	10	429.45
		15	121.20	144.10	128.60	10	231.44	136.70	2	2512.21
VI	300	20	168.03	195.80	179.20	9	2012.20	193.50	1	3600
		25	220	261.60	248	1	3512.89	250.10	1	2851.1
		Avg	154.33	181.05	168.53	7.50	1439.17	174.65	3.50	2348.19
		10	337.30	338.70	338.40	10	3.56	338.40	10	5
		15	532	532.20	532.10	10	2.20	542.30	10	3.6
VII	100	20	731.30	738.10	738.10	10	466.15	738.10	10	332.8
		25	707.70	715.20	714.60	6	1735.13	710.80	7	1342.76
		Avg	577.08	581.05	580.80	9.00	551.76	582.40	9.25	421.04
		10	247.10	285.50	267.60	10	2.94	263.90	10	752.9
		15	396.32	435.10	421.80	10	1153.48	433.10	6	2821.2
$VIII$	100	20	507.50	571.75	559	7	3086.11	561.50	2	3600
		25	635.17	771	693.67	6	3475.83	701.70	1	1880.4
		Avg	446.52	485.52	485.52	8.25	1929.59	490.05	4.75	2263.63
		10	543.70	544.50	543.70	10	0.54	543.70	10	30.7
		15	871.20	871.20	871.20	10	0.10	871.20	10	59.8
IX	100	20	1155.50	1159.80	1155.50	10	0.01	1159.70	9	432.3
		25	1476.90	1479	1477	10	0.26	1479	8	754.1
		Avg	1011.83	1013.65	1011.85	10.00	0.23	1013.40	9.25	319.23

Table 4: Continued.

Problem			Bound		Dichotomic			B&B		
Class	W	n	LB	UB	S_{Dich}	#*Opt*	T(sec)	S_{bb}	#*Opt*	T (sec)
		10	163.40	174.90	170.60	10	0.12	170.5	10	343.2
		15	253.01	268.90	258.50	10	243.07	265.10	4	2500.3
X	100	20	345.60	378.40	365.10	10	1637.05	375	4	2871.2
		25	388.22	416	408.89	5	2581.66	419.67	1	3600
		Avg	287.56	309.55	300.77	8.75	1115.47	307.57	4.75	2328.68

In Table 3, we present the results obtained by testing the two algorithms on random generated instances (TL denotes the time limit). For each 10 instances, we present the average values of

(i) LB: lower bound,

(ii) UB: upper bound,

(iii) S_{Dich} and S_{bb}: the best solution obtained by each algorithm. Each run instance was carried out for 3600 seconds (if no optimal solution was achieved, we take the best one),

(iv) #*Opt*: the number of found optimum, and

(v) T: the average time spent for solving the instance.

The results shown in Table 4 confirm the results obtained previously. The dichotomic algorithm outperforms the branch-and-bound method in the number of optimal solutions found and in the average time to compute a solution.

7. Concluding Remarks

In this paper we considered the strip packing problem under the guillotine constraint. The main contribution consists in the elaboration of new tight lower and upper bounds. The upper bounds are based on new rules for solving the problem under the above constraint. The lower bounds are based on a linear formulation using a set of various valid inequalities with a connection to scheduling on parallel machines. Such bounds were very useful to build an efficient dichotomic method which we compared to an existing branch-and-bound method. Based on the experimental results, several concluding remarks are worthy to note.

On the instances of Hifi our dichotomic algorithm was able to solve all the instances generally in a shorter computation time compared to our previous branch-and-bound method. Indeed, for the same instances our previous branch-and-bound algorithm cannot solve 5 instances of them. The dichotomic algorithm was faster in 16 instances among the 25, where the branch-and-bound algorithm was more efficient only on a few instances (four instances). These results show the effectiveness of the introduction of the new valid inequalities and the new heuristics for computing the bounds. Moreover, our heuristics allow to improve the existing approximate solutions for several instances of the studied benchmark.

For the instances introduced by Martello et al. [4] the same observation remains valid. Here, one can note that the average gap between the solution obtained by the dichotomic algorithm and the lower bound is estimated to 1.5%, which represents a good performance on the various and difficult instances.

Finally, the performance of the dichotomic algorithm is confirmed on the random instances. Such an algorithm is more effective and rapid for solving some instances to optimality. In average, this algorithm yielded an optimal solution for 9 instances on 10, where the branch-and-bound algorithm solves only 6 instances on 10.

As for the bin packing problem, the instances of classes V, VI, and VIII were the most difficult to solve. For example, for Class VI, with 25 items the two algorithms were not able to solve more than only one instance on the 10 generated. The other classes are easier since our dichotomic algorithm yielded generally the optimal solution (classes I, II, III, IX) within a short computation time (less than 10 minutes in average). The branch-and-bound algorithm needs more time, and it is less effective to solve optimally the problem (6.6 on 10). The easiness of these instances can be explained by the fact that they are constituted of many small items, which enables the algorithms to compute tight bounds. The classes IX and X are composed of 70% of items having widths greater than the half of the bin width. The bounds computed for this type of instances are very tight. Indeed, the introduction of the inequalities concerning the high and large items (inequalities in Section 3.7) allowed us to obtain a good performance.

As future research, we aim to extend our approach to other variants of packing problems.

Acknowledgments

This work has been carried out when the authors are with the University of Technology of Troyes (Institue of Charles Delaunay). The research of the first author has been carried out during a doctoral study funded by the Regional Council of Champagne-Ardenne.

References

[1] F. R. K. Chung, M. R. Garey, and D. S. Johnson, "On packing two-dimensional bins," *SIAM Journal on Algebraic and Discrete Methods*, vol. 3, no. 1, pp. 66–76, 1982.

[2] J. O. Berkey and P. Y. Wang, "Two-dimensional finite bin-packing algorithms," *Journal of the Operational Research Society*, vol. 38, no. 5, pp. 423–429, 1987.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, Calif, USA, 1979.

[4] S. Martello, M. Monaci, and D. Vigo, "An exact approach to the strip-packing problem," *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 310–319, 2003.

[5] W. Fernandez de La Vega and V. Zissimopoulos, "An approximation scheme for strip packing of rectangles with bounded dimensions," *Discrete Applied Mathematics*, vol. 82, no. 1–3, pp. 93–101, 1998.

[6] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher, "Exhaustive approaches to 2D rectangular perfect packings," *Information Processing Letters*, vol. 90, no. 1, pp. 7–14, 2004.

[7] C. Kenyon and E. Rémila, "Approximate strip packing," in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, pp. 31–36, Burlington, Vt, USA, October 1996.

[8] D. Zhang, Y. Kang, and A. Deng, "A new heuristic recursive algorithm for the strip rectangular packing problem," *Computers & Operations Research*, vol. 33, no. 8, pp. 2209–2217, 2006.

[9] A. Bortfeldt, "A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces," *European Journal of Operational Research*, vol. 172, no. 3, pp. 814–837, 2006.

[10] J. D. Beltran, J. E. Calderon, R. J. Cabrera, J. A. Moreno Perez, and J. M. Moreno-Vega, "GRASP/VNS hybrid for the strip packing problem," in *Proceedings of the 1st International Workshop on Hybrid Metaheuristics (HM '04)*, pp. 79–90, Valencia, Spain, August 2004.

[11] E. Hopper and B. C. H. Turton, "A review of the application of meta-heuristic algorithms to 2D strip packing problems," *Artificial Intelligence Review*, vol. 16, no. 4, pp. 257–300, 2001.

[12] M. Hifi, "Exact algorithms for the guillotine strip cutting/packing problem," *Computers & Operations Research*, vol. 25, no. 11, pp. 925–940, 1998.

[13] Y. Cui, Y. Yang, X. Cheng, and P. Song, "A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1281–1291, 2008.

[14] A. Bekrar, I. Kacem, and C. Chu, "A comparative study of exact algorithms for the two dimensional strip packing problem," *Journal of Industrial and Systems Engineering*, vol. 1, no. 2, pp. 151–170, 2007.

[15] G. F. Cintra, F. K. Miyazawa, Y. Wakabayashi, and E. C. Xavier, "Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation," *European Journal of Operational Research*, vol. 191, no. 1, pp. 61–85, 2008.

[16] A. Bekrar, I. Kacem, C. Chu, and C. Sadfi, "A branch and bound algorithm for solving the 2D strip packing problem," to apprear in *International Journal of Production Development* .

[17] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete Applied Mathematics*, vol. 28, no. 1, pp. 59–70, 1990.

[18] D. Pisinger and M. Sigurd, "The two-dimensional bin packing problem with variable bin sizes and costs," *Discrete Optimization*, vol. 2, no. 2, pp. 154–167, 2005.

[19] H. Onodera, Y. Taniguchi, and K. Tamaru, "Branch-and-bound placement for building block layout," in *Proceedings of the 28th Conference on ACM/IEEE Design Automation Conference*, pp. 433–439, San Francisco, Calif, USA, June 1991.

[20] C. S. Chen, S. M. Lee, and Q. S. Shen, "An analytical model for the container loading problem," *European Journal of Operational Research*, vol. 80, no. 1, pp. 68–76, 1995.

[21] I. Kacem, "Lower bounds for tardiness minimization on a single machine with family setup times," *International Journal of Operations Research*, vol. 4, no. 1, pp. 18–31, 2007.

[22] W. L. Eastman, S. Even, and I. M. Isaacs, "Bounds for the optimal scheduling of n jobs on m processors," *Management Science*, vol. 11, no. 2, pp. 268–279, 1964.

[23] S. Ben Messaoud, *Caracterisation, modelisation et algorithmes pour des problemes de decoupe guillotine*, Ph.D. thesis, Universite de Technologie de Troyes, Troyes, France, 2004.

[24] A. Lodi, S. Martello, and D. Vigo, "Models and bounds for two-dimensional level packing problems," *Journal of Combinatorial Optimization*, vol. 8, no. 3, pp. 363–379, 2004.

[25] S. P. Fekete, J. Schepers, and J. C. van der Veen, "An exact algorithm for higher-dimensional orthogonal packing," *Operations Research*, vol. 55, no. 3, pp. 569–587, 2007.

[26] F. Clautiaux, J. Carlier, and A. Moukrim, "A new exact method for the two-dimensional orthogonal packing problem," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1196–1211, 2007.

[27] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim, "A new constraint programming approach for the orthogonal packing problem," *Computers & Operations Research*, vol. 35, no. 3, pp. 944–959, 2008.

[28] M. Hifi, "The strip cutting/packing problem: incremental substrip algorithms-based heuristics," *Pesquisa Operacional*, vol. 19, no. 2, pp. 169–188, 1999.