Journal of Combinatorial Optimization, 5, 465–494, 2001

Approximate and Exact Algorithms for Constrained (Un) Weighted Two-dimensional Two-staged Cutting Stock Problems

MHAND HIFI hifi@univ-paris1.fr
CERMSEM, Maison des Sciences Economiques, Université Paris 1 Panthéon-Sorbonne 106-112, boulevard de l'Hôpital, 75647 Paris cedex 13, France; PRiSM-CNRS URA 1525, Université de Versailles-Saint Quentin en Yvelines, 45 avenue des Etats-Unis, 78035 Versailles cedex, France

CATHERINE ROUCAIROL Catherine.Roucairol@prism.uvsq.fr
PRiSM-CNRS URA 1525, Université de Versailles-Saint Quentin en Yvelines, 45 avenue des Etats-Unis, 78035 Versailles cedex, France

Abstract. In this paper we propose two algorithms for solving both *unweighted* and *weighted* constrained two-dimensional two-staged cutting stock problems. The problem is called *two-staged cutting problem* because each produced (sub)optimal cutting pattern is realized by using two cut-phases. In the first cut-phase, the current stock rectangle is slit down its width (resp. length) into a set of vertical (resp. horizontal) strips and, in the second cut-phase, each of these strips is taken individually and chopped across its length (resp. width).

First, we develop an approximate algorithm for the problem. The original problem is reduced to a series of *single bounded knapsack problems* and solved by applying a *dynamic programming procedure*. Second, we propose an exact algorithm tailored especially for the constrained two-staged cutting problem. The algorithm starts with an initial (feasible) lower bound computed by applying the proposed approximate algorithm. Then, by exploiting dynamic programming properties, we obtain good lower and upper bounds which lead to significant branching cuts. Extensive computational testing on problem instances from the literature shows the effectiveness of the proposed approximate and exact approaches.

Keywords: combinatorial optimization, cutting problems, dynamic programming, single knapsack problem, optimality

1. Introduction

We study in this paper the problem of cutting a stock rectangle S of dimensions $L \times W$, into n small rectangles (called pieces) of values (weights) c_i and dimensions $l_i \times w_i$, $i = 1, \ldots, n$. Each piece i, $i = 1, \ldots, n$, is bounded by an upper value b_i, i.e., the number of occurrences of the i-th piece in a *cutting pattern* (feasible solution) does not violate the value b_i. We assume that l_i, w_i, $i = 1, \ldots, n$, L and W are nonnegative integers and we consider that all pieces have *fixed orientation*, i.e., a piece of length l and width w is different from a piece of length w and width l (when $l \neq w$). Furthermore, all applied cuts are of *guillotine* type, i.e., a horizontal or a vertical cut on a (sub)rectangle being a cut from one edge of the (sub)rectangle to the opposite edge which is parallel to the two remaining edges (see figure 1(a)).

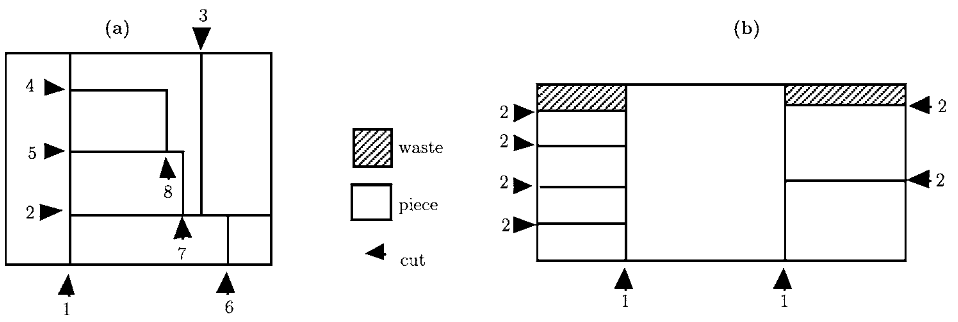

Figure 1. Representation of (a) the guillotine type and (b) a solution when k is fixed to 2; Numbers by the cuts are the stage at which the cut is made; the 2TDC problem: the value '1' corresponds to the first cut and '2' means that the second cut is applied; in this case they are 2 vertical cuts (counted one time) and 6 horizontal cuts (counted one time).

For the cutting stock problem, a large variety of approximate and exact approaches have been devised (see Dyckhoff, 1990).

The *constrained (un)weighted (non-staged) two-dimensional cutting* has been solved exactly by the use of tree-search procedures. Christofides and Whitlock (1977) have proposed a depth-first search method for solving the problem. Christofides and Hadjiconstantinou (1995) and Hifi and Zissimopoulos (1997) have separately proposed modified versions of Christofides and Whitlock's approach. In Viswanathan and Bagchi (1993), a generalized version of Wang's approach (Wang, 1983) has been proposed for both unweighted and weighted versions. The algorithm is principally based upon a best-first search method. In Hifi (1997), a modified version of Viswanathan and Bagchi's algorithm has been proposed and in Cung et al. (2000) a better version of the last algorithm has been developed.

The *unconstrained (un)weighted (non)staged version* of the problem has been solved exactly by applying dynamic programming procedures (Beasley, 1985; Gilmore and Gomory, 1966) and by the use of tree-search procedures (Herz, 1972; Hifi and Zissimopoulos, 1996). Gilmore and Gomory (1965) proposed an optimal procedure for solving the fixed unconstrained two-staged two-dimensional cutting stock problem. Several authors have used their procedure for solving some variants of TDC problems (see Hifi and Zissimopoulos, 1996; Fayard et al., 1998; Hifi, 1999; Morabito and Garcia, 1998). In Hifi (2001), a new exact algorithw has been proposed for solving unconstrained (un)weighted *fixed* and *rotated* two-staged two-dimensional cutting stock problems. In the same paper, another exact approach has been proposed and tailored especially for the unconstrained (un)weighted three-staged two-dimensional cutting stock problem.

Previous works have also proposed different heuristic approaches to both constrained and unconstrained versions of the problem (see Fayard et al., 1998; Hifi, 1999; Morabito et al., 1992; Morabito and Arenales, 1996; Arenales and Morabito, 1997).

2. The two-dimensional (staged) cutting problem

We say that the n-dimensional vector $(x_1, \ldots, x_n)$ of integer and nonnegative numbers corresponds to a *cutting pattern*, if it is possible to produce x_i pieces of type i, $i = 1, \ldots, n$, in the stock rectangle S without overlapping and without violating the upper values b_i. The constrained Two-Dimensional Cutting problem (shortly constrained TDC) consists of determining the cutting pattern (using guillotine cuts) with the maximum value, i.e.,

$$\text{constrained TDC} = \begin{cases} \text{max} & \displaystyle\sum_{i=1}^{n} c_i x_i \\ \text{subject to} & (x_1, \ldots, x_n) \leq (b_1, \ldots, b_n) \\ & \text{corresponds to a cutting pattern} \end{cases} \tag{1}$$

In constrained TDC problem one has, in general, to satisfy a certain demand of pieces, i.e., the upper values b_i, $i = 1, \ldots, n$. The problem is called *unconstrained* if these demand values are not imposed. In this case, the cutting pattern of Eq. (1) can be represented by $(x_1, \ldots, x_n) \leq (b'_1, \ldots, b'_n)$, where $b'_i = \lfloor \frac{L}{l_i} \rfloor \lfloor \frac{W}{w_i} \rfloor$, for $i = 1, \ldots, n$, which represents a natural upper bound-value for each considered piece.

We distinguish two versions of the constrained TDC problem: the *unweighted version* in which the weight c_i of the i-th piece is exactly its area, i.e., $c_i = l_i w_i$, and the *weighted version* in which the weight of each piece is independent of its area, i.e., there exists a piece ℓ for which $c_\ell \neq l_\ell w_\ell$, $1 \leq \ell \leq n$.

Another variant of the TDC problem is to consider a constraint on the total number of cuts, i.e., the sum of some parallel vertical and/or horizontal cuts (using guillotine cuts) does not exceed a certain constant $k < \infty$. In this case the problem is called the *staged* TDC problem or k-staged TDC problem (shortly kTDC). Figure 1(b) shows a 2TDC solution which is produced by applying the following phases:

Phase 1. the stock rectangle is slit down its width into a set of *vertical strips*;
Phase 2. each of these vertical strips is taken individually and chopped across its length.

For simplicity, for the constrained (un)weighted kTDC problem, we use the same notations as introduced in Hifi (2001):

- C_kTDC: corresponds to the *Constrained (un)weighted* kTDC problem. This notation is used if we consider (a) *unweighted* and *weighted* versions, and (b) the *fixed* and the *rotated* cases (in the rotated case, we consider that each piece can be turned of 90°).
- FC_kTDC: corresponds to the *Fixed Constrained (un)weighted* kTDC problem. This notation is used when we refer to *unweighted* and *weighted* versions of the problem;
- FCU_kTDC: represents the *Fixed Constrained Unweighted* kTDC problem;
- FCW_kTDC: corresponds to the *Fixed Constrained Weighted* kTDC problem.

Definition 1. Let (L, β) (resp. (α, W)) be the dimensions of a strip entering in the stock rectangle S. Then,

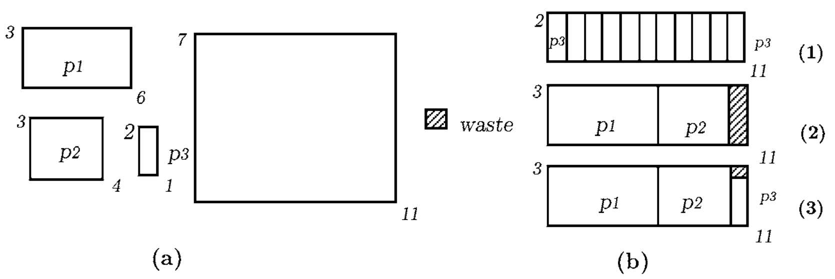

Figure 2. Representation of (**a**) a stock rectangle with three pieces to cut and (**b**) three different strips; **(b.1)** and **(b.2)** are two strips without trimming and **(b.3)** represents a strip with trimming.

- (L, β) (resp. (α, W)) represents a *uniform* horizontal (resp. vertical) strip if it is composed of different pieces having the same width (resp. length).
- (L, β) (resp. (α, W)) represents a *general* horizontal (resp. vertical) strip if it contains at least two pieces with different widths (resp. lengths) which participate to the composition of the strip.

Example 1. Consider the instance problem of figure 2(a). It is composed of a stock rectangle of dimensions $(L, W) = (11, 7)$ and three pieces p_1, p_2 and p_3 respectively of dimensions (6, 3), (4, 3) and (1, 2) respectively.

Figure 2(b.1) shows a *uniform horizontal strip* with dimensions (11, 2). The considered strip is composed of the same piece with width $\beta = 2$ and with value equal to 22. We say that the obtained strip is *without trimming*, i.e., vertical cuts are sufficient to produce all participated pieces (for the considered strip).

Also, figure 2(b.2) shows another *uniform horizontal strip* with dimensions (11, 3) and with value equal to 30. This strip contains two pieces of type p_1 and p_2 respectively, having the same width $\beta = 3$. It is also obtained without trimming.

Now, consider figure 2(b.3). The obtained strip (of dimensions $(L, \beta) = (11, 3)$ and with value 32) represents a *general horizontal strip*, because some pieces have different widths (in this example, the width of p_1 is equal to 3 and the width of p_3 is equal to 2). We say that the strip is obtained *with trimming*, i.e., vertical cuts are not sufficient to produce all participated pieces and so, supplementary horizontal cuts are necessary to extract some pieces (in figure 2(b.3), the piece p_3 is obtained by applying a supplementary horizontal cut).

According to Definition 1 and the strips illustrated by figures 2(b), we can distinguish two cases of the staged cutting problem. The first one is considered when trimming is permitted (called the *nonexact case* in Gilmore and Gomory (1965), Hifi (2001) and Morabito and Garcia (1998)) and the second one represents the *exact case* in which trimming is not permitted.

In this paper we treat both FCU_2TDC and FCW_2TDC problems by considering the *exact* (without trimming) and the *nonexact* (trimming is permitted) cases.

The paper is organized as follows. First (Section 3), we describe an extended version of Gilmore and Gomory's procedure (1965) applied especially for providing an approximate solution to both FCU_2TDC and FCW_2TDC problems. Both exact and non-exact versions of the problem are considered. In Section 3.3, we propose an improved version of the approximate algorithm. Third (Section 4), we present an approximate algorithm for the non-exact case, i.e., trimming is permitted. In Section 5, we develop an exact algorithm for solving unweighted and weighted FC_2TDC problems and by considering the non-trimming case. Section 6 presents an exact algorithm when trimming is permitted. The obtained algorithm is an adaptation of the algorithm proposed in Section 5. Finally, the performance of the proposed algorithms are presented in Section 7. A set of large size problem instances is considered and benchmark results are given. To our knowledge, no other exact solution procedure for the FC_2TDC problem exists in the literature.

3. An approximate algorithm for FC_2TDC: *Without trimming*

In this section, we describe the procedure applied for solving both FCU_2TDC and FCW_2TDC problems. The procedure was originally applied by Gilmore and Gomory (1965) for the fixed unconstrained 2TDC problem and improved later in Hifi (2001). Here we apply an extended version of the procedure by imposing the demand values on each considered piece. Moreover, our procedure can be applied for the two cases of the problem: (i) without trimming and (ii) with trimming.

The procedure is composed of two phases. The *first phase* is composed of two stages: the first stage is applied in order to construct a set of different horizontal strips and, the second one is applied for combining some horizontal strips for producing an *approximate horizontal cutting pattern*. The *second phase* of the procedure is also composed of two stages: first, we try to construct a set of vertical strips and we combine some of them for obtaining an *approximate vertical cutting pattern*.

3.1. The first phase: A horizontal cutting pattern

The first stage. The problem of generating some horizontal bounded strips can be summarized as follows: without loss of generality, assume that the pieces are ordered in nondecreasing order such that $w_1 \leq w_2 \leq \cdots \leq w_n$. Suppose that r denotes the number of different widths of the considered pieces (i.e. we have the following order $\bar{w}_1 < \bar{w}_2 < \cdots < \bar{w}_r$, where $\forall j \in \{1, \ldots, r\}$, $\bar{w}_j \in \{w_1, \ldots, w_n\}$), and $(L, \bar{w}_j)$ is a strip with width $\bar{w}_j$. We define the single *Bounded Knapsack* problems $(BK^{hor}_{L,\bar{w}_j})$, for $j = 1, \ldots, r$, as follows:

$$\left(BK^{hor}_{L,\bar{w}_j}\right)\begin{cases} f^{hor}_{\bar{w}_j}(L) = \max \displaystyle\sum_{i \in S_{L,\bar{w}_j}} c_i x_i \\ \text{subject to} \quad \displaystyle\sum_{i \in S_{L,\bar{w}_j}} l_i x_i \leq L \\ \qquad x_i \leq b_i, \; x_i \in I\!N, \; i \in S_{L,\bar{w}_j} \end{cases}$$

where $S_{L,\bar{w}_j}$ is the set of pieces entering in the strip $(L, \bar{w}_j)$ such that $\forall k \in S_{L,\bar{w}_j}$, $w_k = \bar{w}_j$ (i.e., all pieces have the same width), x_i denotes the number of times the piece i appears in the j-th horizontal strip without exceeding the value b_i, c_i is the weight associated to the piece $i \in S_{L,\bar{w}_j}$ and $f^{hor}_{\bar{w}_j}(L)$ is the solution value of the j-th strip, $j = 1, \ldots, r$.

Remark 1. It consists in solving a series of small independent single bounded knapsack problems $BK^{hor}_{L,\bar{w}_1}$, $BK^{hor}_{L,\bar{w}_2}, \ldots, BK^{hor}_{L,\bar{w}_r}$, respectively. These problems are independent and their resolution is equivalent to solve only a largest single bounded knapsack with n pieces.

The second stage. The aim of this step is to select the best of these horizontal strips for obtaining an *approximate cutting pattern* to the FC_2TDC problem. We do it by solving another single *Bounded Knapsack Problem*, given as follows:

$$\left(BKP^{hor}_{(L,W)}\right) \begin{cases} g^{hor}_L(W) = \max \sum_{j=1}^{r} f^{hor}_{\bar{w}_j}(L) y_j \\ \text{subject to} \quad \sum_{j=1}^{r} \bar{w}_j y_j \le W \\ \qquad y_j \le a_j, \; y_j \in I\!N \end{cases}$$

where y_j, $j = 1, \ldots, r$, is the number of occurrences of the j-th horizontal strip in the stock rectangle S, bounded by its upper bound-value a_j, $f^{hor}_{\bar{w}_j}(L)$ is the solution value of the j-th strip and $g^{hor}_L(W)$ is the solution value of the cutting pattern, for the rectangle (L, W), composed of some horizontal strips. It is clear that each horizontal strip j, $j = 1, \ldots, r$, can be appeared at least one time and at most $\lfloor \frac{W}{\bar{w}_j} \rfloor$ times in the stock rectangle.

Let δ_{ij} be the number of times the i-th piece appears in the j-th strip. So, we can easily remark that a possible way for computing each a_j is the following:

$$a_j = \min\left\{ \left\lfloor \frac{W}{\bar{w}_j} \right\rfloor, \min_{1 \le i \le n} \left\{ \left\lfloor \frac{b_i}{\delta_{ij}} \right\rfloor \middle| \delta_{ij} > 0 \right\} \right\}, \quad j = 1, \ldots, r. \tag{2}$$

3.2. *The second phase: A vertical cutting pattern*

The first stage. By applying the same process as in Section 3.1 (first stage), we can generate the set of the vertical strips such that:

- we consider that pieces are ordered in nondecreasing order such that $l_1 \le l_2 \le \cdots \le l_n$;
- we reverse the role of widths and lengths in the previous single bounded knapsack problems $BK^{hor}_{L,\bar{w}_j}$, i.e., $\bar{w}_j$ by $\bar{l}_j$, l_i by w_i, L by W, $f^{hor}_{\bar{w}_j}(L)$ by $f^{ver}_{\bar{l}_j}(W)$ and r by s (where s denotes the number of distinct lengths such that $\bar{l}_1 < \bar{l}_2 < \cdots < \bar{l}_s$).

We denote the resulting single bounded knapsack problems by $BK^{ver}_{\bar{l}_j,W}$, for $j = 1, \ldots, s$. For more clarity, we present below each problem corresponding to the j-th length, where

$j = 1, \ldots, s.$

$$\left(BK^{ver}_{\bar{l}_j,W}\right)\begin{cases} f^{ver}_{\bar{l}_j}(W) = \max \sum_{i \in S_{\bar{l}_j,W}} c_i x_i \\ \text{subject to} \quad \sum_{i \in S_{\bar{l}_j,W}} w_i x_i \leq W \\ x_i \leq b_i,\ x_i \in I\!N,\ i \in S_{\bar{l}_j,W} \end{cases}$$

where $S_{\bar{l}_j,W}$ is the set of pieces entering in the vertical strip $(\bar{l}_j, W)$ such that $\forall k \in S_{\bar{l}_j,W},\ l_k = \bar{l}_j$ (i.e., all pieces have the same length), x_i denotes the number of times the piece i appears in the j-th vertical strip without exceeding the value b_i, c_i is the weight associated to the piece $i \in S_{\bar{l}_j,W}$ and $f^{ver}_{\bar{l}_j}(W)$ is the solution value of the j-th vertical strip, $j = 1, \ldots, s$. So, by solving these independent problems, we obtain the set of vertical strips.

The second stage. In this section, we also apply the same process as in Section 3.1 (second stage). Indeed, we construct an *approximate vertical cutting pattern* by solving the following single bounded knapsack:

$$\left(BKP^{ver}_{(L,W)}\right)\begin{cases} g^{ver}_{W}(L) = \max \sum_{j=1}^{s} f^{ver}_{\bar{l}_j}(W) y_j \\ \text{subject to} \quad \sum_{j=1}^{s} \bar{l}_j y_j \leq L \\ y_j \leq a_i,\ y_j \in I\!N \end{cases}$$

where $(BKP^{ver}_{(L,W)})$ is obtained by replacing in the above problem $(BKP^{hor}_{(L,W)})$, $\bar{w}_j$ by $\bar{l}_j$, W by L, $g^{hor}_{L}(W)$ by $g^{ver}_{W}(L)$, r by s (where s is the number of distinct lengths) and

$$a_j = \min\left\{\left\lfloor \frac{L}{\bar{l}_j} \right\rfloor, \min_{1 \leq i \leq n}\left\{\left\lfloor \frac{b_i}{\delta_{ij}} \right\rfloor \middle| \delta_{ij} > 0\right\}\right\}, \quad j = 1, \ldots, s, \tag{3}$$

where δ_{ij} is the number of times that piece i appears in the j-th vertical strip.

Remark 2. If we consider that the first-stage cut is unspecified (i.e., the first cut is made horizontally or vertically), then the approximate solution to the FC_2TDC problem is the one that realizes the best solution value between horizontal and vertical feasible two-staged patterns, i.e., the pattern realizing the value $\max\{g^{hor}_{L}(W), g^{ver}_{W}(L)\}$.

3.3. Enhancing the approximate algorithm

In this part, we present a simple improvement to the previous approximate algorithm. The problems $(BK^{hor}_{L,\bar{w}_j})$ are used for producing a set of *r optimal horizontal strips* and, combined later by applying one problem $(BKP^{hor}_{(L,W)})$. The same process is applied for creating the s *optimal vertical strips* and, combined later by solving the problem $(BKP^{ver}_{(L,W)})$.

The improvement consists in generating other horizontal (resp. vertical) substrips in order to combine them by applying the same problem $(BKP^{hor}_{(L,W)})$ (resp. $(BKP^{ver}_{(L,W)})$). All horizontal substrips are constructed by using the following procedure:

1. Let r be the number of the *optimal horizontal strips*, obtained by solving the largest problem $(BK^{hor}_{L,\bar{w}_r})$, where $\bar{w}_r \in \{w_1, \dots, w_n\}$.
2. let $a_j,\ j = 1, \dots, r$, be the maximum number that the j-th horizontal strip can be appeared in the stock rectangle (L, W) (see Eq. (2)).
   For each strip $j,\ j = 1, \dots, r$, compute b'_i as follows:

   $$b'_i = b_i - a_j * \delta_{ij}$$

   where δ_{ij} denotes the number of times that piece i appears in the j-th strip.
3. If there exists a component b'_i, such that $b'_i > 0$, then solve the problem $(BK^{hor}_{L,\bar{w}_{r'}})$, where r' is the number of the new substrips and $\bar{w}_{r'}$ is equal to $\max_{1\le i\le n}\{w_i \mid b'_i > 0\}$.
4. Let $r = r + r'$ be the number of the available (sub)strips. Then, solve $(BKP^{hor}_{(L,W)})$ by considering the new number r of (sub)strips and by setting $a_j,\ j = r + 1, \dots, r + r'$, equal to

   $$\min\left\{\left\lfloor \frac{W}{\bar{w}_j} \right\rfloor, \min_{1\le i\le n}\left\{\left\lfloor \frac{b'_i}{\delta_{ij}} \right\rfloor \,\middle|\, \delta_{ij} > 0\right\}\right\}.$$

Of course, we apply the same procedure for the vertical substrips, by replacing r by $s,\ r'$ by s' and, by solving $(BK^{ver}_{\bar{l}_{s'},W})$ and $(BKP^{ver}_{(L,W)})$. Limited computational results showed that such an improvement, i.e., applying the above procedure one time, produces a satisfactory solution to the problem.

4. An approximate algorithm for FC_2TDC: *Trimming is permitted*

In this section, we present an approximate algorithm for the non-exact case (trimming is permitted). Clearly, the approximate algorithm used for the exact case of the problem (without trimming) is also an approximate algorithm for the non-exact case. So, we apply the improved version of the algorithm (Section 3.3) for computing the first approximate solution to the non-exact case of the problem.

In Hifi (1997), another approximate algorithm has been developed for starting the exact approach used for solving the non-staged constrained cutting problem. This approximate algorithm is composed of two steps:

1. constructing a *set of general horizontal (resp. vertical) strips* (see Definition 1);
2. combining some of these general horizontal (resp. vertical) strips for obtaining an *approximate solution to the non-staged constrained TDC problem.* The solution is obtained through a set packing problem.

This solution can also be considered as an approximate solution to the FC_2TDC problem when trimming is permitted. In what follows, we try to summarize the approach and for more details, the reader can be referred to Hifi (1997).

The first stage. The first stage represents the first phase of Section 3.1 (resp. Section 3.2) using a slight modification. In order to construct the set of horizontal (resp. vertical) strips, we consider the same problem $BK^{hor}_{L,\bar{w}_j}$ (resp. $BK^{ver}_{\bar{l}_j,W}$) and we reorder all pieces of each set $S_{L,\bar{w}_j},\ j = 1,\ldots,r$ (resp. $S_{\bar{l}_j,W},\ j = 1,\ldots,s$) which represents the set of rectangular pieces entering in the strip $(L, \bar{w}_j)$ (resp. $(\bar{l}_j, W)$). For each index j of $S_{L,\bar{w}_j}$ (resp. $S_{\bar{l}_j,W}$), we have $(l_i, w_i) \le (L, \bar{w}_j)$ (resp $(l_i, w_i) \le (\bar{l}_j, W)$).

In this case, by applying a dynamic programming procedure, we only solve the largest problem $BK^{hor}_{L,\bar{w}_r}$ (resp. $BK^{ver}_{\bar{l}_s,W}$), where $\bar{w}_r = \max_{1\le i\le n}\{w_i\}$ (resp. $\bar{l}_s = \max_{1\le i\le n}\{l_i\}$).

The second stage. The second stage consists in selecting the best of these *general optimal horizontal/vertical strips* for constructing an *approximate horizontal/vertical cutting pattern*. This can be realized, by solving the following set (strip) packing problems $(P^{hor}_{(L,W)})$ and $(P^{ver}_{(L,W)})$, respectively:

$$\left(P^{hor}_{(L,W)}\right)\begin{cases} g^{hor}_L(W) = \max \displaystyle\sum_{j=1}^{r} f^{hor}_{\bar{w}_j}(L)y_j \\ \text{subject to} \quad \displaystyle\sum_{j=1}^{r} \bar{w}_j y_j \le W \\ \qquad\qquad\quad \displaystyle\sum_{j=1}^{r} \delta_{ij} y_j \le b_i \\ \qquad\qquad\quad i = 1,\ldots,n,\ y_j \in \mathbb{N} \end{cases}$$

$$\left(P^{ver}_{(L,W)}\right)\begin{cases} g^{ver}_W(L) = \max \displaystyle\sum_{j=1}^{s} f^{ver}_{\bar{l}_j}(W)z_j \\ \text{subject to} \quad \displaystyle\sum_{j=1}^{s} \bar{l}_j z_j \le L \\ \qquad\qquad\quad \displaystyle\sum_{j=1}^{s} \delta_{ij} z_j \le b_i \\ \qquad\qquad\quad i = 1,\ldots,n,\ z_j \in \mathbb{N} \end{cases}$$

where $y_j,\ j = 1,\ldots,r$ (resp. $z_j,\ j = 1,\ldots,s$) is the number of occurrences of the j-th strip in the stock rectangle S, δ_{ij} is the number of occurrences of the i-th piece in the j-th strip and $g^{hor}_L(W)$ (resp. $g^{ver}_W(L)$) is the solution value of the horizontal (resp. vertical) cutting pattern.

Consequently, when we deal with the initial rectangle (L, W), we need to solve two set packing problems. Of course, the set packing problem is NP-complete and the method used to approximately solve this problem will significantly influence the algorithm performance. In our computational results, on one hand, we have used a simple greedy algorithm (for more details the reader is referred to Hifi, 1997) and the approximate solution of Section 3.3, and we take the better solution as the solution to the non-exact case.

5. An exact algorithm for the FC_2TDC: *Without trimming*

Branch-and-Bound (B&B) is a well-known technique for solving combinatorial search problems. Its basic scheme is to reduce the problem search space by dynamically pruning unsearched areas which cannot yield better results than already found. The B&B method searches a finite space T, implicitly given as a set, in order to find one state $t^* \in T$ which is optimal for a given objective function f. Generally, this approach proceeds by developing a tree in which each node represents a part of the state space T. The root node represents the entire state space T. Nodes are branched into new nodes which means that a given part T' of the state space is further split into a number of subsets, the union of which is equal to T'. Hence, the optimal solution over T' is equal to the optimal solution over one of the subsets and the value of the optimal solution over T' is the minimum (or maximum) of the optima over the subsets. The decomposition process is repeated until the optimal solution over the part of the state space is reached.

The proposed approach uses a simple strategy for solving exactly the FC_2TDC problem. This strategy can be summarized as follows:

1. Starting by an initial (feasible) lower bound;
2. Constructing a set of horizontal/vertical (sub)strips;
3. Combining some of these (sub)strips for producing an optimal solution to the FC_2TDC problem.

5.1. An exact algorithm for the horizontal FC_2TDC problem: Without trimming

5.1.1. Uniform horizontal substrips and the optimal horizontal solution on (L, W). The construction of each horizontal (sub)strip is obtained by combining all pieces and their copies by horizontal builds. In what follows, we use the Wang's construction (see Wang, 1983 and initially used by Albano and Sapuppo, 1980) for obtaining the different horizontal (sub)strips and (sub)optimal solutions. Of course, the definition is also applicable when we consider the construction of the vertical strips.

Definition 2. Let (l_A, w) and (l_B, w) be the dimensions of two substrips A and B (see figure 3(a)). We call $R = (l_A + l_B, w)$, where $l_A + l_B \leq L$, an internal horizontal rectangle (substrip) obtained by joining the two patterns using a horizontal build (see figure 3(b)). Moreover, the number of copies of each piece contained on the resulting internal horizontal rectangle does not exceed the demand values b_i, $i = 1, \ldots, n$.

Also, the optimal horizontal solution (which is also an internal horizontal rectangle) to the FC_2TDC problem is obtained by combining some horizontal substrips and/or some internal horizontal rectangles having the strip structure. Each of these rectangles is obtained by applying a vertical build according to the following definition.

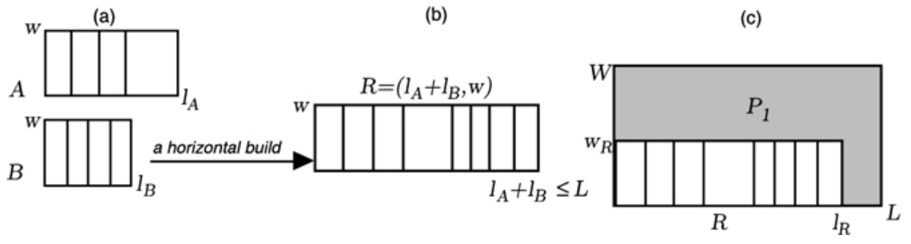

Figure 3. Representation of (a) two internal horizontal rectangles, (b) a resulting substrip using a horizontal build and (c) the internal horizontal rectangle placed in S with the remaining area P_1.

Definition 3. Let (l_A, w_A) and (l_B, w_B) be the dimensions of two internal horizontal rectangles A and B respectively (see figure 4(a)). We call $R = (\max\{l_A, l_B\}, w_A + w_B)$, where $w_A + w_B \leq W$, an *internal vertical rectangle* obtained by joining the two internal rectangles using a vertical build (see figure 4(b)). Moreover, the number of copies of each piece contained on the resulting internal vertical rectangle does not exceed the demand values $b_i,\ i = 1, \ldots, n$

Here, each *internal horizontal rectangle*: (i) is composed by a set of pieces having the same width according to Definition 2 and (ii) is composed by a set of horizontal (sub)strips according to Definition 3.

So, given a stock rectangle (L, W), a set of pieces and an internal (horizontal) rectangle R, the objective is to seek an *upper bound* for the value of the best cutting pattern on (L, W), that is constrained to include R. Let $g(R)$ be the *internal value* of R that is the sum of the weights of the pieces contained in R. Let $h(R)$ be the *maximum value* obtainable, without violating the demand constraints, from the region P_1 of figure 3(c) (resp. the region P_2 of figure 4(b)). Then,

$$f(R) = g(R) + h(R)$$

represents the *maximum value of a feasible cutting pattern*, that is constrained to include R. We are going now to show how we can compute the estimation of $h(R)$ without much

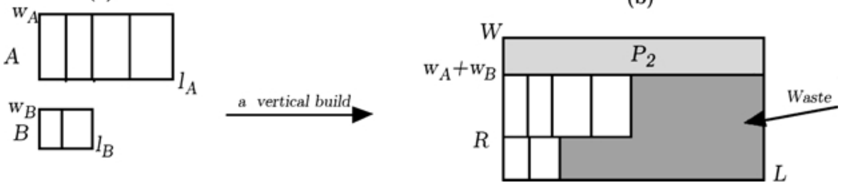

Figure 4. Representation of (a) two (sub)strips and (b) a resulting internal rectangle using a vertical build, placed in the initial stock rectangle.

computational effort. Generally, it is not evident to compute directly the value of $h(R)$ and so, we try to use an estimation to $h(R)$, say $h'(R)$, where $h'(R) \geq h(R)$. In what follows, we develop some results representing some valid estimations to $h(R)$ (and $f(R)$).

5.1.2. An upper bound for a subrectangle (α, β). In this part, we develop some results for computing the upper bound $f'(R)$. We distinguish two cases:

1. The estimation of $f(R)$ when a horizontal build is applied. In this case, we principally consider the area P_1 of figure 3(c).
2. If the build is vertical according to Definition 3, then the value $f'(R)$ is estimated by using another formula (estimation of the region P_2 of figure 4(b)).

So, consider a subrectangle (α, β) $(\leq (L, W))$ and a (sub)set of the initial pieces. It is easy to see that the solution value of the *horizontal* FC_2TDC problem (on (α, β)) represents an upper bound to the horizontal FC_2TDC problem (on α, β)). This upper bound can be computed by solving the two following single *Unbounded Knapsack* problems (for more details, the reader can be referred to Gilmore and Gomory (1965) and Hifi (2001):

$$\left(UK^{hor}_{\alpha,w_j}\right) \begin{cases} F^{hor}_{\bar{w}_j}(\alpha) = \max \sum_{i \in S_{\alpha,\bar{w}_j}} c_i x_i \\ \text{subject to} \quad \sum_{i \in S_{\alpha,\bar{w}_j}} l_i x_i \leq \alpha \\ x_i \in \mathbb{N}, \ i \in S_{\alpha,\bar{w}_j} \end{cases}$$

$$\left(UKP^{hor}_{(\alpha,\beta)}\right) \begin{cases} G^{hor}_{\alpha}(\beta) = \max \sum_{j=1}^{p} F^{hor}_{\bar{w}_j}(\alpha) y_j \\ \text{subject to} \quad \sum_{j=1}^{p} \bar{w}_j y_j \leq \beta \\ y_j \in \mathbb{N} \end{cases}$$

where $S_{\alpha,\bar{w}_j}$ is the set of rectangular pieces entering in the strip $(\alpha, \bar{w}_j)$ (i.e., $\forall i \in S_{\alpha,\bar{w}_j}$, we have $w_i = \bar{w}_j$), x_i denotes the number of times the piece i appears in the j-th horizontal strip, c_i is the weight associated to the piece $i \in S_{\alpha,\bar{w}_j}$ and $F^{hor}_{\bar{w}_j}(\alpha)$ is the solution value of the j-th horizontal strip, $j = 1, \dots, p$ (p is the number of the different strips, with $p \leq r$). Also, y_j, $j = 1, \dots, p$, is the number of occurrences of the j-th strip in the subrectangle (α, β), $F^{hor}_{\bar{w}_j}(\alpha)$ is the solution value of the j-th horizontal substrip and $G^{hor}_{\alpha}(\beta)$ is the solution value of the *fixed unconstrained 2TDC problem* which represents an upper bound to the FC_2TDC problem.

Of course, at the root node of the developed tree, we just replace α by L and β by W for obtaining the first upper bound of the problem.

5.1.3. Upper bounds at internal nodes. The purpose of this section is to obtain an upper bound on the optimal solution of the FC_2TDC problem from single unbounded and bounded knapsack problems. The upper bound concerns the estimation of $h'(R)$, when R is considered as an internal rectangle.

The horizontal construction that we have considered for producing the set of horizontal builds can be viewed as a two stage method:

1. a horizontal build provides an internal horizontal rectangle R,
2. an estimation of $h'(R)$ attempts to minimize the upper bound $f'(R)$, when R is considered as a (feasible) solution to the FC_2TDC problem. In this case, we can reject some (sub)strips if their upper bounds is not greater than the best current feasible solution value.

The first point is realized by applying Definition 2 and the second point, which corresponds to the value of $f'(R)$, is represented by the following results.

Lemma 1. *Let $\{(\alpha, \beta_1), (\alpha, \beta_2), \ldots, (\alpha, \beta_m)\}$ be a set of horizontal substrips having the same length α. Suppose that all widths β_i, $i = 1, \ldots, m$, are given in nondecreasing order and each substrip (α, β_i) is characterized by its solution value $F^{hor}_{\beta_i}(\alpha)$. Then, only one single knapsack problem permits to produce the upper bounds to all considered subrectangles (α,γ), where $0 \le \gamma \le \beta_m$.*

Proof: Consider the fictive subrectangle (α, β), where

$$\beta = \max_{1 \le i \le m} \{\beta_i\}.$$

By applying a dynamic programming procedure to the resulting problem $(UKP^{hor}_{\alpha,\beta})$, we obtain:

1. an upper bound for the fictive rectangle (α, β), with value $G^{hor}_{\alpha}(\beta)$,
2. an upper bound to each subrectangle (α, y), with value $G^{hor}_{\alpha}(y)$ such that $y \le \beta$.

We recall that this is possible because dynamic programming techniques are considered. Hence, if x $(= \beta_i)$ coincides with one of the widths of the set $\{\beta_1, \beta_2, \ldots, \beta_m\}$, then necessary we have the solution value $G^{hor}_{\alpha}(\beta_i)$ which corresponds to an upper bound to the subrectangle (α, β_i). □

In particular, if we consider that $\beta_1 = w_1, \beta_2 = w_2, \ldots, \beta_m = w_m$, where $m \le r$, and by applying a dynamic programming procedure to the problem $UKP^{hor}_{L,W}$, then we have necessary the solution value $G^{hor}_{L}(y)$ of each subrectangle (L, y), with $0 \le y \le W$.

Lemma 2. *Let (L,β) be a horizontal substrip with length L and width β, where $\beta \le W$. Then, the resolution of the single unconstrained knapsack problem $(UK^{hor}_{L,\bar{w}_j})$ produces an upper bound for each substrip (x, β) with values $F^{hor}_{\beta}(x)$, where $x \le L$.*

Proof: We use the same reasoning as for Lemma 1.

By solving initially the problem $UK^{hor}_{L,\bar{w}_j}$ and by applying a dynamic programming procedure, we obtain the following results:

1. we construct the strip (L, β) with value $F_\beta^{hor}(L)$. This solution value represents an upper bound to the strip (L, β), when we consider the FC_2TDC problem (see Section 5.1.2),
2. all substrips with dimensions (x, β), where $x \leq L - 1$, are available with their solution values $F_\beta^{hor}(x)$. This solution value is also an upper bound to each subrectangle (x, β), where $x \leq L$. □

In what follows, we present upper bounds representing the area P_1 of figure 3(c). These bounds are principally based upon single knapsack problems and the time complexity for the computation of these bounds is apparently constant.

Theorem 1. *Given an internal rectangle R with dimensions (l_R, w_R), a valid upper bound for $f(R)$, when a horizontal build is applied, is*

$$f_1(R) = g(R) + G_L^{hor}(W - w_R) + F_{w_k}^{hor}(L - l_R) \tag{4}$$

where $G_L^{hor}(W - w_R)$ is the solution value of the single unconstrained knapsack $UKP_{(L,W-w_R)}^{hor}$ and $F_k^{hor}(L - l_R)$ is the solution value of the single unconstrained knapsack $UK_{L-l_R,W-\bar{w}_k}^{hor}$, where $k \in \{1, \dots, r\}$.

Proof: Recall that $f'(R) = g(R) + h'(R)$ is an upper bound to the solution containing the internal rectangle R as one of its component. The time complexity for computing $g(R)$ is constant, since we just use an addition.

We can remark that the optimal solution, to the constrained two-staged problem, containing the internal rectangle R is necessarily composed of a strip (L, w_R) and a complementary solution which can be represented by the second part $(L, W - w_R)$ of the stock rectangle. So, by applying Lemmas 1 and 2, we can affirm that:

1. the available value $G_L^{hor}(W - w_R)$ (of the problem $UKP_{(L,W-w_R)}^{hor}$) represents an upper bound to the subrectangle $(L, W - w_R)$.
2. the available value $F_{w_R}^{hor}(L - l_R)$ (of the problem $UK_{L-l_R,W-\bar{w}_k}^{hor}$) represents an upper bound to the subrectangle (L, w_R), since w_R coincides with one of the following widths $\bar{w}_1, \bar{w}_2, \dots, \bar{w}_r$.

Hence, the solution value

$$f_1(R) = g(R) + G_L^{hor}(W - w_R) + F_{\bar{w}_k}^{hor}(L - l_R) \geq f(R)$$

represents an upper bound to the FC_2TDC problem, containing the internal rectangle R. Clearly, the time complexity for computing $f_1(R)$ is constant, because:

1. $g(R)$ is computed in constant time,
2. $h_1(R) = G_L^{hor}(W - w_R) + F_{\bar{w}_k}^{hor}(L - l_R)$ is also computed in constant time. In this case, we just remark that the two upper bounds corresponding to the subrectangles $(L, W - w_R)$ and $(L - l_R, w_R)$ are available. □

The above result shows that the upper bound to the FC_2TDC problem can be obtained by considering some remaining solution values derived from single unbounded knapsack problems. Let us now see how we can introduce the single bounded knapsack problem for giving another and better upper bound. This bound dominate the previous one (Eq. (4)) and its computation time complexity is also constant.

Corollary 1. *A valid and a better upper bound for $f(R)$ is*

$$f_2(R) = g(R) + G_L^{hor}(W - w_R) + f_{\bar{w}_k}^{hor}(L - l_R) \tag{5}$$

where $f_{\bar{w}_k}^{hor}(L - l_R)$ is the solution value of the single bounded knapsack $BK_{L-l_R, W-\bar{w}_k}^{hor}$ and $k \in \{1, \ldots, r\}$.

Proof: Let $BK_{(L,w_R)}^{hor}$ be the single bounded knapsack for which the width w_R coincides with a width w_j, where $w_j \in \{\bar{w}_1, \ldots, \bar{w}_r\}$.

By applying Lemma 2 to the problem $BK_{(L,w_R)}^{hor}$, we can deduce that all optimal substrips (x, w_R), where $x \leq L$, are available and so, in particular the optimal substrip $(L - l_R, w_R)$ is also obtained. So, the optimal solution value $f_{\bar{w}_k}^{hor}(L - l_R)$ represents also an upper bound for the subrectangle (substrip) $(L - l_R, w_R)$ and so,

$$f_2(R) = g(R) \;+\; G_L^{hor}(W - w_R) \;+\; f_{\bar{w}_k}^{hor}(L - l_R)$$

is an upper bound for the solution containing the internal rectangle R.

Since the solution value $f_{\bar{w}_k}^{hor}(L - l_R)$ denotes the value of the best solution value of the substrip $(L - l_R, w_R)$, then necessary this solution value is better than the solution value $F_{\bar{w}_k}^{hor}(L - l_R)$. This is true because the solution value of the problem $UK_{L-l_R, W-\bar{w}_k}^{hor}$ is obtained by fixing the demand values b_i to $\lfloor \frac{L}{l_i} \rfloor \lfloor \frac{W}{w_i} \rfloor$, for $i = 1, \ldots, n$. Hence,

$$f_{\bar{w}_k}^{hor}(L - l_R) \leq F_{\bar{w}_k}^{hor}(L - l_R) \Leftrightarrow f_2(R) \leq f_1(R) \qquad \square$$

The following result is used when the vertical build is applied for combining two horizontal internal rectangles, i.e. two (sub)strips. In this case, the value of $h'(R)$ concerns the estimation of the area P_2 of figure 4(b).

Theorem 2. *Let A and B be two internal rectangles with dimensions (l_A, w_A) and (l_B, w_B) respectively. A valid upper bound for $f(R)$, where R is the resulting internal vertical rectangle, is*

$$f_3(R) = g(R) + G_L^{hor}(W - w_R) \tag{6}$$

where $G_L^{hor}(W - w_R)$ is the solution value of $UKP_{(L,W-w_R)}^{hor}$.

Proof: Let R be the internal vertical rectangle obtained by combining vertically A and B respectively (see figure 4).

According to Definition 3, the dimensions of R are at least equal to $(\max\{l_A, l_B\}, w_A + w_B)$ and since R is considered as an internal vertical rectangle, then R will never combined horizontally with another internal rectangle. Hence,

$$(l_R, w_R) = (L, w_A + w_B) \tag{7}$$

and

$$\begin{aligned} f_3(R) &= g(R) + g(B) + h'(R) = g(R) + G_L^{hor}(W - (w_A + w_B)) \\ &= g(R) + G_L^{hor}(W - w_R) \end{aligned} \tag{8}$$

since R is an internal vertical rectangle and it is considered as a strip with dimensions $(L, w_A + w_B)$. □

ALGO. The Exact (Horizontal) Algorithm for the FC_2TDC problem.

```
Input: an instance of the FC_2TDC problem: a set of rectangles $(R_1, \ldots, R_n)$ and a
stock rectangle $(L, W)$.
Output: an optimal (horizontal) solution Best with value $Opt^{hor}(Best)$..
• Solve the following single (un)bounded knapsack problems :
    • $UK_{L,w_j},\ j = 1, \ldots, r$ (by using only the largest single knapsack) and $UK^{hor}_{(L,W)}$;
    • $BK_{L,w_j},\ j = 1, \ldots, r$ (by using only the largest single knapsack) and $BK^{hor}_{(L,W)}$.
• Set $Opt^{hor}(Best) = g_L^{hor}(W)$;
• If $Opt^{hor}(Best) = G_L^{hor}(W)$ then exit with the optimal solution Best with value
  $Opt^{hor}(Best)$;
• Set $Open = \{R_1, R_2, \ldots, R_n\}$;
• Set Closed equal to empty set and set Stop = false;
Repeat
  Take one internal (horizontal) rectangle $R$ from Open having the highest value $f'$;
  If $f'(R) - Opt(Best) \le 0$, then Stop:= true
  Else
  Begin
        1. transfer $R$ from Open to Closed;
        2. construct all internal rectangles $Q$ such that:
            (a) each element $q$ of $Q$ is constructed by applying a horizontal build
                (see Definition 2) or a vertical build (see Definition 3) of $R$ with some
                internal horizontal rectangles, say $R'$, of Closed;
            (b) the width $w_{R'}$ of each $R'$ of Closed is equal to $w_R$, when the
                horizontal build is applied
            (c) the dimensions $(l_q, w_q)$ are less than or equal to $(L, W)$;
            (d) $\forall q \in Q,\ d_i^q \le b_i,\ i = 1, \ldots, n$;
        3. for each element $q$ of $Q$, do:
            (a) compute $g(q)$;
            (b) compute $h'(q)$ (using equation (5) for the horizontal build and
          equations (6) or (9) when the vertical build is considered)
          If $g(q) > Opt^{hor}(Best)$ then set $Best = q$ and $Opt^{hor}(Best) = g(q)$;
          If $g(q) + h'(q) > Opt^{hor}(Best)$ then set $Open = Open \cup \{q\}$;
        4. If $Open = \{\}$ then $Stop := true$;
  end;
Until $Stop$;
• Exit with the optimal solution $Best$ with value $Opt^{hor}(Best)$.
```

Corollary 2. *Let R be a resulting internal rectangle constructed by combining two internal rectangles A and B. If $W - w_R = w_{min}$, where w_{min} denotes the smallest width entering in the strip $(L, W - w_R)$, then the valid upper bound for $f(R)$ is*

$$f_4(R) = g(R) + g_L^{hor}(W - w_R) \leq f_3(R) \tag{9}$$

where $g_L^{hor}(W - w_R)$ is the solution value of $BKP^{hor}_{(L,W-w_R)}$.

Proof: On one hand, from Theorem 2, we have $f_3(R) = g(R) + G_L^{hor}(W - w_R)$ and since $W - w_r = w_{min}$, then necessary $g_L^{hor}(W - w_R)$ is a better upper bound for the strip (L, w_{min}).

On the other hand, $g_L^{hor}(W - w_R)$ is less than or equal to $G_L^{hor}(W - w_R)$, since $G_L^{hor}(W - w_R)$ is the better solution value representing the unconstrained version of the problem. Then, we deduce that $f_4(R)$ is a valid upper bound and it is better than $f_3(R)$. □

The algorithmic outline, for solving the *horizontal* FC_2TDC problem, is represented by ALGO. The algorithm works as follows. Initially, the set of bounded and unbounded horizontal strips are constructed by solving BK_{L,w_r} and BK_{L,w_r} respectively. This is possible because *dynamic programming procedures* are used. Also, the problems $BK^{hor}_{L,W}$ and $UK^{hor}_{L,W}$ are solved by applying dynamic programming procedures. They produce an initial upper bound, say $G_L^{hor}(W)$, to the stock rectangle (L, W) and an initial lower bound, say $g_L^{hor}(W)$, to the FC_2TDC problem.

The other steps of the algorithm are explained as follows: the procedure uses two main lists, *Open* and *Closed*. The *Open* list initially contains n elements such that each element $R_i(\in Open)$, $i = 1, \ldots, n$, is composed by the dimensions of the i-th piece $(l_{R_i}, w_{R_i}) = (l_i, w_i)$, the *internal value* $g(R_i) = c_i$, the *estimation value* $h'(R_i)$ and a vector d^{R_i} of dimension n, where $d_j^{R_j} \leq b_j$, $j = 1, \ldots, n$, is the number of times the piece j appears in R_i. The *Closed* list is initialized to empty set. At each iteration, an element R from *Open* is taken and transferred to *Closed*. A set Q of new *internal horizontal rectangles* is created by combining R with some elements of *Closed*, using a horizontal (according to Definition 2) and/or a vertical build (according to Definition 3). We distinguish the following cases:

(a) for a horizontal build: (i) $q \in Q$ if $w_R = w_{R'}$, (ii) $l_q \leq L$, (iii) $d_j^q \leq b_j$, $j = 1, \ldots, n$, and (iv) $g(q) + h'(q) > Opt^{hor}(Best)$, where $Opt^{hor}(Best)$ is the best current solution value.

(b) for a vertical build: (i) $w_q \leq W$, (ii) $d_j^q \leq b_j$, $j = 1, \ldots, n$, and (iii) $g(q) + h'(q) > Opt^{hor}(Best)$, where $Opt^{hor}(Best)$ is the best current solution value.

The algorithm stops when *Open* is reduced to empty set, which means that is not possible to construct a better candidate internal (horizontal) rectangle.

5.1.4. Detecting some duplicates. The aim of this section is to avoid some duplicate (sub)strips (patterns) produced by the construction process according to Definitions 2 and 3. First, we present (See Proposition 1) a simple way for neglecting some constructions. Second, we apply a simple procedure for neglecting other patterns.

Proposition 1. *Let A and B be two internal rectangles, where A and B are composed at least by two pieces. Assume that $A \in$ Open and $B \in$ Closed and suppose that the combination between A and B produces an internal rectangle. Then the horizontal build between A and B represents a duplicate internal rectangle.*

Proof: Consider that A (resp. B) is composed of two pieces A_1 and A_2 (resp. B_1 and B_2) and suppose that A_1, A_2, B_1 and B_2 coincide with some pieces of the instance problem. Of course, we can always decompose an internal rectangle into a series of pieces. We recall that `ALGO` contains two main steps: `Step 1.` an element R is taken from *Open* and transferred to *Closed*; `Step 2.` the same element R is combined with all elements of *Closed*.

Regarding that B is an element of *Closed* and A is an element of *Open*, then B_1 and B_2 are also included in *Closed*. By applying Step 2, the internal rectangle B is transferred from *Open* to *Closed*, A is combined with B_1 (producing an internal rectangle C) and A is combined with B_2 (giving the internal rectangle D). Both elements C and D are directly transferred to *Open*, where C is composed of A_1, A_2 and B_1 and, D contains A_1, A_2 and B_2.

However, consider a certain iteration of `ALGO` for which C is selected from *Open* and transferred to *Closed*. In this case, the following internal rectangles are constructed: C with B_1, C with B_2, etc.

We remark that the combination of C with B_2, say C', can be decomposed into a series of pieces given as follows: A_1, A_2, B_1 and B_2. Also, the combination of C with B_1, say C'', can be decomposed into a series of pieces given as follows: A_1, A_2, B_2 and B_1.

Both resulting internal rectangles C' and C'' are identical to the combination of A with B, with some shifting. Hence, the combination between A and B represents a duplicate pattern □

In `ALGO`, we use Proposition 1 as follows:

Horizontal builds: (i) if A is taken from *Open* and A is composed of one piece, then we combine A with all elements of the *Closed* list; (ii) if A is composed at least by two pieces, then A is combined only with each element B of *Closed* containing one piece.
Vertical builds: (i) if A is taken from *Open* and A is a (sub)strip, then we combine A with all elements of the *Closed* list; (ii) if A is composed at least by two (sub)strips, then A is combined only with each element B of *Closed* containing only a (sub)strip.

In order to accelerate the search process, we use another representation of the *Closed* list (for more details, see Cung et al., 2000). We recall that *Closed* stores the best (sub)patterns already constructed. At each step of ALGO an element R from *Open* is selected and transferred to *Closed*. This element is combined with *some elements*, say $\mathcal{Q}$, of *Closed*.

We consider that all elements of *Closed* are taken on both non-decreasing order of lengths and widths respectively. The interval of lengths (resp. widths) is limited to the dimensions of the initial stock rectangle (L, W). Moreover, all elements of Q can be distinguished as follows: let R be the candidate element taken from *Open* and transferred to *Closed*; let $\mathcal{Q}_{l_R}$ and $\mathcal{Q}_{w_R}$ be the sets of the candidate elements of *Closed* for combinations with R, given as follows:

$$\mathcal{Q}_{l_R} = \{q \in \mathit{Closed} \mid l_q = l_R + l_p \leq L,\ \ p \in \mathit{Closed}\}$$
and
$$\mathcal{Q}_{w_R} = \{q \in \mathit{Closed} \mid w_q = w_R + w_p \leq W,\ \ p \in \text{Closed}\}.$$

A manner to reject some duplicate patterns can be described as follows. Let q be a new generated (feasible) pattern, with dimensions (l_q, w_q), obtained by combining R and another element p of *Closed*. Consider that $d_i^q,\ i = 1, \ldots, n$, is the number of times that piece i appears in q. Then,

– $\mathcal{Q}_{l_R}$ is the subset of the valid candidates (authorized) of the *Closed* list when a *horizontal build* is applied and,
– $\mathcal{Q}_{w_R}$ is the subset of the valid candidates (authorized) of the *Closed* list when a *vertical build* is used.

So, the resulting pattern q represents a duplicate pattern if there exists a pattern q' such that: (i) $(l_q, w_q) \geq (l_{q'}, w_{q'})$ and (ii) $d_i^q \leq d_i^{q'},\ i = 1, \ldots, n$.

5.2. *An exact algorithm for the vertical FC_2TDC problem: Without trimming*

In this part, we give an adaptation of the exact algorithm of Section 5.1 for solving exactly the vertical FC_2TDC problem.

Consider now that the dimensions of each piece i are permuted and coming now equal to (w_i, l_i). In this case, we can easily present an adaptation of the previous algorithm for solving exactly the considered problem. Indeed, the transformation can be applied as follows:

1. We apply ALGO by setting $(l_i, w_i) = (w_i, l_i)$, for $i = 1, \ldots, n$, and by setting $(L, W) = (W, L)$. The set of pieces are initially ordered in nondecreasing order of widths $w_i,\ i = 1, \ldots, n$, which correspond originally to the different lengths $l_i,\ i = 1, \ldots, n$. So,
   (a) The problems $UK_{L,\bar{w}_r}^{hor}$ and $UKP_{(L,W)}^{hor}$ are solved for producing *(internal) upper bounds*.
   (b) The problems $BK_{L,\bar{w}_r}^{hor}$ and $BKP_{(L,W)}^{hor}$ are solved for producing respectively the *set of vertical strips* and the *approximate vertical feasible solution* to FC_2TDC problem. Also, by using a dynamic programming procedure for solving $BKP_{(L,W)}^{hor}$, we create all (*refinement*) *upper bounds* on the (sub)rectangles $(x, W),\ x \leq L$.

(c) By applying the other steps of `ALGO`, we construct the optimal vertical pattern composed by a set of vertical substrips.
(d) Moreover, if the *vertical build* (according to Definition 3), then the results developed in Section 3.1 remain valid.

2. Also, when the *horizontal build* (according to Definition 2) is used, then Theorem 2 and Corollary 2 remain valid and each obtained internal vertical rectangle, say R, generally has an upper bound equal to $g(R) + G_L^{hor}(W - w_R)$, where $G_L^{hor}(W - w_R)$ denotes the solution value of the problem $UKP^{hor}_{(L,W-w_R)}$.
3. By applying the principle of Proposition 1, we can also neglect some constructions representing duplicate builds.

Remark 3. If we consider that the *first-stage cut is unspecified*, then the optimal solution of the FC_2TDC problem is obtained by running together the exact algorithm using the *horizontal* and *vertical* `ALGO`s.

6. Trimming is permitted: An exact algorithm

We present in this section an adaptation of the algorithm described in Section 5 for the trimming case. We use some modifications in `ALGO`. Let A and B two internal patterns with dimensions (l_A, w_A) and (l_B, w_B), respectively.

1. In Definition 1, we use the *general* (*sub*)*strips* and Definition 2 considers that the widths w_A and w_B can be different.
2. At the root node, the initial solution value is computed by applying the approximate algorithm of Section 4.
3. The point (b) of `ALGO` (the main loop `repeat`) is removed.
4. The results developed in Sections 5.1.2–5.1.4 remain valid. Of course, we consider the single knapsack problems developed in Sections 4 and 5.1.2, by considering that: $w_1 \leq w_2, \ldots, \leq w_n$ (resp. $l_1 \leq l_2, \ldots, \leq l_n$), $\bar{w}_1 < \bar{w}_2, \ldots, < \bar{w}_r$ (resp. $\bar{l}_1 < \bar{l}_2, \ldots, < \bar{l}_s$) and $\forall i \in S_{\alpha,\bar{w}_j}$, we have $w_i \leq \bar{w}_j$ (resp. $\forall i \in S_{\bar{l}_j,\beta}$, we have $l_i \leq \bar{l}_j$) for the horizontal (resp. vertical) strips.

With the above modifications, `ALGO` produces an optimal solution when the first stage-cut is specified horizontally. If the first stage-cut is specified vertically, we just apply the principle described in Section 5.2.

Also, when the first stage-cut is unspecified, the algorithm `ALGO` is modified for producing an optimal solution to the FC_2TDC problem. Computational results showed that such a combination (combining horizontal and vertical stage-cuts) makes a good behavior to the resulting algorithm.

7. Computational results

All algorithms were executed on an `UltraSparc10` (250 Mhz and with 128 Mo of RAM) with CPU time limited to two hours. Our computational study was conducted on

Table 1. Test problem details.

The different problem instances							
Weighted				Unweighted			
Inst	L	W	n	Inst	L	W	n
HH	127	98	5	2s	40	70	10
HH	127	98	5	2s	40	70 10	
2	40	70	10	3s	40	70 20	
3	40	70	20	A1s	50	60 20	
A1	50	60	20	A2s	60	60 20	
A2	60	60	20	STS2s	55	85 30	
STS2	55	85	30	STS4s	99	99 20	
STS4	99	99	20	OF1	70	40 10	
CHL1	132	100	30	OF2	70	40 10	
CHL2	62	55	10	W	70	40 20	
CW1	125	105	25	CHL1s	132	100 30	
CW2	145	165	35	CHL2s	62	55 10	
CW3	267	207	40	A3	70	80 20	
Hchl2	130	130	30	A4	90	70 20	
Hchl9	130	130	35	A5	132	100 20	
				CHL5	20	20 10	
				CHL6	130	130 30	
				CHL7	130	130 35	
				CU1	100	125 25	
				CU2	150	175 35	
				Hchl3s	127	98 10	
				Hchl4s	127	98 10	
				Hchl6s	253	244 22	
				Hchl7s	263	241 40	
				Hchl8s	49	20 10	

38 problem instances of various sizes and densities. These test problem instances are standards and publicly available from (`ftp://PANDRAMIX.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting`). For the *unweighted version* (i.e., each c_i is equal to $l_i w_i$), 24 problem instances are considered and the other problem instances represent the *weighted version*. In Table 1, for each problem instance we report its dimensions $L \times W$ and the number of pieces to cut n.

The computational results of these instances are shown in Tables 2 and 3 for the approximate algorithm, and in Tables 4 and 5 for the exact algorithm. We report for each problem instance (with and without trimming):

- The *Approximate horizontal* (resp. *vertical*) solution value (denoted A^{hor}, resp. A^{ver}), which represents a *lower bound* produced by the approximate algorithm (algorithm of Section 3 when trimming is not permitted and algorithm of Section 4 otherwise).
- The *Optimal horizontal* (resp. *vertical*) solution value, denoted Opt^{hor} (resp. Opt^{ver}), reached by the exact algorithm. The optimal solution value, denoted *Opt*, when the first stage-cut is unspecified.
- The experimental *Approximation Ratio* (A.R.) obtained by applying the usual formula $\frac{A(I)}{Opt(I)}$, where I is an instance of the problem, $H(I)$ denotes the approximate solution value and $Opt(I)$ is the optimal solution value.
- The *Execution Time* (E.T., measured in seconds) which is the time that the considered approximate (resp. exact) algorithm takes to reach the approximate (resp. optimal) horizontal or vertical solution.
- The number of generated nodes, denoted `Nodes`, when the exact algorithm is applied.

7.1. Performance of the approximate algorithms

Table 2 shows the behavior of the approximate algorithm when *trimming is not permitted* (Columns 4, 8 and 11 respectively). For all treated instances, excellent lower bounds are obtained. We remark that A.R.s are generally close to one, except for some instances:

- The first-cut is unspecified: the instances `STS4`, `Hchl2`, `2s`, `STS4s`, `CHL6` and `CHL7`, for which the A.R.s varie between 0.973 and 0.993.
- The first-cut is horizontal: the instances `STS4`, `Hchl1`, `Hchl2`, `2s`, `STS4s`, `CHL1s`, `CHL6` and `CHL7` (the A.R.s are between 0.951 and 0.997).
- The first-cut is vertical: the instances `2` and `2s`, which realizes A.R.s equal to 0.862 and 0.856, respectivelly.

Also, we observe that the computational time is generally under 0.3 sec (Columns 5, 9 and 12 respectively).

When *trimming is permitted* (see Table 3), the approximate algorithm produces:

- The first-cut is unspecified: an A.R. which varies between 0.883 (instance `CW3`) and 1 (see Table 3, column 11) and by consuming less than 1 Sec.
- The first-cut is horizontal (resp. vertical): an A.R. which varies between 0.838 (instance `A2`) (resp. 0.883 (instance `CW3`)) and 1 (see Table 3, column 4 (resp. column 8)) and by consuming less than 1 Sec.

Figure 5 shows the different A.R.s produced by the approximate algorithm on the 38 problem instances. It shows that it produces better A.R.s when trimming is not permitted. This can be explained by the fact that the single knapsack problems are solved separately and they are totally independent, which is not the case for the non-exact case: in this case, we have applied a simple greedy algorithm for solving each set packing problem or we

Table 2. The effectiveness of the approximate algorithm on 38 problem instances representing the FC_2TDC problem: Trimming is not permitted.

	The FC_2TDC problem: Without trimming										
	The first stage-cut is horizontal				The first stage-cut is vertical				The first stage-cut is unspecified		
Inst	Opt^{hor}	A^{hor}	A.R.	E.T.	Opt^{ver}	A^{ver}	A.R.	E.T.	$\max\{A^{hor}, A^{ver}\}$	A.R.	E.T.
HH	10545	10545	1.000	<0.1	8298	8298	1.000	<0.1	10545^h	1.000	<0.1
2	2208	2208	1.000	<0.1	2022	1743	0.862	<0.1	2208^h	1.000	<0.1
3	1520	1520	1.000	<0.1	1440	1440	1.000	<0.1	1520^h	1.000	<0.1
A1	1820	1820	1.000	<0.1	1640	1640	1.000	<0.1	1820^h	1.000	<0.1
A2	1940	1940	1.000	<0.1	2240	2240	1.000	<0.1	2240^v	1.000	<0.1
STS2	4280	4280	1.000	<0.1	4200	4200	1.000	<0.1	4280^h	1.000	<0.1
STS4	9084	8965	0.987	<0.1	8531	8531	1.000	<0.1	8965^h	0.987	<0.1
CHL1	7019	7019	1.000	<0.1	7597	7597	1.000	<0.1	7597^v	1.000	0.1
CHL2	2076	2076	1.000	<0.1	1662	1662	1.000	<0.1	2076^h	1.000	<0.1
CW1	6040	6040	1.000	<0.1	5698	5698	1.000	<0.1	6040^h	1.000	<0.1
CW2	4562	4562	1.000	<0.1	4277	4277	1.000	<0.1	4562^h	1.000	0.1
CW3	4428	4428	1.000	0.1	5026	5026	1.000	0.1	5026^v	1.000	0.3
Hchl2	9340	8882	0.951	<0.1	9274	9274	1.000	<0.1	9274^v	0.993	0.1
Hchl9	4610	4610	1.000	0.1	4680	4680	1.000	0.1	4680^v	1.000	0.2
2s	2184	2177	0.997	<0.1	2022	1730	0.856	<0.1	2177^h	0.997	<0.1
3s	2405	2405	1.000	<0.1	2470	2470	1.000	<0.1	2470^v	1.000	<0.1
A1s	2950	2950	1.000	<0.1	2571	2571	1.000	<0.1	2950^h	1.000	<0.1
A2s	3372	3372	1.000	<0.1	3445	3445	1.000	<0.1	3445^v	1.000	<0.1
STS2s	4294	4294	1.000	<0.1	4342	4342	1.000	<0.1	4342^v	1.000	<0.1
STS4s	9185	9055	0.986	<0.1	8563	8563	1.000	<0.1	9055^h	0.986	<0.1
OF1	2437	2437	1.000	<0.1	2660	2660	1.000	<0.1	2660^v	1.000	<0.1
OF2	2307	2307	1.000	<0.1	2442	2442	1.000	<0.1	2442^v	1.000	<0.1
W	2470	2470	1.000	<0.1	2405	2405	1.000	<0.1	2470^h	1.000	0.1
CHL1s	12276	11729	0.955	<0.1	12314	12314	1.000	<0.1	12314^h	1.000	0.1
CHL2s	3162	3162	1.000	<0.1	2427	2427	1.000	<0.1	3162^h	1.000	<0.1
A3	5348	5348	1.000	<0.1	5082	5082	1.000	<0.1	5348^h	1.000	<0.1
A4	5109	5109	1.000	<0.1	5077	5077	1.000	<0.1	5109^h	1.000	<0.1
A5	12276	12276	1.000	<0.1	12276	12276	1.000	<0.1	$12276^{h,v}$	1.000	<0.1
CHL5	322	322	1.000	<0.1	344	344	1.000	<0.1	344^v	1.000	<0.1
CHL6	16325	16157	0.990	<0.1	15862	15862	1.000	<0.1	16157^h	0.988	0.1
CHL7	16556	16037	0.969	<0.1	16111	16111	1.000	<0.1	16111^v	0.973	0.1
CU1	12243	12243	1.000	<0.1	12200	12200	1.000	<0.1	12243^h	1.000	<0.1
CU2	26100	26100	1.000	<0.1	24750	24750	1.000	<0.1	26100^h	1.000	0.1
Hchl3s	11410	11410	1.000	<0.1	10369	10369	1.000	<0.1	11410^h	1.000	<0.1
Hchl4s	10545	10545	1.000	<0.1	8427	8427	1.000	<0.1	10545^h	1.000	<0.1
Hchl6s	56105	56105	1.000	<0.1	58121	58121	1.000	0.1	58121^v	1.000	0.1
Hchl7s	60384	60384	1.000	0.1	60683	60683	1.000	0.2	60683^v	1.000	0.3
Hchl8s	534	534	1.000	<0.1	631	631	1.000	<0.1	631^v	1.000	<0.1

$'h'$ if the optimal solution is realized when the first-stage cut is horizontal, $'v'$ otherwise.

Table 3. The effectiveness of the approximate algorithm on 38 problem instances representing the FC_2TDC problem: trimming is permitted.

	The FC_2TDC problem: With trimming										
	The first stage-cut is horizontal				The first stage-cut is vertical				The first stage-cut is unspecified		
Inst	Opt^{hor}	A^{hor}	A.R.	E.T.	Opt^{ver}	A^{ver}	A.R.	E.T.	$\max\{A^{hor}, A^{ver}\}$	A.R.	E.T.
HH	10689	10545	0.987	<0.1	9246	8298	0.897	<0.1	10545^{h}	0.987	<0.1
2	2535	2375	0.937	<0.1	2444	2305	0.943	<0.1	2375^{v}	0.937	<0.1
3	1720	1660	0.965	<0.1	1740	1440	0.828	<0.1	1660^{h}	0.954	<0.1
A1	1820	1820	1.000	<0.1	1820	1640	0.901	<0.1	1820^{h}	1.000	<0.1
A2	2315	1940	0.838	<0.1	2310	2240	0.970	<0.1	2240^{v}	0.968	<0.1
STS2	4450	4280	0.962	<0.1	4620	4620	1.000	<0.1	4620^{v}	1.000	0.1
STS4	9409	9263	0.984	<0.1	9468	8531	0.901	<0.1	9263^{h}	0.978	0.1
CHL1	8360	7421	0.888	0.1	8208	7597	0.926	0.1	7597^{v}	0.909	0.2
CHL2	2235	2076	0.929	<0.1	2086	2086	1.000	<0.1	2086^{v}	0.933	<0.1
CW1	6402	6402	1.000	<0.1	6402	6402	1.000	<0.1	$6402^{h,v}$	1.000	0.1
CW2	5354	5354	1.000	0.1	5159	5032	0.975	0.1	5354^{h}	1.000	0.3
CW3	5287	4434	0.839	0.3	5689	5026	0.883	0.3	5026^{v}	0.883	0.6
Hchl2	9630	9079	0.943	0.2	9528	9274	0.973	0.2	$9274^{h,v}$	0.963	0.3
Hchl9	5100	4610	0.904	<0.1	5060	4680	0.925	0.1	4680^{v}	0.918	0.1
2s	2430	2375	0.977	<0.1	2450	2188	0.893	<0.1	2375^{h}	0.969	<0.1
3s	2599	2470	0.950	<0.1	2623	2470	0.942	<0.1	$2470^{h,v}$	0.942	<0.1
A1s	2950	2950	1.000	<0.1	2910	2812	0.966	<0.1	2950^{h}	1.000	<0.1
A2s	3423	3423	1.000	<0.1	3451	3445	0.998	<0.1	3445^{v}	0.998	<0.1
STS2s	4569	4342	0.950	<0.1	4625	4342	0.939	<0.1	$4342^{h,v}$	0.939	<0.1
STS4s	9481	9258	0.976	<0.1	9481	8563	0.903	<0.1	9258^{h}	0.976	0.1
OF1	2713	2437	0.898	0.1	2660	2660	1.000	<0.1	2660^{v}	0.980	<0.1
OF2	2515	2307	0.917	<0.1	2522	2442	0.968	<0.1	2442^{v}	0.968	<0.1
W	2623	2470	0.942	<0.1	2599	2432	0.936	<0.1	2470^{h}	0.942	0.1
CHL1s	13036	12276	0.942	0.1	12602	12314	0.977	0.1	12314^{v}	0.945	0.2
CHL2s	3162	3162	1.000	<0.1	3198	3198	1.000	<0.1	3198^{v}	1.000	<0.1
A3	5380	5348	0.994	<0.1	5403	5082	0.941	<0.1	5348^{h}	0.990	<0.1
A4	5885	5885	1.000	<0.1	5905	5705	0.966	<0.1	5885^{h}	0.997	<0.1
A5	12553	12276	0.978	<0.1	12449	12276	0.986	<0.1	$12276^{h,v}$	0.978	0.1
CHL5	363	330	0.909	<0.1	344	344	1.000	<0.1	344^{v}	0.948	<0.1
CHL6	16572	16157	0.975	0.1	16281	15862	0.974	0.1	16157^{h}	0.975	0.2
CHL7	16728	16037	0.959	0.2	16602	16111	0.970	0.2	16111^{v}	0.963	0.3
CU1	12312	12243	0.994	<0.1	12200	12200	1.000	<0.1	12243^{h}	0.950	0.1
CU2	26100	26100	1.000	0.1	25260	24750	0.980	0.2	26100^{h}	1.000	0.4
Hchl3s	11961	11410	0.954	<0.1	11829	10836	0.916	<0.1	11410^{h}	0.954	<0.1
Hchl4s	11408	10545	0.924	<0.1	11258	9573	0.850	<0.1	10545^{h}	0.924	<0.1
Hchl6s	60170	56105	0.932	0.1	59853	58121	0.971	0.1	58121^{v}	0.966	0.3
Hchl7s	62459	60384	0.967	0.4	62845	60683	0.966	0.4	60683^{v}	0.966	0.8
Hchl8s	729	662	0.908	<0.1	791	715	0.904	<0.1	715^{v}	0.904	<0.1

$'h'$ means that the optimal solution is realized when the first-stage cut is horizontal, $'v'$ otherwise.

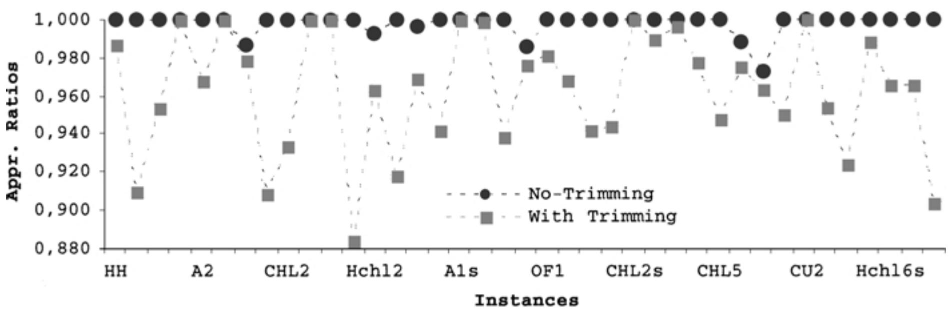

Figure 5. The behavior of the approximate algorithms on the different problem instances.

have applied a simple estimation for constructing the second single knapsack problem (see Section 4).

Therefore, by including these good quality lower bounds and by exploiting some dynamic programming properties, in general tree search procedures, one also expects good behavior from the resulting algorithms.

7.2. *Performance of the exact algorithms*

We examine now the performance of the proposed exact algorithm for the two cases: with and without trimming. In order to evaluate the effectiveness of the initial lower bound and the different used strategies, we have considered two versions of the algorithm:

- The first version uses the initial lower bound (at the root node) and the different strategies.
- The second algorithm runs without using the lower bound and the duplicate strategies.

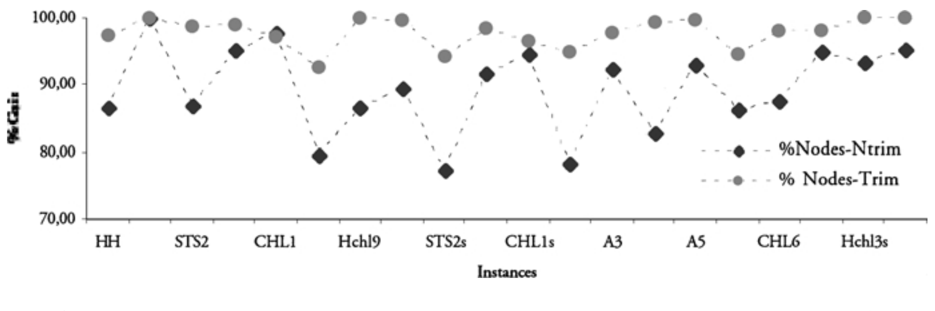

Figure 6. Variation of the percentage generated nodes on some problem instances: Comparison of the two versions of the algorithm.

Table 4. Performance of the different versions of the exact algorithm:*Without trimming*.

	The FC_2TDC problem: Without trimming								
	The first stage-cut is horizontal			The first stage-cut is vertical			The first stage-cut is unspecified		
Inst	Opt^{hor}	`Nodes`	E.T.	Opt^{ver}	`Nodes`	E.T.	Opt	`Nodes`	E.T.
`HH`	10545	24	0.2	8298	46	0.2	10545^h	33	0.2
`2`	2208	373	0.3	2022	72	0.1	2208^h	400	0.3
`3`	1520	88	0.1	1440	21	0.1	1520^h	106	0.1
`A1`	1820	28	0.1	1640	30	0.1	1820^h	50	0.2
`A2`	1940	63	0.2	2240	39	0.2	2240^v	61	0.2
`STS2`	4280	59	0.4	4200	76	0.4	4280^h	122	0.4
`STS4`	9084	167	0.6	8531	96	0.6	9084^h	208	0.6
`CHL1`	7019	241	1.2	7597	1318	3.5	7597^v	1413	3.7
`CHL2`	2076	18	0.1	1662	20	0.1	2076^h	28	0.1
`CW1`	6040	24	1.0	5698	16	0.9	6040^h	32	1.0
`CW2`	4562	51	2.1	4277	23	2.1	4562^h	67	2.2
`CW3`	4428	24	6.2	5026	31	6.1	5026^v	36	6.3
`Hchl2`	9340	2253	6.7	9274	342	1.9	9340^h	1004	2.8
`Hchl9`	4610	269	0.6	4680	209	0.6	4680^v	431	0.7
`2s`	2184	341	0.2	2022	58	0.1	2184^h	368	0.3
`3s`	2405	25	0.1	2470	21	0.1	2470^v	42	0.1
`A1s`	2950	1	0.1	2571	21	0.1	2950^h	5	0.2
`A2s`	3372	14	0.2	3445	11	0.2	3445^v	21	0.2
`STS2s`	4294	80	0.4	4342	67	0.4	4342^v	125	0.5
`STS4s`	9185	176	0.6	8563	88	0.6	9185^h	212	0.6
`OF1`	2437	17	0.1	2660	23	0.1	2660^v	33	0.1
`OF2`	2307	42	0.1	2442	23	0.1	2442^v	44	0.1
`W`	2470	21	0.1	2405	25	0.1	2470^h	42	0.1
`CHL1s`	12276	213	1.2	12314	274	1.2	12314^v	412	1.3
`CHL2s`	3162	17	0.1	2427	25	0.1	3162^h	26	0.1
`A3`	5348	28	0.2	5082	63	0.3	5348^h	60	0.3
`A4`	5109	27	0.2	5077	153	0.3	5109^h	159	0.4
`A5`	12276	85	0.7	12276	98	0.7	$12276^{h,v}$	183	0.8
`CHL5`	322	49	0.1	344	23	0.1	344^v	52	0.1
`CHL6`	16352	168	1.5	15862	122	1.5	16352^h	224	1.6
`CH7`	16556	576	2.1	16111	387	1.9	16556^h	636	2.2
`CU1`	12243	11	1.0	12200	2	1.0	12243^h	12	1.0
`CU2`	26100	2	2.7	24750	21	2.6	26100^h	3	2.7
`Hchl3s`	11410	219	0.6	10369	221	0.6	11410^h	256	0.6
`Hchl4s`	10545	130	0.4	8427	351	0.6	10545^h	165	0.4
`Hchl6s`	56105	170	4.3	58121	99	4.3	58121^h	160	4.5
`Hchl7s`	60384	319	7.6	60683	1347	9.6	60683^v	1610	10.4
`Hchl8s`	534	72	0.1	631	273	0.1	631^v	319	0.2

Table 5. Performance of the different versions of the exact algorithm when *trimming is permitted*. N/A means that the algorithm takes more than two hours for producing the optimal solution.

	The FC_2TDC problem: With trimming								
	The first stage-cut is horizontal			The first stage-cut is vertical			The first stage-cut is unspecified		
Inst	Opt^{hor}	Nodes	E.T.	Opt^{ver}	Nodes	E.T.	Opt	Nodes	E.T.
HH	10689	79	0.2	9246	360	0.3	10689^h	157	0.2
2	2535	1789	2.9	2444	1935	3.8	2535^h	3013	6.8
3	1720	258	0.2	1740	238	0.2	1740^v	462	0.3
A1	1820	254	0.2	1820	405	0.3	1820^h	496	0.4
A2	2315	820	0.8	2310	813	0.7	2315^h	1274	1.3
STS2	4450	2218	5.6	4620	137	0.5	4620^v	388	0.7
STS4	9409	3411	9.2	9468	1567	1.9	9468^v	3348	6.9
CHL1	8360	18643	610.1	8208	49196	1496.3	8360^h	19315	595.6
CHL2	2235	133	0.1	2086	96	0.1	2235^h	187	0.1
CW1	6402	44	1.0	6402	2	1.0	$6402^{h,v}$	46	1.1
CW2	5354	87	2.1	5159	205	2.2	5354^h	144	2.4
CW3	5287	205	6.4	5689	136	6.3	5689^v	194	6.7
Hchl2	9630	87202	N/A	9528	132272	N/A	9630^h	160114	N/A
Hchl9	5100	23428	858.0	5060	30497	1017.4	5100^h	42528	1567.0
2s	2430	1923	4.6	2450	2447	4.8	2450^v	3201	9.4
3s	2599	148	0.1	2623	231	0.1	2623^h	372	0.2
A1s	2950	3	0.1	2910	67	0.2	2950^h	7	0.2
A2s	3423	53	0.2	3451	22	0.2	3451^v	59	0.2
STS2s	4569	1015	1.1	4625	801	0.8	4625^v	1423	0.6
STS4s	9481	3710	8.9	9481	1727	1.9	$9481^{h,v}$	4219	10.6
OF1	2713	63	0.1	2660	37	0.1	2713^h	67	0.1
OF2	2515	145	0.1	2522	101	0.1	2522^v	167	0.1
W	2623	231	0.2	2599	177	0.1	2623^h	372	0.2
CHL1s	13036	2723	6.1	12602	5416	21.5	13036^h	3174	7.1
CHL2s	3162	93	0.1	3198	55	0.1	3198^v	118	0.1
A3	5380	110	0.3	5403	550	0.4	5403^h	272	0.4
A4	5885	958	1.6	5905	1600	2.1	5905^h	1832	3.1
A5	12553	1264	2.4	12449	1601	2.5	12553^h	2411	4.2
CHL5	363	151	0.1	344	115	0.1	363^h	231	0.1
CHL6	16572	5515	38.2	16281	5569	33.4	16572^h	6236	44.5
CH7	16728	7159	44.6	16602	6753	50.6	16728^h	9449	59.4
CU1	12312	31	1.0	12200	3	1.0	12312^h	22	1.2
CU2	26100	2	2.7	25260	154	2.8	26100^h	3	2.9
Hchl3s	11961	5251	15.3	11829	5947	20.4	11961^h	5832	18.8
Hchl4s	11408	18317	507.4	11258	16067	268.4	11408^h	22199	612.7
Hchl6s	60170	3937	14.8	59853	1853	7.2	60170^h	3698	11.8
Hchl7s	62459	10184	96.2	62845	6217	39.1	62845^v	9152	54.5
Hchl8s	729	4204	26.4	791	4603	28.7	791^v	7073	59.0

In Table 4 we can observe the performance of the exact algorithm when trimming is not permitted. We remark, generally, that the algorithm runs under 10 sec when the first stage-cut is specified and nearly 10 sec if the first stage-cut is unspecified.

In Table 5 we remark the performance of the exact algorithm when trimming is permitted. In this case, the algorithm is great time consuming. Indeed, in this case the construction procedure generates more horizontal and vertical builds which means that the algorithm evaluates a large number of nodes. This last fact draws an explanation for the superiority of the method to the non-exact case, i.e., trimming is not permitted.

We have already mentioned that we compare two versions of the algorithm. In figures 6 and 7, the percentage average generated nodes and computing times are shown for some problem instances of Table 1. These figures represents the results of Table 6 in which the performance of the second version of the algorithm is reported. It is clear that the first version of the algorithm largely outperforms the second version of the algorithm.

Table 6. Comparison of two versions of the algorithm: The effect of the lower bound and symmetric strategies.

	Without trimming				With trimming			
Inst	Nodes	E.T.	% Nodes	% E.T.	Nodes	E.T.	% Nodes	% E.T.
HH	249	0.3	86.75	33.33	1297	0.8	97.46	75.00
2	79836	194.7	99.92	99.85	181307	N/A	99.97	100.00
STS2	937	0.7	86.98	42.86	10492	7.0	98.84	94.29
STS4	4230	5.0	95.08	88.00	19278	41.8	98.92	98.56
CHL1	66600	1770	97.88	99.79	50470	1133.3	97.20	99.67
CHL2	138	0.2	79.71	50.00	379	0.2	92.61	50.00
Hchl9	3181	2.2	86.45	68.18	552155	N/A	99.92	100.00
2s	3525	4	89.56	92.50	169277	1635.7	99.78	99.98
STS2s	551	0.6	77.31	16.67	2208	1.1	94.34	54.55
STS4S	2554	1.1	91.70	45.45	12603	29.0	94.34	54.55
CHL1s	7604	8.1	94.58	83.95	12090	20.2	96.59	93.56
CHL2s	120	0.2	78.33	50.00	517	0.2	94.97	50.00
A3	791	0.4	92.41	25.00	2611	0.7	97.70	57.14
A4	928	0.5	82.87	20.00	20155	71.3	99.21	99.44
A5	2561	1.6	92.85	50.00	69559	934.1	99.74	99.91
CHL5	376	0.2	86.17	50.00	955	0.4	94.55	75.00
CHL6	1788	2.1	87.47	23.81	11789	61.7	98.10	97.41
CHL7	12218	6.2	94.79	64.52	30992	170.4	97.95	98.71
Hchl3s	3853	3.2	93.36	81.25	226816	N/A	99.89	100.00
Hchl4s	3525	4.2	95.32	90.48	235686	186.6	99.93	99.79

N/A means that the algorithm is not able to produce the optimal solution after two hours.

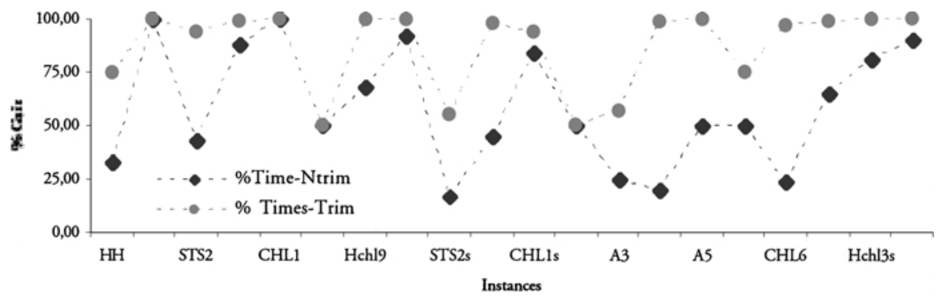

Figure 7. The computing time improvement of the first version of the algorithm.

As one can see from both figures (percentage nodes and times, respectively), the slope of the curve corresponding to the trimming case (Columns 4 and 5) is generally greater than the slope of the curve corresponding the the no-trimming case (Columns 8 and 9). This fact constituting a confirmation of the hardness of the problem when trimming is considered.

8. Conclusion

In this paper, we have considered a variant of the constrained two-dimensional cutting stock problem, namely the constrained two-staged two-dimensional cutting. We have considered two versions of the problem: with and without trimming. We have proposed two algorithms.

First, an approximate algorithm which is based upon the classical dynamic programming techniques. It consists of decomposing the original problem to a series of single bounded knapsack problems and solve them by applying a dynamic programming procedure. If trimming is permitted, another set packing problem is approximately solved.

Second, an exact algorithm is proposed for solving both with and without trimming cases of the problem. The algorithm is principally based upon branch and bound procedure combined with dynamic programming techniques. We also used two strategies for neglecting some duplicate patterns.

The empirical results of the algorithms has been reported through a number of experiments. These experiments were conducted on some problem instances of the literature with different sizes and densities. The computational results indicate that the proposed approaches are performant and show that the algorithms are able to solve efficiently the different problem instances within reasonable computing times.

References

A. Albano and G. Sapuppo, “Optimal allocation of two-dimensional irregular shapes using heuristic search methods,” *IEEE, Trans. Sys. Man. Cyb.*, vol. 10, no 5, pp. 242–248, 1980.

M. Arenales and R. Morabito, “An overview of and-or-graph approach to cutting and packing problems,” ISBN 5-86911-161-7, 1997, pp. 207–224.

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

N. Christofides and E. Hadjiconstantinou, “An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts,” *European Journal of Operational Research*, vol. 83, pp. 21–38, 1995.

N. Christofides and C. Whitlock, “An algorithm for two-dimensional cutting problems,” *Operations Research*, vol. 2, pp. 31–44, 1977.

V-D. Cung, M. Hifi, and B. Le Cun, “Constrained two-dimensional cutting stock problems: A best-first branch-and-bound algorithm,” *International Transactions in Operational Research*, vol. 7, pp. 185–210, 2000.

H. Dyckhoff, “A typology of cutting and packing problems,” *European Journal of Operational Research*, vol. 44, pp. 145–159, 1990.

D. Fayard, M. Hifi, and V. Zissimopoulos, “An efficient approach for large-scale two-dimensional guillotine cutting stock problems,” *Journal of the Operational Research Society*, vol. 49, pp. 1270–1277, 1998.

P.C. Gilmore, “Cutting stock, linear programming, knapsacking, dynamic programming and integer programming, some interconnections,” *Annals of Discrete Mathematics*, vol. 4, pp. 217–235, 1979.

P.C. Gilmore and R.E. Gomory, “Multistage cutting problems of two and more dimensions,” *Operations Research*, vol. 13, pp. 94–119, 1965.

P.C. Gilmore and R.E. Gomory, “The theory and computation of knapsack functions,” *Operations Research*, vol. 14, pp. 1045–1074, 1966.

J.C. Herz, “A recursive computing procedure for two-dimensional stock cutting,” *IBM Journal of Research and Development*, vol. 16, pp. 462–469, 1972.

M. Hifi, “An improvement of Viswanathan and Bagchi’s exact algorithm for cutting stock problems,” *Computers and Operations Research*, vol. 24, pp.727–736, 1997.

M. Hifi, “Contribution à la résolution de quelques problèmes difficiles de l’optimisation combinatoire,” Habilitation Thesis. PR*i*SM, Université de Versailles St-Quentin en Yvelines, 1999.

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

M. Hifi and V. Zissimopoulos, “A recursive exact algorithm for weighted two-dimensional cutting problems,” *European Journal of Operational Research*, vol. 91, pp. 553–564, 1996.

M. Hifi and V. Zissimopoulos, “Constrained two-dimensional cutting: An improvement of Christofides and Whitlock’s exact algorithm,” *Journal of the Operational Research Society*, vol. 5, pp. 8–18, 1997.

E.L. Lawler, “Fast approximation algorithms for knapsack problems,” *Mathematics of Operations Research*, vol. 4, pp. 339–356, 1979.

S. Martello and P. Toth, “An exact algorithm for large unbounded knapsack problems,” *Operations Research Letters*, vol. 9, pp. 15–20, 1990.

S. Martello and P. Toth, “Upper bounds and algorithms for hard 0-1 knapsack problems,” *Operations Research*, vol. 45, pp. 768–778, 1997.

R. Morabito and M. Arenales, “Staged and constrained two-dimensional guillotine cutting problems: An and-or-graph approach,” *European Journal of Operational Research*, vol. 94, no. 3, pp. 548–560, 1996.

R. Morabito, M. Arenales, and V. Arcaro, “An and-or-graph approach for two-dimensional cutting problems,” *European Journal of Operational Research*, vol. 58, no. 2, pp. 263–271, 1992.

R. Morabito and V. Garcia, “The cutting stock problem in hardboard industry: A case study,” *Computers and Operations Research*, vol. 25, pp. 469–485, 1998.

D. Pisinger, “A minimal algorithm for the 0-1 knapsack problem,” *Operations Research*, vol. 45, pp. 758–767, 1997.

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

M. Syslo, N. Deo, and J. Kowalik, *Discrete Optimization Algorithms*, Prentice-Hall: New Jersey, 1983.

K.V. Viswanathan and A. Bagchi, “Best-first search methods for constrained two-dimensional cutting stock problems,” *Operations Research*, vol. 41, no. 4, pp. 768–776, 1993.

P.Y. Wang, “Two algorithms for constrained two-dimensional cutting stock problems,” *Operations Research*, vol. 31, no. 3, pp. 573–586, 1983.