Computational Optimization and Applications, 18, 63–88, 2001

Exact Algorithms for Large-Scale Unconstrained Two and Three Staged Cutting Problems

MHAND HIFI hifi@univ-paris1.fr
CERMSEM, Maison des Sciences Economiques, Université de 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

Received March 11, 1998; Accepted June 28, 1999

Abstract. In this paper we propose two exact algorithms for solving both two-staged and three staged unconstrained (un)weighted cutting problems. The two-staged problem is solved by applying a dynamic programming procedure originally developed by Gilmore and Gomory [Gilmore and Gomory, Operations Research, vol. 13, pp. 94–119, 1965]. The three-staged problem is solved by using a top-down approach combined with a dynamic programming procedure. The performance of the exact algorithms are evaluated on some problem instances of the literature and other hard randomly-generated problem instances (a total of 53 problem instances). A parallel implementation is an important feature of the algorithm used for solving the three-staged version.

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

1. Introduction

Cutting and Packing problems belong to an old and very well-known family, called CP in Dyckhoff [7]. This is a family of natural combinatorial optimization problems, admitted in numerous real-world applications from computer science, industrial engineering, logistics, manufacturing, production process, etc. Long and intensive research on the problems of this family has led (and leads always) to the development of models and mathematical tools, so interesting by themselves that their application domains surpass the framework of CP problems.

We study in this paper the problem of cutting a large rectangle S of dimensions $L \times W$, into n small rectangles (called pieces) of values (weights) c_i and dimensions $l_i \times w_i$, $i = 1, \ldots, n$. On one hand, we say that all pieces have *fixed orientation* if each piece of length l and width w is different from a piece of length w and width l (when $l \neq w$). On the other hand, if each piece can be *rotated* of 90°, then we consider that the dimensions (l_i, w_i) and (w_i, l_i) represent the dimensions of the same piece i (see figure 1(a)).

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(b)).

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 large rectangle S without overlapping. The Two-Dimensional Cutting problem (shortly

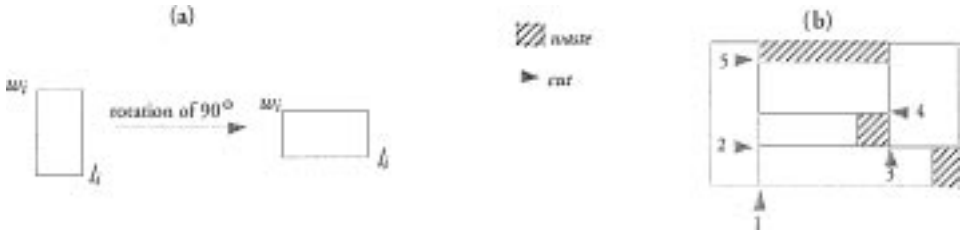

Figure 1. (a) representation of the same piece i before (the left side) and after (the right side) applying a rotation of 90°, and (b) guillotine cutting.

TDC) consists of determining the cutting pattern (using guillotine cuts) with the maximum value, i.e.

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

In TDC problem one has, in general, to satisfy a certain demand of pieces. The problem is called *unconstrained* if these demand values are not imposed, and *constrained* if the demand is represented by a vector $b = (b_1, \ldots, b_n)$, where b_i is the maximum number of times that piece of type $i, i = 1, \ldots, n$, can appear in a feasible cutting pattern. In this case, the cutting pattern of Eq. (1) can be represented by $(x_1, \ldots, x_n) \leq (b_1, \ldots, b_n)$.

We distinguish two versions of the (un)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. $c_i \neq l_i w_i$.

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 2 shows the 2TDC and 3TDC solutions respectively. For example, the solution given by figure 2(a) is produced by applying the following phases:

Phase 1. the large 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 the 3TDC, another cut can be applied (see figure 2(b)) on each resulting substrip produced after the second phase (*Phase* 2).

For simplicity, for the unconstrained (un)weighted kTDC problem, we introduce the following notations:

- FUU_kTDC: represents the Fixed Unconstrained Unweighted kTDC problem;
- RUW_kTDC: denotes the Rotated Unconstrained Weighted kTDC problem;
- FU_kTDC: corresponds to the Fixed Unconstrained (un)weighted kTDC problem;
- RU_kTDC: denotes the Rotated Unconstrained (un)weighted kTDC problem;

Figure 2. Representation of two solutions when k is fixed to 2 and 3 respectively; Numbers by the cuts are the stage at which the cut is made. (a) a solution of 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); (b) a solution to the 3TDC problem: The value '1' represents the first cut, '2' denotes the second cut and '3' is the third cut; in this case, they are: one vertical cut (counted one time), 5 horizontal cuts (counted one time) and four internal vertical cuts (counted one time).

- UU_kTDC: is the Unconstrained Unweighted kTDC problem. We use this notation if both *fixed* and *rotated* versions are considered;
- UW_kTDC: represents the Unconstrained Weighted kTDC problem. Also, the notation is used if both *fixed* and *rotated* versions are considered;
- U_kTDC: corresponds to the Unconstrained (un)weighted kTDC problem. This notation is used if we consider (a) unweighted and weighted versions, and (b) the fixed and the rotated cases.

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

- (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 3(a). The instance is represented by a large rectangle of dimensions $(L, W) = (11, 14)$ and three pieces p_1, p_2 and p_3 respectively of dimensions (6, 5), (4, 5) and (1, 4) respectively. We consider the unweighted version of the cutting problem.

Figure 3(b.1) shows a *uniform horizontal strip* with dimensions (11, 4). We can remark that the strip is composed of the same piece with width $\beta = 4$ (the value of this strip is equal to 44). 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 3(b.2) shows another *uniform horizontal strip* with dimensions (11, 5) (with value 50). This strip is composed of two pieces p_1 and p_2 respectively, having the same width $\beta = 5$. It is also obtained without trimming.

Now, consider figure 3(b.3). The obtained strip (of dimensions $(L, \beta) = (11, 5)$ and with value 54) represents a *general horizontal strip*, because some pieces have different widths (in this example, the width of p_1 is equal to 5 and the width of p_3 is equal to 4). We

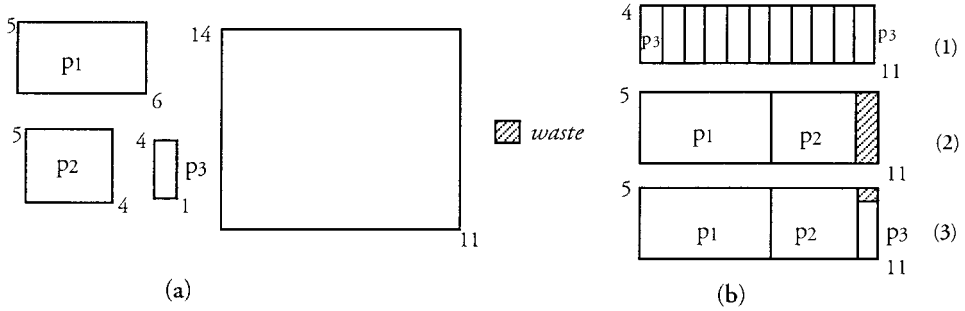

Figure 3. Representation of (a) a large rectangle with three pieces and (b) three different strips; (b.1) and (b.2) are two strips without trimming and (b.3) a strip with trimming.

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 3(b.3), the piece p_3 is obtained by applying a supplementary horizontal cut).

According to the above definitions and the strips illustrated by figures 3(b), we can distinguish two cases of the U_*k*TDC problem. The first one is considered when the trimming is permitted (called the nonexact case in Gilmore and Gomory [11], pp. 101, 102) and the second one represents the U_*k*TDC problem without trimming (called the exact case in Gilmore and Gomory [11]).

In this paper we treat the following variants of the U_*k*TDC problem:

- the RU_2TDC problem with and without trimming;
- the FU_3TDC and RU_3TDC problems with and without trimming.

The paper is organized as follows. First (Section 2), we repeat briefly Gilmore and Gomory's procedure ([11]) applied especially for providing an optimal solution to the FU_2TDC problem (with and without trimming). An adaptation of this procedure is presented in Section 2.3 applied especially for solving the RU_2TDC problem with and without trimming. In Section 2.3.2, a faster algorithm is proposed for solving the same problem. Second (Section 3), we develop a new algorithm for solving both FU_3TDC and RU_3TDC problems. The algorithm is based upon a top-down approach using a depth-first search strategy combined with dynamic programming techniques. The performance of the proposed algorithms are presented in Section 4. A set of large size problem instances is considered and benchmark results are given. Finally, we discuss some interesting perspectives of the approach tailored for both FU_3TDC and RU_3TDC problems. For the staged TDC problem, to our knowledge, only few exact and approximate algorithms are known (example: Beasley ([3]), Gilmore and Gomory ([11, 12]), Morabito and Arenales ([25]).

2. The unconstrained (un)weighted two staged cutting problem

In ([11], pp. 101, 102), Gilmore and Gomory proposed an efficient optimal procedure for solving the FU_2TDC problem. Several authors have used their procedure for solving some

variants of TDC (stock) problems (see Hifi et al. ([14, 16–19]), Morabito et al. ([24, 25, 27]), Fayard et al. [8, 9, 31]). An exact algorithm has been proposed in [18] for solving unconstrained (un) weighted TDC problems in which the solution given by Gilmore and Gomory's procedure has been used as an initial lower bound. For the same problem, in [24] the authors have also compared the performance of Gilmore and Gomory's procedure to the heuristic due to Beasley ([3]), on a set of randomly generated problem instances, and proved that the procedure out-performs the Beasley's algorithm. In [15], we have developed a hybrid approach combining a depth-first search and dynamic programming techniques for solving unconstrained (un)weighted TDC problems. In the same paper, it was proved that the approach outperforms some existing algorithms on a set of problem instances of the literature.

In what follows Gilmore and Gomory's procedure is denoted DPP which means that Dynamic Programming Procedures are used for solving optimally the U_2TDC problem and approximately the unconstrained (un)weighted TDC problem.

2.1. The optimal horizontal pattern for the FU_2TDC

The problem of generating an optimal solution for the horizontal FU_2TDC problem can be decomposed into two phases. First, pieces are chosen to be combined in order to construct a set of horizontal strips. Second, all obtained strips are chosen to be combined along the width of the initial rectangle.

Gilmore and Gomory's procedure can be summarized as follows: without loss of generality, assume that the rectangular pieces are ordered in increasing order such that $w_1 \leq w_2 \leq \cdots \leq w_n$. Suppose that r denotes the number of different widths of the rectangular pieces, and (L, w_j) is a strip with width w_j. We define the one-dimensional *unbounded* knapsack problems (K_{L,w_j}), for $j = 1, \ldots, r$, as follows:

$$\left(K_{L,w_j}\right) \begin{cases} f_j^{\text{hor}}(L) = \max & \displaystyle\sum_{i \in S_{L,w_j}} c_i x_i \\ \text{subject to} & \displaystyle\sum_{i \in S_{L,w_j}} l_i x_i \leq L \\ & X_i \in \mathbb{N}, i \in S_{L,w_j} \end{cases}$$

where S_{L,w_j} is the set of rectangular pieces entering in the strip (L, w_j) (i.e. $\forall i \in S_{L,w_j}$, we have $(l_i, w_i) \leq (L, w_j)$), 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 $i \in S_{L,w_j}$ and $f_j^{\text{hor}}(L)$ is the solution value of the j-th strip, $j = 1, \ldots, r$.

Notice that it suffices to solve the largest knapsack K_{L,w_r} by using a dynamic programming procedure (see Lawler ([21], and Martello and Toth ([22, 23])) for generating r optimal strips with respect to the length of the initial rectangle and the widths w_j, $j = 1, \ldots, r$. Each strip j, $j = 1, \ldots, r$, is characterized by its solution value $f_j^{\text{hor}}(L)$ and its width w_j.

By selecting the best of these strips, an *optimal horizontal cutting pattern* is constructed. This can be realized, by solving the following one-dimensional unbounded knapsack

problem $(K^{\text{hor}}_{(L,W)})$:

$$\left(K^{\text{hor}}_{(L,W)}\right)\begin{cases} g^{\text{hor}}_L(W) = \max & \displaystyle\sum_{j=1}^{r} f^{\text{hor}}_j(L)y_j \\ \text{subject to} & \displaystyle\sum_{j=1}^{r} w_j y_j \le W,\ y_j \in \mathbb{N} \end{cases}$$

where y_j, $j = 1, \ldots, r$, is the number of occurrences of the j-th strip in the initial rectangle S, $f^{\text{hor}}_j(L)$ is the solution value of the j-th strip and $g^{\text{hor}}_L(W)$ is the solution value of the cutting pattern, for the rectangle (L, W), composed by some horizontal strips.

2.2. The optimal vertical pattern for the FU_2TDC problem

By applying the same process as in Section 2.1, we can generate the set of the vertical strips such that we reverse the role of widths and lengths in the previous knapsack problems K_{L,w_j} and $K^{\text{hor}}_{(L,W)}$, i.e. w_i by l_i, l_i by w_i, L by W,W by L, $f^{\text{hor}}_j(L)$ by $f^{\text{ver}}_j(W)$, $g^{\text{hor}}_L(W)$ by $g^{\text{ver}}_W(L)$, S_{L,w_j} by $S_{l_j,W}$, and r by s, where s is the number of distinct lengths. We denote these knapsack problems $K_{l_j,W}$, and $K^{\text{ver}}_{(L,W)}$ respectively. For more clarity, we give the two unbounded knapsack problems and for more details, the reader can be refereed to [11, 14].

By using the above permutations and by considering the order $l_1 \le l_2 \le \ldots, \le l_n$, we construct the following knapsack problems (for $j = 1, \ldots, s$):

$$\left(K_{l_j,W}\right)\begin{cases} f^{\text{ver}}_j(W) = \max & \displaystyle\sum_{i\in S_{l_j,W}} c_i x_i \\ \text{subject to} & \displaystyle\sum_{i\in S_{l_j,W}} w_i x_i \le W \\ & x_i \in \mathbb{N},\ i \in S_{l_j,W} \end{cases}$$

$$\left(K^{\text{ver}}_{(L,W)}\right)\begin{cases} g^{\text{ver}}_W(L) = \max & \displaystyle\sum_{j=1}^{r} f^{\text{ver}}_j(W)y_j \\ \text{subject to} & \displaystyle\sum_{j=1}^{r} l_j y_j \le L \\ & y_j \in \mathbb{N} \end{cases}$$

As previously, the set of vertical strips is obtained by solving the problem $K_{l_s,W}$ and the *optimal vertical pattern* (with value $g^{\text{ver}}_W(L)$) is obtained by solving the resulting knapsack problem $(K^{\text{ver}}_{(L,W)})$.

The optimal solution: the optimal solution of the FU_2TDC problem is the one that realizes the best solution value between horizontal and vertical two-staged patterns, i.e. the pattern realizing the value $\max\{g^{\text{hor}}_L(W), g^{\text{ver}}_W(L)\}$. Of course, we consider that the first-stage cut is unspecified (i.e. the first cut is made horizontally or vertically), which represents a natural version of the problem.

Remark 1. We recall that there exists two cases of the problem: in the first case, trimming is not permitted and in the second one, the trimming is permitted (see figure 3(b)).

For the first case, the set of distinct widths (resp. lengths) is divided into r (resp. s) different (small) knapsack problems and each strip is obtained independently from the other ones. Each generated (sub) strip contains a subset of pieces with the same widths (resp. lengths). For the second case, one knapsack problem is considered and each strip with width $w_j > 0$, $j = 1, \ldots, r$, (resp. length $l_j > 0$, $j = 1, \ldots, s$) depends on the value of the $(j-1)$-th strip where $w_{j-1} < w_j$ and $w_0 = 0$ (resp. $l_{j-1} < l_j$ and $l_0 = 0$).

2.3. The optimal solution for the RU_2TDC problem

Now, consider that each piece i of S can be rotated by 90 degrees. In this case, we can easily present an adaptation of the previous algorithm DPP and a faster one (called MDPP) for solving optimally the problem. Computational results show that the second version turns more efficient than the first one, especially for large-scale problem instances.

2.3.1. The DPP applied to the RU_2TDC problem. A simple phase can be added to the previous ones for solving the RU_2TDC problem. It consists in considering that the number of pieces is doubled (equals now to $2 \times n$ pieces), the original pieces are indexed from 1 to n and the other ones are indexed from $n+1$ to $2 \times n$. Of course, if there exists two identical pieces (i.e. their dimensions coincide), then the number $2 \times n$ is reduced. So, the optimal solution can be provided by applying the DPP to the resulting problem.

2.3.2. A Modified version of DPP: MDPP. Remark that the DPP solves four unbounded knapsack problems. The two first knapsack problems are independently applied for constructing the sets of horizontal and vertical strips. The two other knapsack problems are used for combining independently the different strips in order to produce the optimal solution. The following result proves that the optimal solution of the problem can be produced by considering three knapsack problems. For a particular case, the three knapsack problems can be reduced to only two knapsack problems.

Theorem 1. *The optimal solution for both RUU_2TDC and RUW_2TDC problems can be obtained by considering three unbounded knapsack problems.*

Proof: Suppose that the number of the different pieces is equal to n' ($\leq 2 \times n$). Then, order all pieces in increasing order of widths such that $w_1 \leq w_2 \leq \cdots \leq w_k \leq \cdots \leq w_{n'}$ and let r' be the number of distinct widths. Consider $\mathcal{T} = \max\{L, W\}$ and the following unbounded knapsack problems.

$$\left(K_{\mathcal{T},w_j}\right) \begin{cases} f_j^{\text{hor}}(\mathcal{T}) = \max & \displaystyle\sum_{i \in S_{\mathcal{T},w_j}} c_i x_i \\ \text{subject to} & \displaystyle\sum_{i \in S_{\mathcal{T},w_j}} l_i x_i \leq \mathcal{T} \\ & x_i \in \mathbb{N},\ i \in S_{\mathcal{T},w_j} \end{cases}$$

By solving $K_{\mathcal{T},w_{r'}}$, all strips with different widths w_j, $j = 1, \ldots, n$, and with length $\mathcal{T}$ are obtained. This is possible because dynamic programming techniques are used.

Since $L \leq \mathcal{T}$ and $W \leq \mathcal{T}$ then: (i) the set of horizontal strips with their values $f_j^{\text{hor}}(L)$, $j = 1, \ldots, r'$, are obtained, and (ii) the set of vertical strips with their values $f_j^{\text{ver}}(W)$ are also available.

The case (i) is evident because we consider that the length is represented by $\mathcal{T}$. For the case (ii), it suffices to remark that if there exists a piece of width w_j, then necessarily the same piece exists with a length $\ell = w_j$ and so, by considering $W \leq \mathcal{T}$, all substrips with different lengths $l_i, i = 1, \ldots, r'$ and with length W are available. By using a rotation of 90 degrees on each of these strips, we obtain the set of vertical strips on (L, W).

Now, the aim is to construct (1) the optimal horizontal 2-staged pattern and (2) the optimal vertical 2-staged one. Consider that $L > W$ (if $L < W$, it suffices to inverse the role of each of them).

Let $K^{\text{hor}}_{(L,\mathcal{T})}$ be the knapsack problem obtained by replacing in $K^{\text{hor}}_{(L,W)}$, W by $\mathcal{T}$ and r by r'. Clearly, the *optimal horizontal pattern* is obtained by solving $K^{\text{hor}}_{(L,\mathcal{T})}$ in which $\mathcal{T}$ is fixed to W. It is clear that if the horizontal pattern is created, then the vertical can also be obtained. This is possible because all substrips with lengths $l_i, i = 1, \ldots, r'$, with their different values $f_i^{\text{ver}}(W)$, are also available (we just apply a rotation as above).

This means that, with some permutations, the *optimal vertical pattern* can be obtained by solving another knapsack problem. □

Remark 2. The optimal solution of the RU_2TDC problem is obtained by considering only two knapsack problems when $L = W$. Indeed, the value of $\mathcal{T}$ corresponds to $L(= W)$ and the two sets of linear combinations coincide, i.e. $\mathcal{L} = \mathcal{W}$, and so, the optimal horizontal pattern coincides with the optimal vertical pattern.

Example 2. Consider the following instance with $(L, W) = (7, 4)$ and 2 pieces to cut with dimensions (5, 2) and (2, 2) respectively (see figure 4). By considering the RU_2TDC problem, we obtain 3 pieces with dimensions (5, 2), (2, 5) and (2, 2) respectively. The optimal horizontal strips (figure 4(a)) are obtained by considering only one knapsack problem $K_{\mathcal{T},w_{r'}}$ of Theorem 1.

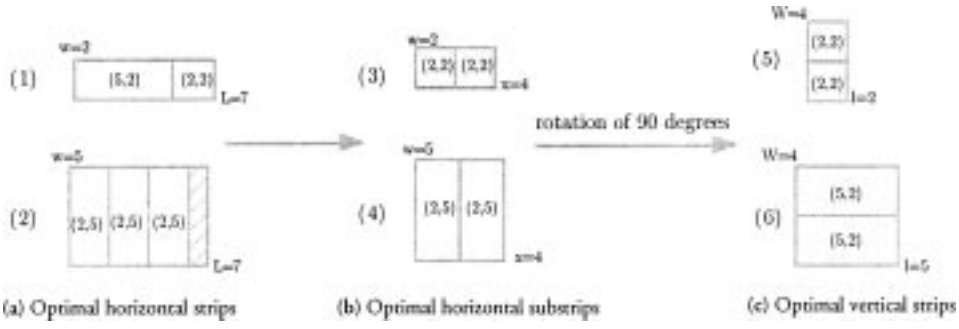

Figure 4. Representation of an instance of the RU_2TDC problem.

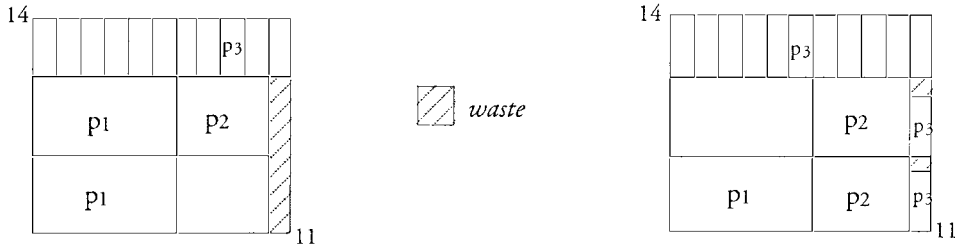

Figure 5. Two cutting patterns characterizing the FU_2TDC problem. The first solution (with value equal to 144, left side) represents a solution without trimming and the second one (with value equal to 152, right side) is a solution with trimming, where the arrangement is applied on the pieces of type p_3.

The value of the first horizontal strip (pattern (1) of figure 4(a)) is equal to 14 and the second one (pattern (2) of figure 4(a)) is equal to 30. Since dynamic programming techniques are used, it is clear that all optimal substrips are created. As shown in figure 4(b), the substrip (3) represents an optimal horizontal substrip with width $w = 2$, length $x = 4$ and value 8. Also, the substrip (4) represents an optimal substrips with dimensions (5, 4) and value 20. Now, consider the two strips of figure 4(c) ((5) and (6)) which represent a rotation of 90 degrees of the strips (3) and (4) respectively. It is easy to see that the obtained strips represent exactly the vertical ones for the RU_2TDC problem.

Therefore, by combining the optimal horizontal strips (it suffices to solve $K^{\text{hor}}_{(L,\mathcal{T})}$) we can use a slight modification in the dynamic programming procedure for obtaining also the optimal vertical pattern. Finally, the MDPP does not consider the region $x = W + 1, \ldots, L - W$ for the horizontal strips.

Figure 5 illustrates two cutting patterns. The first one (see the left side of figure 5) represents a 2-staged pattern obtained without trimming, and the second one (see the right side of figure 5) is a 2-staged pattern in which the trimming is permitted.

3. The unconstrained (un)weighted three staged cutting problem

In this section, we propose an optimal algorithm for solving the following versions of the cutting problem:

- the FU_3TDC problem in which the orientation of the pieces is not permitted and by considering the unweighted and weighted versions;
- the RU_3TDC problem in which the orientation of each piece is permitted (see figure 1(a)), and by considering both unweighted and weighted versions.

The algorithm is based principally upon depth-first search strategy characterizing the U_3TDC problem. The main idea of the method is to generate all possible guillotine cutting patterns by developing a tree where branchings represent cuts on the large rectangle or a subrectangle, and nodes represent the state of the large rectangle or the current subrectangle, after cutting has taken place. For bounding purposes we use

- an initial lower bound;
- an upper bound on the maximum value obtainable from each node;
- some established optimality criteria.

Several authors (see [6, 13, 15, 26]) have used a similar representation and called it the graph (or tree) structure. These approaches were proposed for solving exactly and approximately some variants of unconstrained and constrained TDC problems. Generally, the search process is considered as a search in a directed graph (or tree), where each (internal) node represents a subproblem and each arc represents a relationship between some nodes.

The problem is defined as a search in a graph (or tree) by specifying the initial node, the final nodes and the rules that define the allowed moves. As mentioned above, the large rectangle can be represented by the initial node and the pieces by the final nodes. However, each internal node represents a subrectangle which represents a part of the initial node. The allowed moves can be considered as the possible cuts on each (sub) rectangle and the different cuts can be applied to the large rectangle which corresponds to the degree of this node (S in figure 6). After the cut has taken place, two internal nodes are created which corresponds to two succeeding subrectangles. Other cuts can be applied to the resulting internal nodes and so on, until the given subrectangles coincide with the pieces (figure 6: B and C are generated by applying a horizontal cut on A, or D and E obtained by using a vertical cut on A). For more details, the reader is refereed to Hifi ([15]) and Morabito et al. ([26]).

In order to deal with a finite number of cuts without loss of optimality, the cuts are made at points which form linear combinations of lengths and widths. Herz ([13]) showed for the UU_TDC problem that the guillotine cuts can, without loss of generality, be integer nonnegative linear combinations of sizes of the ordered pieces. For a given (sub) rectangle (ℓ, w), the cuts can be reduced along the length L (resp. width W) to the following sets:

$$\mathcal{L} = \left\{ x \mid x = \sum_{i=1}^{n} l_i z_i \leq \ell, z_i \in \mathbb{N}, w_i \leq w \right\}$$
$$\left(\text{resp.} \quad \mathcal{W} = \left\{ y \mid y = \sum_{i=1}^{n} w_i z_i \leq w, z_i \in \mathbb{N}, l_i \leq \ell \right\} \right)$$

The sets $\mathcal{L}$ and $\mathcal{W}$ are called *discretization* (or normalized) sets and each of them can be computed by using the procedure due to Christofides and Whitlock ([6]). Herz also proved that a kind of duplication can be avoided by defining for each subrectangle (ℓ, w) the sets $\mathcal{L}_{\ell/2}$ and $\mathcal{W}_{w/2}$ respectively. This means that the set $\mathcal{L}$ (resp. $\mathcal{W}$) contains the points less or equal to half of the length ℓ (resp. the width w) of the subrectangle (in order to avoid effects of symmetry). In our algorithm, we adopt this idea by limiting the search process only to the elements of the sets $\mathcal{L}/2$ and $\mathcal{W}/2$.

3.1. The initial lower bound

The initial lower bound for both FU_3TDC and RU_3TDC problems can be considered as the optimal solution produced by the DPP approach (or MDPP for the rotated case) used

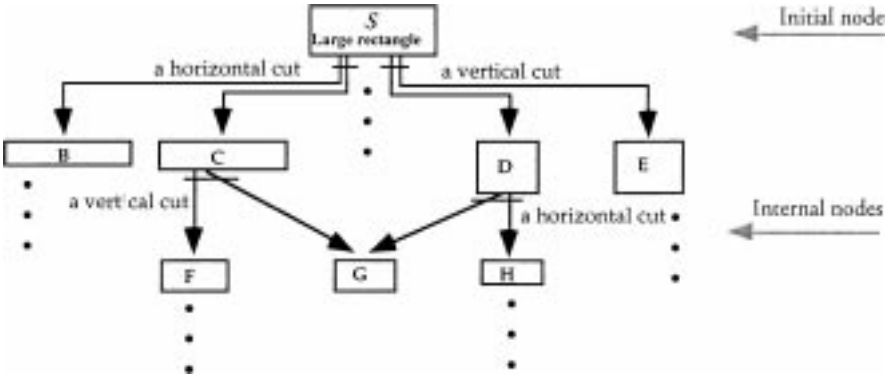

Figure 6. The search process used on the directed graph.

for the two-staged version. The maximum value between the vertical and the horizontal two-staged patterns can be considered as an initial lower bound for the U_3TDC problem. In this case a slight modification is introduced in the dynamic programming procedure. It consists in dealing only with the elements of the sets $\mathcal{L}$ and $\mathcal{W}$ when one-dimensional knapsack problems are solved.

Note that it is easily seen that the initial lower bound is also represented by the maximum between the two two-staged patterns, when the first-stage cut is made vertically (or horizontally).

3.2. An upper bound at each node

A simple upper bound can be applied to each (internal) node. This bound has been introduced in a similar tree-search (see [14, 15, 18, 26]) and computed by solving another one-dimensional knapsack problem. The knapsack problem is given as follows:

$$(K_u)\begin{cases} \mathcal{U}(\ell, w) = \max & \displaystyle\sum_{j \in R_{\ell w}} c_j z_j \\ \text{s.t.} & \displaystyle\sum_{j \in R_{\ell w}} (l_j w_j) z_j \le (\ell w) \\ & z_j \le \left\lfloor \frac{\ell}{j_i} \right\rfloor \times \left\lfloor \frac{w}{w_j} \right\rfloor, z_j \ge 0,\ j \in R_{\ell w} \end{cases}$$

where $R_{\ell w}$ is the set of pieces entering in (ℓ, w), z_j is the number of times that the piece j can be produced in $R_{\ell w}$. Of course, the problem (K_u) is a large one-dimensional knapsack one and so, we just solve the problem heuristically by considering the following representation: reorder initially the pieces in nonincreasing order of $(c_i)/(l_i w_i)$, $i = 1, \ldots, n$, and if two terms realize the same value, we first take the term with a greater value. By applying the greedy algorithm for solving (K_u), (see Pisinger [28]) we take the value $\lfloor \mathcal{U}(\ell, w) \rfloor$ as the upper bound for (ℓ, w).

3.3. A top-down exact algorithm

In this section, we use a top-down exact algorith for solving both FU_3TDC and RU_3TDC problems. The algorithm is based principally on the following results.

Theorem 2. *An optimal pattern for both FU_3TDC and RU_3TDC problems is always represented by a set of strips, and each of these strips represents an optimal horizontal or vertical* 2*-staged pattern.*

Proof: We consider only the vertical linear combinations and the proof is similar for the horizontal ones. We distinguish two cases: (i) the set of linear combinations $\mathcal{L}$ is empty and (ii) $\mathcal{L} \neq \emptyset$.

The case (i): the pattern which contains the combination of some vertical strips does not exist, since all pieces realizes $l_i > L, i = 1, \ldots, n$. So, in this case the optimal vertical pattern is represented by the value $g_L^{\text{hor}}(W)$ which represents the combination of some horizontal strips.

The case (ii): $\mathcal{L} \neq \emptyset$ means that there exist at least a linear combination $x \in \mathcal{L}$. So, there is at least a partition for which the optimal vertical 3-staged pattern realizes the following value:

$$V^{3\text{stg}}(L, W) = \max\left\{ V_{0-\text{cut}}^{2\text{stg}}(L, W), V^{3\text{stg}}(x, W) + V^{3\text{stg}}(L - x, W) \right\} \tag{2}$$

where $V_{0-\text{cut}}^{2\text{stg}}(s, t) = \max\{g_t^{\text{ver}}(s), g_s^{\text{hor}}(t)\}, s \leq L$ and $t \leq W$.

On one hand, consider that x is the smallest linear combination and so, $V^{3\text{stg}}(x, W) = V_{0-\text{cut}}^{2\text{stg}}(x, W)$ which represents a horizontal or a vertical 2-staged pattern for (x, W). On the other hand, by applying the same reasoning for the subrectangle $(L - x, W)$ (as above), the solution value can be written as follows:

$$V^{3\text{stg}}(L - x, W) = \max\left\{ V_{0-\text{cut}}^{2\text{stg}}(L - x, W), V^{3\text{stg}}(x', W) + V^{3\text{stg}}(L - x - x', W) \right\} \tag{3}$$

where x' is a linear combination such that $x' \geq x$. From Eq. (3), the optimal pattern for $(L - x, W)$ is either a 2-staged pattern with value $V_{0-\text{cut}}^{2\text{stg}}(L - x, W)$ or the value of the sum of two 3-staged patterns $V^{3\text{stg}}(x', W)$ and $V^{3\text{stg}}(L - x - x', W)$ respectively.

Remark that $V^{3\text{stg}}(x', W)$ can also be written as Eq. (2) and so, by recurrence, the optimal structure of (x', W) (resp. $(L - x - x', W)$) has either a 2-staged form or the sum of different 2-staged forms.

Hence, the optimal vertical 3-staged pattern is always composed by a set of strips and each of these strips has a 2-staged structure. □

Figure 7 illustrates the search process used on the directed graph. By following the Theorem 2, we can show that the structure of the search graph can be reduced to the first level of each created subrectangle (strip). This is possible because, on one hand, each vertical cut (see the left side of figure 7) produces two subrectangles (strips) and all horizontal

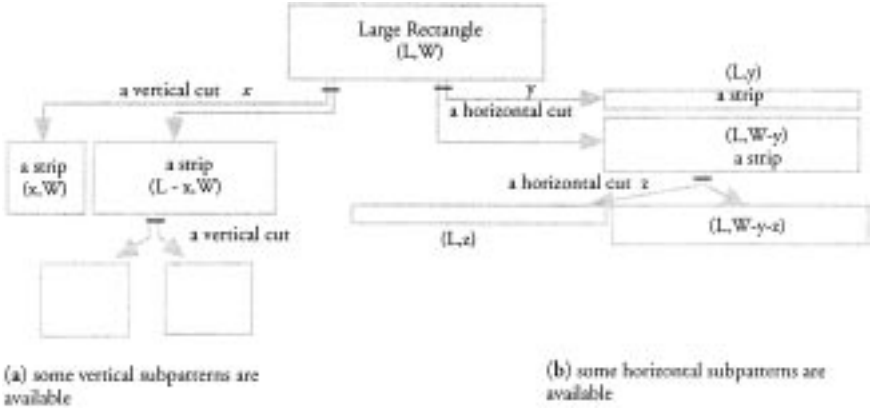

Figure 7. Representation of the search process.

cuts on the resulting subrectangles (strips) can be neglected (since dynamic programming procedures are used). On the other hand, when horizontal cuts are applied (see the right side of figure 7), all vertical cuts are also neglected on the resulting subrectangles (strips). We recall that we limit the vertical cuts to the points of $\mathcal{L}/2$ and the horizontal cuts to the points of $\mathcal{W}/2$; in Theorem 2, we have supposed that $\mathcal{L} \neq 0$ (resp. $\mathcal{W} \neq 0$) and the demonstration remains true when we consider $\mathcal{L}/2$ (resp. $\mathcal{W}/2$).

In what follows, we shall show how we can establish some optimality which permits to neglect the resolution of some knapsack problems at some internal nodes.

Proposition 1. *Let (x, W) (resp. (L, y)) be a strip where the first cut is made vertically (resp. horizontally). If $x = \ell_{\min}$, (resp. $y = w_{\min}$) or $x/2 < \ell_{\min}$ (resp. $y/2 < w_{\min}$), where $\ell_{\min}$ (resp. $w_{\min}$) denotes the smallest length (resp. width) entering in the considered strip, then its optimal solution value is available.*

Proof: Let x be a vertical cut producing the subrectangle (or the strip) (x, W). By solving initially the problem $K^{\text{ver}}_{(L,W)}$, we have necessarily all values $g^{\text{ver}}_W(t), t = 0, \ldots, L$, which means that the value $g^{\text{ver}}_W(x)$ is initially available.

On one hand, the optimal 2-staged pattern of (x, W) is available, and on the other hand $g^{\text{ver}}_W(x)$ is the value of the optimal vertical 3-staged pattern. This is possible because $g^{\text{ver}}_W(x)$ forms the best combination of the vertical strips and since $x < 2\ell_{\min}$, then necessarily the first vertical cut, say x', verifies $x' \geq x$ and so, $x' = 0$ or x.

Now, if (x, W) is a strip, necessarily there exists an optimal horizontal (sub) pattern containing some pieces $(\ell, w) \mid \ell_{\min} \leq \ell \leq x$ and $w \leq W$. So, we can distinguish the two following cases:

1. $\ell_{\min} = x \Longrightarrow g^{\text{hor}}_{\ell_{\min}}(W) = g^{\text{ver}}_W(x)$, which means that it is not necessary to solve the knapsack problem $K^{\text{hor}}_{(\ell_{\min}, W)}$.
2. $x/2 < \ell_{\min} \Longrightarrow g^{\text{hor}}_x(W) = g^{\text{ver}}_W(x)$. We can distinguish two cases:

(a) If $g_x^{\text{hor}}(W) \leq g_W^{\text{ver}}(x)$, implies that it is not necessary to solve the knapsack problem $K_{(x,W)}^{\text{hor}}$ for the strip (x, W).

(b) consider that $g_x^{\text{hor}}(W) > g_W^{\text{ver}}(x)$, which means that there exist a partition (x, z) and $(x, W - z)$ such that $z \in \mathcal{W}$ for which

$$g_x^{\text{hor}}(z) + g_x^{\text{hor}}(W - z) > g_W^{\text{ver}}(x) \tag{4}$$

where $g_z^{\text{hor}}(x)$ is the solution value when the problem $K_{(x,z)}^{\text{hor}}$ is solved for (x, z). Equation (4) contradicts the fundamental principle of the dynamic programming.

Hence, the solution value of the strip (x, W) is always represented by the value $g_W^{\text{ver}}(x)$, and the demonstration is similar when the cut is made horizontally at the point y. □

Proposition 2. *Let* (L, W) *be the dimensions of the large rectangle and consider the RU_3TDC problem. Then, some horizontal and vertical* (*sub*) *patterns have an equivalent structure.*

Proof: We distinguish two cases: (i) $L = W$ and (ii) $L \neq W$.

The case (i): it is clear that for the RU_3TDC version, the set of horizontal strips is constructed. By using Theorem 1 (see Remark 2), we can affirm that the set of vertical strips coincides with the set which represents the horizontal strips (by applying a rotation of 90 degrees on each strip). Since only a knapsack problem permits to combine the horizontal strips, then we can easily see that the vertical pattern is also equivalent to the horizontal one.

The case (ii): suppose that $L > W$.

The *horizontal cut*. By using Theorem 1, the set containing the horizontal strips and substrips is created and so, by applying a rotation on each of these (sub)strips, then we have necessarily all vertical (sub)strips.

Let y be a horizontal cut such that $y \in \mathcal{W}/2$, and (L, y) and $(L, W-y)$ be the two resulting (sub)rectangles. By applying Theorem 1, we have to solve three knapsack problems for each created subrectangle.

On one hand, consider the rectangle (L, y) and the values produced by solving (initially) the problem $K_{(L,W)}^{\text{hor}}$. It is clear that the value $g_L^{\text{hor}}(y)$ is available since all values $g_L^{\text{hor}}(t)$, $t = 0, \ldots, W$, are computed initially by solving $K_{(L,W)}^{\text{hor}}$. On the other hand, if we consider the second rectangle $(L, W - y)$, then necessarily its values $g_L^{\text{hor}}(W - y)$ is also available.

So, the algorithm needs to solve two small knapsack problems instead of six ones. The first one is the problem $K_{(L,y)}^{\text{ver}}$ by considering only the available vertical (sub)strips for which the width $w_j \leq y$, $j = 1, \ldots, r$, and the second one $(K_{(L,W-y)}^{\text{ver}})$ for which $w_j \leq W - y$, $j = 1, \ldots, r$.

The *vertical cut*. Let x be a vertical cut such that $x \in \mathcal{L}/2$, and (x, W) and $(L - x, W)$ be the two resulting (sub)rectangles. By using the same reasoning (as above), for each subrectangle we solve only one unbounded knapsack problem, because initially all values $g_t^{\text{ver}}(W)$, $t = 0, \ldots, L$, are computed and so, $g_x^{\text{ver}}(W)$ and $g_{L-x}^{\text{ver}}(W)$ are also available. So, we just combine twice some different available horizontal substrips for calculating the solution values $g_W^{\text{hor}}(x)$ and $g_W^{\text{hor}}(L - x)$ respectively.

Hence some knapsack problems do not need be solved by the proposed algorithm. □

We note that Proposition 2 can also be applied to the upper bounds. Indeed, let (x, y) denote an internal rectangle and consider that $\mathcal{U}(x, y)$ is available when a x (or y) cut is made. Then, the upper bound of another rectangle (y, x) can be represented exactly by the upper bound $\mathcal{U}(x, y)$. This is possible because (y, x) represent a simple rotation of the rectangle (x, y).

4. Computational results and future works

All approaches (the proposed algorithms, Gilmore and Gomory's procedure and Beasley's one) were coded in `C` and tested on a `DATA General AV8000 (Quadri-Processor Motorola 88100, 100Mhz)` with CPU time limited to three hours. This section is divided into two parts. First, we present empirical evidence of the performance of the proposed exact algorithms. In the second part we discuss some perspectives of the proposed approaches.

4.1. Computational results

To our knowledge, there exist no publicly available standard U_2TDC and U_3TDC test problems. Therefore, in order to better test the effectiveness of the different exact algorithms, we have considered 53 problem instances of different sizes. To aid further development of algorithms (exact and approximate) for the cutting problem, we have made these test problems publicly available (`ftp://panoramix.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting`).

The computational results are shown in Table 2 for the RU_2TDC problem and in Tables 3 and 4 for the FU_3TDC and RU_3TDC problems.

A. Some details about the instances. For the unweighted version (i.e., each c_i is equal to $l_i w_i$), 27 problem instances are considered and the other 26 problem instances represent the weighted version. Some of these instances are taken from the literature and the other ones are randomly generated or constructed from the existing ones. These instances are taken as follows (see Table 1):

For the *unweighted version* (see column 1 of Table 1), we have considered the following 27 instances: `H` is an instance given in [13], `HZ1` in [14, 18], `M1`–`M5` in [26], `U1`–`U3` in [15], `U4` is constructed as follows: the dimensions of the large rectangle (L, W) are equal to $(\max\{L_{U_i}\}, \max\{W_{U_i}\})$, $i = 1, \ldots, 3$, where L_{U_1}, L_{U_2} and L_{U_3} (resp. W_{U_1}, W_{U_2} and W_{U_3}) denote respectively the length (resp. width) of the instances `U1`–`U3`. Also, `UU1`–`UU11` are given in [8], `B` is the instance used by Beasley ([3]) and `LU1`–`LU4` are four large-size instances generated as follows: L (resp. W) is uniformly taken in the interval [20000, 50000], the number of pieces to cut in the interval [100, 200] and the dimensions of the pieces to cut in the interval $[0.01\,\gamma, 0.75\,\gamma]$ (where $\gamma = L$ or W).

For the *weighted version* (see column 2 of Table 1), 26 problem instances are considered. `HZ2` is given in [17], `MW1`–`MW5` represent the instances `M1`–`M5` but each piece has a weight c_j, $j = 1, \ldots, n$, taken in the interval $[0.1 l_j w_j, 0.4 l_j w_j]$. `W1`–`W3` are given in [15], `W4` is

Table 1. Test problem details.

	The different problem instances						
	Unweighted				Weighted		
Inst	*L*	*W*	*n*	Instd	*L*	*W*	*n*
H	127	98	5				
HZ1	78	67	6	HZ2	90	80	5
M1	100	156	10	MW1	100	156	10
M2	253	294	10	MW2	253	294	10
M3	318	473	10	MW3	318	473	10
M4	501	556	10	MW4	501	556	10
M5	750	806	10	MW5	750	806	10
U1	4500	5000	10	W1	5000	5000	20
U2	5050	4070	10	W2	3427	2769	20
U3	7350	6579	20	W3	7500	7381	40
U4	7350	6579	40	W4	7500	7381	80
UU1	500	500	25	UW1	500	500	25
UU2	750	800	30	UW2	560	750	35
UU3	1100	1000	25	UW3	700	650	35
UU4	1000	1200	38	UW4	1245	1015	45
UU5	1450	1300	50	UW5	1100	1450	35
UU6	2050	1457	38	UW6	1750	1542	47
UU7	1465	2024	50	UW7	2250	1875	50
UU8	2000	2000	55	UW8	2645	2763	55
UU9	2500	2460	60	UW9	3000	3250	45
UU10	3500	3450	55	UW10	3500	3650	60
UU11	3500	3765	25	UW11	555	632	25
B	3000	3000	32	BW	3000	3000	32
LU1	20789	23681	100	LW1	20789	23681	100
LU2	25587	34563	100	LW2	25587	34563	100
LU3	37587	27563	150	LW3	37587	27563	150
LU4	45237	35983	200	LW4	45237	35983	200

the fusion of the three instances W1–W3 (as for U4), UW1–UW11 are taken in [8], BW is the Beasley's instance for which the weights have been generated as previously and the other instances LW1–LW4 represent exactly the instances LU1–LU4 but each piece to cut has a weight generated in the same interval as for MW1–MW5.

B. The RUU_2TDC and RUW_2TDC problems. The results of these instances are shown in Table 2. For each instance, we report its optimal horizontal (Opt. Hor.) and optimal vertical (Opt. Vert.) patterns, the first time (T_{DPP}) which is the time that the DPP takes in

Table 2. Performance of the MDPP on 53 problem instances representing the RU_2TDC problem.

	The RUU_2TDC and RUW_2TDC problems							
	The trimming is not permitted: *NTr*				The trimming is permitted: *Tr*			
Inst	Opt. Hor.	Opt. Vert.	T_{DPP}	T_{MDPP}	Opt. Hor.	Opt. Vert.	T_{DPP}	T_{MDPP}
H	11988[a]	11631	<0.01	<0.01	12132[a]	11904	<0.01	<0.01
HZ1	5226[a]	5148	<0.01	<0.01	5226[a]	5148	<0.01	<0.01
M1	15424[a]	15156	0.08	0.07	15424	15468[a]	0.08	0.07
M2	72172[a]	72130	0.19	0.18	72172[a]	72130	0.27	0.20
M3	141922	143074[a]	0.32	0.29	146995[a]	145561	0.38	0.31
M4	269928	272927[a]	0.48	0.32	270779	273991[a]	0.49	0.32
M5	577882[a]	567999	0.67	0.40	577882[a]	577882[a]	0.77	0.49
B	8997780[a]	8997780[a]	8.85	6.41	8997780[a]	8997780[a]	9.43	7.03
HZ2	7864	8145[a]	<0.01	<0.01	8196[a]	8145	<0.01	<0.01
MW1	3882[a]	3882[a]	0.10	0.07	3882	3916[a]	0.09	0.07
MW2	24950[a]	24950[a]	0.20	0.18	24950[a]	24950[a]	0.33	0.22
MW3	35486	37835[a]	0.32	0.26	36595	39637[a]	0.41	0.32
MW4	59382	64044[a]	0.43	0.29	59382	64044[a]	0.53	0.31
MW5	189924[a]	189924[a]	0.73	0.60	189924[a]	189924[a]	0.77	0.63
BW	2349861[a]	2349861[a]	12.53	7.35	2352288[a]	2352288[a]	12.93	8.21
U1	22338048[a]	22203764	7.75	5.50	22338048[a]	22238948	8.45	6.10
U2	20204757[a]	20001327	5.37	3.91	20355161[a]	20267132	5.85	4.31
U3	47961789	47977258[a]	17.76	12.71	48144064[a]	48000756	19.20	13.70
U4	48264744	48350130[a]	42.85	30.60	48316432	48350130[a]	45.95	32.56
W1	162279[a]	162279[a]	12.94	8.91	162279[a]	162279[a]	13.15	8.91
W2	36960	36972[a]	6.14	4.50	36972	37621[a]	6.57	4.59
W3	245256	250830[a]	33.00	22.51	245256	250830[a]	34.24	23.08
W4	377910[a]	377910[a]	82.65	56.00	377910[a]	377910[a]	83.67	57.29
UU1	239076[a]	239076[a]	1.02	0.70	246046[a]	246046[a]	1.14	0.80
UU2	595288[a]	578283	2.02	1.31	593256	595288[a]	2.12	1.47
UU3	1051808	1054746[a]	2.22	1.50	1082504[a]	1076640	2.43	1.61
UU4	1175431	1179896[a]	3.64	2.48	1187353	1188638[a]	3.83	2.79
UU5	1833168	1868985[a]	5.74	4.00	1868739	1878253[a]	6.24	4.40
UU6	2944368[a]	2944368[a]	5.74	4.22	2944368	2951202[a]	6.14	4.60
UU7	2925848[a]	2914086	7.64	5.67	2935834[a]	2935277	8.34	6.07
UU8	3924280[a]	3924280[a]	9.24	6.00	3959352[a]	3959352[a]	10.00	6.6
UU9	6061968	6100692[a]	12.44	8.20	6100692	6108427[a]	13.52	8.91
UU10	11674068[a]	11674068[a]	16.38	10.87	11929575	11973240[a]	17.74	12.00
UU11	13103206	13107194[a]	14.45	10.32	13104891	13110755[a]	15.74	11.07

(*Continued on next page*.)

Table 2. (*Continued*).

	The RUU_2TDC and RUW_2TDC problems							
	The trimming is not permitted: *NTr*				The trimming is permitted: *Tr*			
Inst	Opt. Hor.	Opt. Vert.	T_{DPP}	T_{MDPP}	Opt. Hor.	Opt. Vert.	T_{DPP}	T_{MDPP}
UW1	6539[a]	6539[a]	1.03	0.67	6539[a]	6539[a]	1.13	0.70
UW2	9348	8384[a]	2.04	1.47	9348[a]	8784	2.14	1.51
UW3	5888	6500[a]	2.04	1.37	5888	6500[a]	2.13	1.40
UW4	7748[a]	7620	4.44	3.09	7748[a]	7689	4.62	3.30
UW5	7986[a]	7887	3.82	2.78	8398[a]	8398[a]	4.04	2.94
UW6	6894[a]	6712	6.54	4.70	6894[a]	6712	6.93	4.71
UW7	11574[a]	10464	8.93	6.30	11585[a]	10464	9.43	6.70
UW8	7692[a]	7692[a]	12.54	8.52	8088[a]	8088[a]	13.34	8.89
UW9	7527[a]	6948	11.83	8.07	7527[a]	7038	12.44	8.53
UW10	8172[a]	8172[a]	18.12	12.43	8172[a]	8172[a]	19.24	13.10
UW11	17500[a]	16800	1.54	1.07	17500	17700[a]	1.55	1.09
LU1	492259840[a]	492259840[a]	928.87	662.73	492259840[a]	492259840[a]	948.74	704.24
LU2	884303332[a]	884303332[a]	1693.84	1286.31	884318062	884329158[a]	1725.64	1336.07
LU3	1035945332[a]	1035945332[a]	4030.84	3110.35	1035971524[a]	1035962256	4165.44	3251.85
LU4	1627681752[a]	1627681752[a]	7749.94	5878.67	1627681752[a]	1627681752[a]	8063.64	6170.27
LW1	170797032[a]	169995168	948.24	698.86	171079321[a]	170962230	913.46	686.61
LW2	324323665	325111781[a]	1714.94	1309.55	325168131	325725070[a]	1630.24	1281.06
LW3	431362980	433587165[a]	4055.65	3118.57	432052029	433859655[a]	3917.38	3094.57
LW4	567915051[a]	565325676	7834.89	5924.83	568436545[a]	566912535	7619.15	6016.90

[a]represents the optimal solution when the first-stage cut is unspecified.

order to reach the optimal solution and the execution time (T_{MDPP}) which is the time that the MDPP takes to give the optimal solution.

As shown in Table 2, we have considered the no-trimming (*NTr*) and the trimming (*Tr*) cases. Examining Table 2, we observe that:

1. Of the 53 considered instances, the optimal vertical solution is realized for 18 instances for the *NTr* case, and 20 instances for the *Tr* case.
2. We can see that the MDPP performs better than the standard version DPP. The computational time is in average equal to 425 seconds for the MDPP algorithm and equal to 564 seconds when the DPP is applied.
3. An important point is that the average time gain is (generally) increasing with instances size. Note that for all treated instances, the average computational time gain is of 22% (resp. 24%) for the *NTr* (resp. *Tr*) case, which represents globally a percentage of 23.24%. Figure 8 illustrates the variation of the percentage gain of the treated instances.

Table 3. Performance of the TDE algorithm compared to the BE approach.

	The trimming is not permitted: *NTr*							
	The FU-3TDC version				The RU-3TDC version			
Inst.	Opt. Hor.	Opt. Vert.	T_{BE}	T_{TDE}	Opt. Hor.	Opt. Vert.	T_{BE}	T_{TDE}
H	12132[a]	12192[a]	10.65	0.27	12348[a]	12240	15.72	0.88
HZ1	5226[a]	5226[a]	4.35	<0.01	5226[a]	5226[a]	5.54	0.10
M1	15024[a]	15024[a]	14.82	0.47	15424	15468[a]	18.72	1.70
M2	72564[a]	72263	144.76	2.05	73080[a]	72774	180.45	7.56
M3	140717	142817[a]	348.45	1.86	146995	147054[a]	404.05	5.80
M4	265768[a]	265768[a]	852.00	3.57	273213	273991[a]	1047.71	9.07
M5	577882[a]	577882[a]	2587.15	5.62	577930[a]	577882	3135.10	21.32
B	8997780[a]	8997780[a]	[b]	1745.27	9000000[a]	9000000[a]	[b]	87.10
U1	22300011	22351950[a]	[d]	1924.08	22397400[a]	22362510	[d]	7693.57
U2	20089060	20194715[a]	[d]	338.40	20355161[a]	20351821	[d]	1652.90
U3	48095058[a]	47951344	[d]	4624.90	48239155[a]	48165659	[d]	[b]
UU1	241260[a]	238121	793.66	8.05	246046[a]	246046[a]	965.70	18.07
UU2	595288[a]	595288[a]	3069.65	12.66	595288[a]	595288[a]	3629.30	49.81
UU3	1059674	1065051[a]	6918.40	22.07	1087112[a]	1083712	8268.00	54.49
UU4	1177371[a]	1170598	8860.80	73.05	1187630	1188638[a]	10277.00	179.88
UU5	1868985[a]	1868985[a]	[b]	106.67	1868985	1878253[a]	[b]	459.47
UU6	2944368	2950760[a]	[b]	43.67	2944368	2951202[a]	[b]	280.08
UU7	2924000	2925362[a]	[b]	301.37	2949043[a]	2947961	[b]	950.78
UU8	3959352[a]	3959171	[b]	116.07	3963854[a]	3963854[a]	[b]	576.24
UU9	6100692[a]	6100692[a]	[b]	158.68	6103960	6108427[a]	[b]	790.70
UU10	11930414[a]	11915920	[b]	641.00	11979337	11996982[a]	[b]	2511.20
UU11	13146050[a]	13141175	[b]	4489.00	13161640[a]	13158873	[b]	8870.60
HZ2	8046[a]	8046[a]	5.33	0.17	8238	8298[a]	7.45	0.77
MW1	3882[a]	3882[a]	14.75	0.37	3882	3916[a]	18.70	1.78
MW2	24950[a]	24950[a]	145.85	1.97	24950[a]	24950[a]	181.55	8.57
MW3	35486	37068[a]	350.20	1.66	37835	39637[a]	404.93	6.18
MW4	59576[a]	59576[a]	852.35	3.66	64044[a]	64044[a]	1046.35	7.88
MW5	189924[a]	189924[a]	2582.60	4.65	189924[a]	189924[a]	3108.55	18.62
BW	2307817[a]	2296271	[b]	2705.10	2370620[a]	2370620[a]	[b]	6477.50
W1	161424	163430[a]	[d]	3933.54	167751[a]	167751[a]	[d]	8280.80
W2	34520	36972[a]	[d]	281.17	36972	37621[a]	[d]	1427.35
W3	234108	250830[a]	[d]	10623.45	253617[a]	253617[a]	[d]	[c]
UW1	6036[a]	6036[a]	836.36	13.37	6696[a]	6696[a]	995.21	29.17
UW2	8468[a]	8468[a]	1987.80	54.47	9348	9732[a]	2328.25	122.90

(*Continued on next page*.)

Table 3. (*Continued*).

	The trimming is not permitted: *NTr*							
	The FU-3TDC version				The RU-3TDC version			
Inst.	Opt. Hor.	Opt. Vert.	T_{BE}	T_{TDE}	Opt. Hor.	Opt. Vert.	T_{BE}	T_{TDE}
UW3	5964	6226[a]	2079.80	32.06	6576	7188[a]	2520.57	111.31
UW4	7900	8326[a]	4127.82	135.06	8452[a]	8450	4851.44	322.57
UW5	7780[a]	7780[a]	5514.20	58.76	8398[a]	8398[a]	6055.10	206.90
UW6	6615[a]	6615[a]	[b]	194.76	6937[a]	6894	[b]	599.75
UW7	10464[a]	10464[a]	[b]	305.86	11585[a]	11585[a]	[b]	1099.57
UW8	7692[a]	7692[a]	[b]	453.36	8088[a]	8088[a]	[b]	1215.76
UW9	7038[a]	7038[a]	[b]	265.34	7527[a]	7527[a]	[b]	1575.47
UW10	7507[a]	7461	[b]	977.88	8172[a]	8172[a]	[b]	1286.20
UW11	15747[a]	15747[a]	1850.62	84.27	18200[a]	18200[a]	2110.54	158.54

[a]

[b]means that the approach is not able to produce the optimal solution after three hours.
[c]means that the TDE algorithm takes more than three hours for giving the optimal solution for the RU_3TDC version.
[d]means that the approach can not support the capacity of this instance.

C. The FU_3TDC and RU_3TDC problems. In this section, we compare the performance of the Top-Down Exact algorithm (shortly TDE) to Beasley's Exact one (shortly BE) (Beasley [3]). This comparison is given in Table 3 when trimming is not permitted, and given in Table 4 when trimming is considered. We not that only instances solved within CPU time under three hours are represented (except for U3 and W3 when the orientation of the pieces is permitted, because for these instances the TDE algorithm produces the optimal solutions under three hours for both FUU_3TDC and FUW_3TDC versions).

For each instance, we report its optimal horizontal (Opt. Hor.) and optimal vertical (Opt. Vert.) patterns, the first time (T_{BE}) which is the time that the BE algorithm takes to give the optimal solution and the execution time (T_{TDE}) which is the time that the TDE algorithm takes to produce the optimal solution. As shown in Tables 3 and 4, we have considered the no-trimming (*NTr*, Table 3) and the trimming (*Tr*, Table 4) cases.

Examining Tables 3 and 4, we observe that:

1. The TDE algorithm performs better than the BE algorithm.
2. The computational time gain (when the BE algorithm resolves the instances with the imposed CPU time) represents, for the *NTr* case, at least 89.66% (HZ2) and it is very important for some instances (96.58% (UW5), 99.32% (M5), 99.34% (UU3) and 99.40% (MW5); for the *Tr* case, the same phenomenon is produced, i.e. at least 79.12% (HZ2) and for the other ones, 96.77% (UW5), 98.59% (MW5), 98.74% (M5) and 98.76% (UU3).

   The average percentage gain is equal to 97.39% for the *NTr* case, and to 95.14% for the *Tr* case.

Table 4. Comparison of the TDE algorithm to the BE approach on a set of problem instances of Table 1.

	The trimming is permitted: *Tr*							
	The FU_3TDC version				The RU_3TDC version			
Inst.	Opt. Hor.	Otp. Vert.	T_{BE}	T_{TDE}	Opt. Hor.	Opt. Vert.	TBE	TTDE
H	12132	12192[a]	5.39	0.27	12348[a]	12240	7.44	0.72
HZ1	5226[a]	5226[a]	2.32	<0.01	5226[a]	5226[a]	3.21	0.05
M1	15024[a]	15024[a]	7.62	0.58	15468[a]	15468[a]	9.43	1.76
M2	72564[a]	72263	64.62	2.07	73080[a]	72774	77.08	7.46
M3	142735	142817[a]	167.82	1.56	146995	147054[a]	183.93	5.66
M4	265768[a]	265768[a]	404.67	3.67	273991[a]	273991[a]	441.78	9.00
M5	577882[a]	577882[a]	1242.65	5.58	581797	586879[a]	1321.3	16.60
B	8997780[a]	8997780[a]	[b]	1748.86	9000000[a]	9000000[a]	[b]	87.82
U1	22300011	22351950[a]	[d]	1992.84	22364970	22416630[a]	[d]	5328.70
U2	20118655	20194715[a]	[d]	338.25	20361791	20382215[a]	[d]	1653.00
U3	48095058[a]	48074355	[d]	4727.30	48239155[a]	48203000	[b]	[c]
UU1	241260[a]	240346	371.72	8.06	246046[a]	246046[a]	433.65	17.27
UU2	595288[a]	595288[a]	1335.87	12.87	595288	595655[a]	1530.00	49.00
UU3	1065051	1072764[a]	3129.10	18.87	1089308[a]	1083712	3437.20	42.46
UU4	1177371	1178295[a]	3720.30	67.00	1188638[a]	1188638[a]	4228.95	186.47
UU5	1868985[a]	1868985[a]	[b]	105.87	1878253[a]	1878253[a]	[b]	352.17
UU6	2950760[a]	2950760[a]	[b]	43.00	2951202[a]	2951202[a]	[b]	262.86
UU7	2930654[a]	2930654[a]	[b]	255.17	2949043[a]	2947961	[b]	942.38
UU8	3959352[a]	3959352[a]	[b]	167.87	3974828[a]	3974828[a]	[b]	444.47
UU9	6100692[a]	6100692[a]	[b]	159.68	6110442[a]	6108427	[b]	739.71
UU10	11955852[a]	11945134	[b]	519.57	11981182	11996982[a]	[b]	2239.16
UU11	13146050[a]	13141175	[b]	4552.87	13161640[a]	13158873	[b]	8993.00
HZ2	8046[a]	8046[a]	2.87	0.08	8238	8298[a]	3.64	0.76
MW1	3882[a]	3882[a]	7.66	0.39	3916[a]	3916[a]	9.34	1.87
MW2	24950[a]	24950[a]	64.53	2.07	24950[a]	24950[a]	77.54	8.57
MW3	37068[a]	37068[a]	168.73	1.68	39637[a]	39637[a]	185.00	5.88
MW4	59576[a]	59576[a]	403.83	3.77	64044[a]	64044[a]	439.93	8.00
MW5	189924[a]	189924[a]	1237.44	4.47	189924	190937[a]	1318.04	18.60
BW	2307817[a]	2307817[a]	[b]	2730.80	2370620[a]	2370620[a]	[b]	6873.10
W1	161424	165528[a]	[b]	3948.57	168289[a]	168289[a]	[b]	8412.40
W2	34520	36972[a]	[b]	285.10	37621[a]	37621[a]	[b]	1415.45
W3	234108	250830[a]	[b]	10766.50	253617[a]	253617[a]	[b]	[c]
UW1	6036[a]	6036[a]	376.41	13.47	6696[a]	6696[a]	436.57	29.26
UW2	8468[a]	8468[a]	840.23	53.82	9732[a]	9732[a]	993.76	123.57
UW3	5964	6226[a]	914.43	32.00	6576	7188[a]	1082.00	111.40

(*Continued on next page*.)

Table 4. (*Continued*).

	The trimming is permitted: *Tr*							
		The FU-3TDC version				The RU-3TDC version		
Inst.	Opt. Hor.	Otp. Vert.	T_{BE}	T_{TDE}	Opt. Hor.	Opt. Vert.	TBE	TTDE
UW4	7900	8326[a]	4148.14	135.68	8452[a]	8450	4870.35	381.46
UW5	7780[a]	7780[a]	5533.90	58.44	8398[a]	8398[a]	6082.15	196.20
UW6	6615[a]	6615[a]	[b]	176.66	6937[a]	6937[a]	[b]	634.57
UW7	10464[a]	10464[a]	[b]	300.78	11585[a]	11585[a]	[b]	1104.37
UW8	7692[a]	7692[a]	[b]	452.27	8088[a]	8088[a]	[b]	920.70
UW9	7038[a]	7038[a]	[b]	258.38	7527[a]	7527[a]	[b]	1570.38
UW10	7507[a]	7461	[b]	932.47	8172[a]	8172[a]	[b]	1282.50
UW11	15747[a]	15747[a]	714.32	83.68	18200[a]	18200[a]	851.53	138.37

[a]

[b]means that the approach is not able to produce the optimal solution after three hours.

[c]means that the TDE algorithm takes more than three hours for giving the optimal solution for the R_3TDC version.

[d]means that the approach can not support the capacity of this instance.

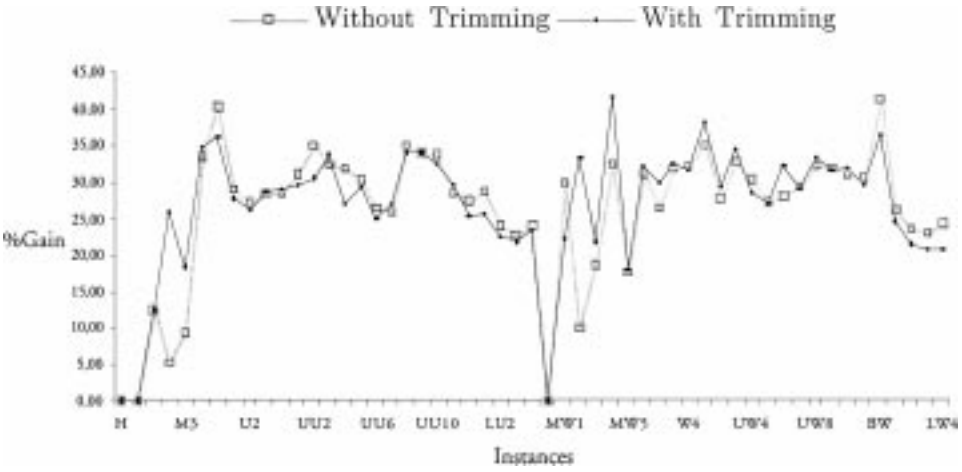

Figure 8. Variation of the percentage gain for the RU_2TDC problem (with and without trimming).

3. Figure 9 shows the behaviour of the TDE algorithm for both FU-3TDC and RU-3TDC problems when trimming is not permitted. We remark that, generally, the computational time is more important for the rotated version (except for the instance B). This can be explained by the fact that the rotated version considers more pieces to cut and so, more linear horizontal and vertical combinations are considered.
4. Also, figure 10 illustrates the behaviour of the TDE algorithm for the RU_3TDC problem, when both trimming and no-trimming versions are considered. We can remark that,

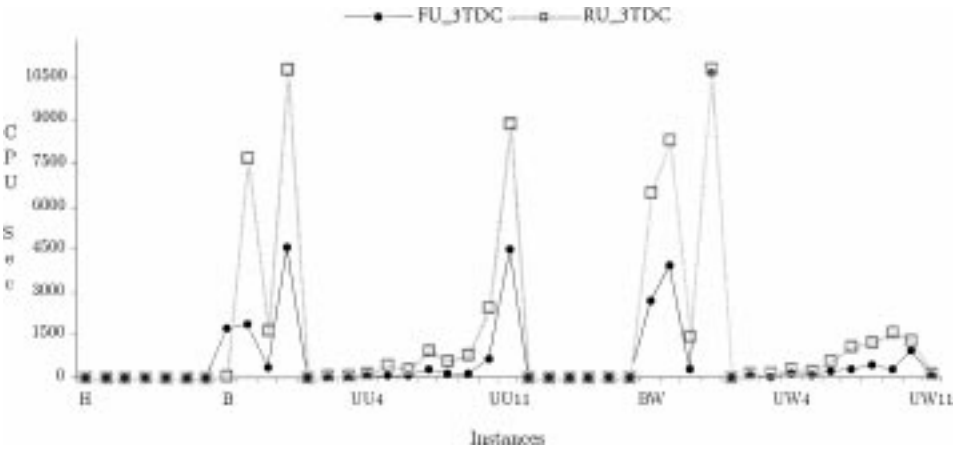

Figure 9. The behaviour of the TDE algorithm on both FU_3TDC and RU_3TDC problem instances (without trimming).

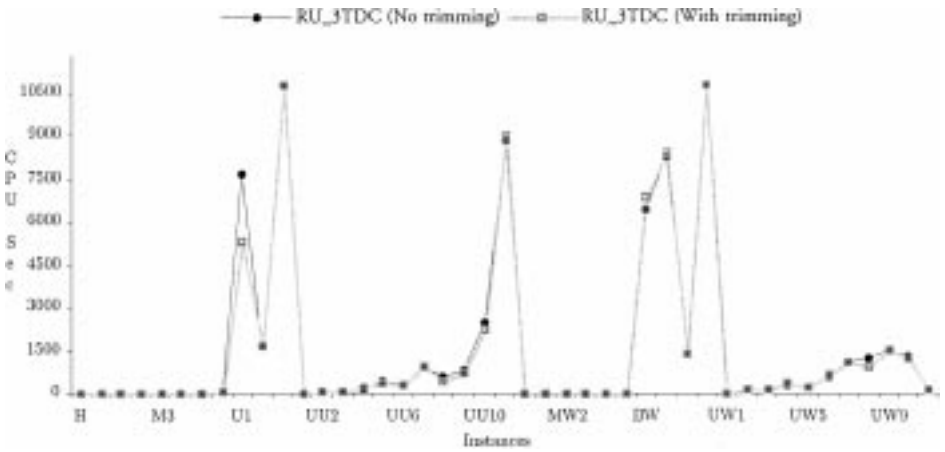

Figure 10. Variation of the computational time of the TDE algorithm for the RU_3TDC problem (with and without trimming).

generally, the computational times are almost equivalent (except for the instance `U1`: the gap between the computational times is more important). This can be explained by the effectiveness of the initial lower bound (the optimal two-staged pattern), the good internal lower bounds and the limits imposed on each internal node (the optimality criteria).

4.2. Future works

In reality, the efficient and optimal solutions of some medium (and large) combinatorial optimization problems are highly important for many applications in the field of science and

engineering. Using today's sequential algorithmic approaches, it is sometimes not possible to solve some of these instances. Indeed, the problem that seems to be interesting consists in conceiving parallel algorithms able to solve approximately and exactly the large-scale k-staged and non-staged TDC problems. The use of parallel approaches may increase the problem size which can be solved efficiently. We think that there are two important directions of research.

- Since dynamic programming techniques are used for solving a series of one-dimensional knapsack problems, so the first direction is to apply the parallel approach proposed by Andonov et al. [1, 2] and Chen et al. [4, 5], who proposed a pipeline architecture containing a linear array of q processors (where $q \leq n$ the number of pieces to cut), and queue and memory modules of size T (where $T = L$ or $T = W$ when we treat a rectangle (L, W)), for solving the one-dimensional knapsack problem.
- Another interesting direction of research is to compare the sequential TDE algorithm proposed in this paper, using parallel machines. For example, consider that processors are simultaneously attributed to each horizontal (resp. vertical) cut of the set $\mathcal{L}$ (resp. W) and each of them uses the same solution. If one of the processors finds a better solution, then the current solution is communicated to all processors. In this way, load distribution to all processors can be achieved while the high quality lower and upper bounds preserve from going deep down in the directed graph. We think that such a parallelization may solve some large-scale instances and may lead to a fast convergence.

5. Conclusion

In this paper we have considered unconstrained (un)weighted two and three staged cutting problems. We have proposed two exact algorithms. The first algorithm is an extended version of Gilmore and Gomory's procedure used for solving efficiently the rotated unconstrained (un)weighted two-staged cutting problem. The second algorithm is conceived for solving both fixed and rotated unconstrained (un)weighted three-staged cutting problems. This algorithm is based upon a graph search combined with dynamic programming procedures. Computational studies show that the two algorithms, when tested on several (medium and large size) problem instances of the literature and some generated ones, are able to give optimal solutions within reasonable computational time.

Acknowledgment

Many thanks to anonymous referees for their helpful comments and suggestions contributing to improving the presentation of the paper.

References

1. R. Andonov, F. Raimbault, and P. Quinton, "Dynamic programming parallel implementation for the knapsack problem," Technical Report, PI-740, IRISA, Campus de Banlieu, Rennes, France, 1993.

2. R. Andonov and S. Rajopadhye, “Optimal orthogonal tiling of 2-D iterations,” Journal of Parallel and Distributed Computing, vol. 45, pp. 159–165, 1997.
3. J.E. Beasley, “Algorithms for unconstrained two-dimensional guillotine cutting,” Journal of the Operational Research Society, vol. 36, pp. 297–306, 1985.
4. G.H. Chen, M.S. Chern, and J.H. Jang, “Pipeline architectures for dynamic programming algorithms,” Parallel Computing, vol. 13, pp. 111–117, 1990.
5. G.H. Chen, M.S. Chern, and J.H. Jang, “An improved parallel algorithm for 0/1 knapsack problem,” Parallel Computing, vol. 18, pp. 811–821, 1992.
6. N.Christofides and C. Whitlock, “An algorithm for two-dimensional cutting problems,” Operations Research, vol. 2, pp. 31–44, 1977.
7. H. Dyckhoff, “A typology of cutting and packing problems,” European Journal of Operational Research, vol. 44, pp. 145–159, 1990.
8. 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.
9. D. Fayard and V. Zissimopoulos, “An approximation algorithm for solving unconstrained two-dimensional knapsack problems,” European Journal of Operational Research, vol. 84, pp. 618–632, 1995.
10. 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.
11. P.C. Gilmore and R.E. Gomory, “Multistage cutting problems of two and more dimensions,” Operations Research, vol. 13, pp. 94–119, 1965.
12. P.C. Gilmore and R.E. Gomory, “The theory and computation of knapsack functions,” Operations Research, vol. 14, pp. 1045–1074, 1966.
13. J. Herz, “A recursive computing procedure for two-dimensional stock cutting,” IBM Journal of Research and Development, vol. 16, pp. 462–469, 1972.
14. M. Hifi, “Study of some combinatorial optimization problems: Cutting stock, packing and set covering problems,” PhD Thesis. Université Paris 1 Panthéon-Sorbonne, CERMSEM, 1994.
15. M. Hifi, “The DH/KD algorithm: A hybrid approach for unconstrained two-dimensional cutting problems,” European Journal of Operational Research, vol. 97, pp. 41–52, 1997.
16. 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.
17. M. Hifi and V. Zissimopoulos, “Une amélioration de l’algorithme récursif de Herz pour le problème de découpe à deux dimensions,” RAIRO, Operations Research, vol. 30, pp. 111–126, 1996.
18. M. Hifi and V. Zissimopoulos, “A recursive exact algorithm for weighted two-dimensional cutting,” European Journal of Operational Research, vol. 91, pp. 553–564, 1996.
19. 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. 48, pp. 324–331, 1997.
20. L.K. Kantorovich, “Mathematical methods of organizing and planning production,” Management Science, vol. 6, pp. 363–422, 1960.
21. E.L. Lawler, “Fast approximation algorithms for knapsack problems,” Mathematics of Operations Research,” vol. 4, pp. 339–356, 1979.
22. S. Martello and P. Toth, “An exact algorithm for large unbounded knapsack problems,” Operations Research Letters, vol. 9, pp. 15–20, 1990.
23. S. Martello and P. Toth, “Upper bounds and algorithms for hard 0–1 knapsack problems,” Operations Research, vol. 45, pp. 768–778, 1997.
24. R. Morabito and M. Arenales, “Performance of two heuristics for solving large scale two-dimensional guillotine cutting problems,” IFOR, vol. 33, pp. 145–155, 1995.
25. 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, pp. 548–560, 1996.
26. R. Morabito, M. Arenales, and V. Arcaro, “An and—or-graph approach for two-dimensional cutting problems,” European Journal of Operational Research, vol. 58, pp. 263–271, 1992.
27. 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.

28. D. Pisinger, “A minimal algorithm for the 0–1 knapsack problem,” Operations Research, vol. 45, pp. 758–767, 1997.
29. 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.
30. M. Syslo, N. Deo, and J. Kowalik, Discrete Optimization Algorithms, Prentice-Hall: New Jersey, 1983.
31. V. Zissimopoulos, “Heuristic methods for solving (un)constrained two-dimensional cutting stock problems,” Methods of Operations Research, vol. 49, pp. 345–357, 1984.