OPERATIONS RESEARCH
Vol. 53, No. 1, January–February 2005, pp. 140–150
ISSN 0030-364X | EISSN 1526-5463 | 05 | 5301 | 0140

informs

DOI 10.1287/opre.1040.0154

An Exact Algorithm for Constrained Two-Dimensional Two-Staged Cutting Problems

Mhand Hifi
LaRIA, Laboratoire de Recherche en Informatique d'Amiens, 5 rue du Moulin Neuf, 80000 Amiens, France;
CERMSEM-CNRS UMR 8095, Maison des Sciences Economiques, Université Paris 1 Panthéon-Sorbonne, France; and
LRIA-LPBD, EA 3387, EPHE, Paris, France, hifi@univ-paris1.fr

Rym M'Hallah
Department of Statistics and Operations Research, Kuwait University, P.O. Box 5969,
Safat 13060, State of Kuwait, mhallah@kuc01.kuniv.edu.kw

The *constrained two-dimensional cutting* (C_TDC) problem consists of determining a *cutting pattern* of a set of n small rectangular piece types on a rectangular stock plate S with length L and width W, to maximize the sum of the profits of the pieces to be cut. Each piece type i, $i = 1, \ldots, n$, is characterized by a length l_i, a width w_i, a profit (or weight) c_i, and an upper demand value b_i. The upper demand value is the maximum number of pieces of type i that can be cut on S. In this paper, we study the two-staged C_TDC problem, noted C_2TDC. It is a classical variant of the C_TDC where each piece is produced, in the final cutting pattern, by at most two cuts. We solve the C_2TDC problem using an exact algorithm that is mainly based on a bottom-up strategy. We introduce new lower and upper bounds and propose new strategies that eliminate several duplicate patterns. We evaluate the performance of the proposed exact algorithm on problem instances extracted from the literature and compare it to the performance of an existing exact algorithm.

Subject classifications: dynamic programming; industries; programming: algorithms—branch-and-bound; simulation: applications, efficiency.
Area of review: Optimization.
History: Received August 2002; revisions received April 2003, October 2003; accepted November 2003.

1. Introduction

Many operations research and artificial intelligence applications in the fields of science and engineering are complex combinatorial optimization problems (Garey and Johnson 1979, Kantorovich 1960). Cutting and packing problems are a special case of these combinatorial problems. They belong to an old and very well-known family, called CP in Dyckhoff (1990) and Dyckhoff et al. (1997). CP is a family of natural combinatorial problems, admitted in numerous real-world applications such as computer science, industrial engineering, logistics, manufacturing, etc. Long and intensive research on this family of problems has led (and still leads) to the development of models and mathematical tools, so interesting by themselves that their application domains surpass the framework of CP problems.

The *two-dimensional cutting* (TDC) problem is a special CP problem. It consists of cutting n small rectangular pieces on a rectangular stock plate S of dimensions $L \times W$. Each piece type $i \in I$, where $I = \{1, \ldots, n\}$, is characterized by a length l_i, a width w_i, and a profit (or weight) $c_i > 0$ (input data are positive integers). Herein, we focus on the *nonexact fixed orientation constrained two-staged* two-dimensional cutting problem, denoted as FC_2TDC in Hifi (1999, 2001) and Hifi and Roucairol (2001). The FC_2TDC problem is a TDC problem such that the following are true:

— An upper demand value b_i is imposed on each piece type $i \in I$, so the problem C_TDC is *constrained*.

— Each piece type has a *fixed* orientation, so the problem FC_TDC is a *fixed orientation* problem.

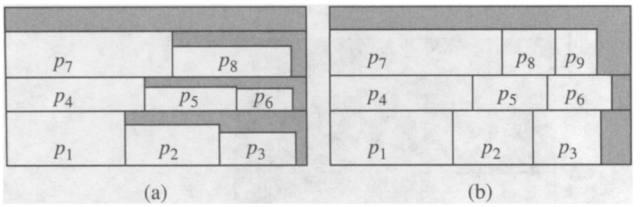
— The maximum number of guillotine cuts allowed to obtain each piece, in the final feasible cutting pattern, is fixed to two; therefore, the problem is two-staged (Gilmore and Gomory 1965, Hifi 2001, Hifi and Roucairol 2001, Lodi and Monaci 2003, Vanderbeck 2001).

— Trimming is allowed; in fact, an additional cutting stage is necessary for extracting a piece (Gilmore and Gomory 1965, Hifi 1999, Lodi and Monaci 2003), so the problem is nonexact.

The *non-exact case* or 2TDC *with trimming* is illustrated by Figure 1a, whereas the *exact case*, or the 2TDC *without trimming* is displayed in Figure 1b. The exact version of the problem is a special case of the more general nonexact case.

We consider both the weighted and unweighted versions of the FC_2TDC problem. The problem is *weighted* if the profit of a piece type is not equal to its area, and *unweighted* if $c_i = l_i w_i$, $i \in I$. We assume that pieces are in nondecreasing order of the widths; that is, $w_1 \leqslant w_2 \leqslant \cdots \leqslant w_n$. In addition, $\bar{w}_1 < \bar{w}_2 < \cdots < \bar{w}_r$, where $\forall i \in I$, $w_i \in \{\bar{w}_j : j = 1, \ldots, r\}$.

Figure 1. Examples of two-staged (guillotine) patterns: (a) nonexact and (b) exact cases.



An extensive survey of the literature on the 2TDC problem is available in Hifi (1999). A few approximate and exact approaches are known in the literature (e.g., Beasley 1985, Hifi and Roucairol 2001, Lodi and Monaci 2003, Morabito and Garcia 1998, Morabito and Arenales 1996, Vanderbeck 2001). Hifi (1999) undertook an extensive study of both U_2TDC and C_2TDC problems. Hifi and Roucairol (2001) proposed both approximate and exact algorithms for the FC_2TDC. Morabito and Garcia (1998) applied dynamic programming and column generation algorithms for a special case of the staged TDC problem. Recently, Lodi and Monaci (2003) provided two integer linear programming models for optimally solving several versions of the 2TDC problem. The case where k is fixed to three has been considered by Hifi (2001) and Vanderbeck (2001). Hifi (2001) solved the rotated problems and the fixed orientation U_3TDC problems using depth-first search exact algorithms. Vanderbeck (2001) investigated the 3TDC problem using nested decomposition within a classical column generation formulation.

This paper addresses the *fixed orientation constrained two-staged* two-dimensional cutting problem. More precisely, it considers the nonexact version of the FC_2TDC problem, i.e., where trimming is permitted.

This paper is organized as follows. In §2, we detail the exact algorithm specifying the lower bounds being used, the construction process of a strip, the development of upper bounds, the combination of strips, and the pruning strategies. In §3, we undertake an extensive computational experiment; evaluate the quality of the upper bounds and compare it to that of existing ones; and test the performance of the proposed algorithm. Finally, in §4, we summarize the main results of the present paper.

2. An Exact Algorithm

We solve the FC_2TDC problem using a new exact algorithm based on a branch-and-bound procedure using a bottom-up strategy. The proposed algorithm uses the same steps defined in Hifi and Roucairol (2001). It, however, uses different lower bounds and introduces new upper bounds and new pruning strategies. The algorithm (i) starts by computing an initial lower bound, (ii) constructs a set of horizontal or vertical (sub)strips and computes new upper bounds, and (iii) combines some of these (sub)strips to produce an optimal solution to the FC_2TDC problem. While combining (sub)strips, the algorithm uses new pruning strategies that avoid duplicate patterns.

This section is organized as follows. First, in §2.1 we summarize the four approximate algorithms considered in Hifi and M'Hallah (2001) and justify the adoption of two of these approximate algorithms to compute lower bounds for the new proposed algorithm. Second, in §2.2 we describe the two construction processes, called horizontal and vertical builds, which constitute the main mechanism of the algorithm. Third, in §2.3 we propose new upper bounds obtained at each node of the developed tree by solving constrained knapsack problems. Fourth, in §2.4 we describe the main steps of the exact algorithm, which combine previously built strips. Fifth and last, in §2.5, we propose several strategies that eliminate duplicate patterns (or symmetries): The first strategy uses the principle of a *dominated pattern* whereas the second uses an *order search* function.

2.1. Initial Lower Bound

The exact algorithm starts by computing an initial lower bound. A valid lower bound can be obtained using any of the following four approximate algorithms: BLP, SGA, ESGA, and HESGA (Hifi and M'Hallah 2001). These algorithms find a feasible solution to the *nonexact* FC_2TDC problem.

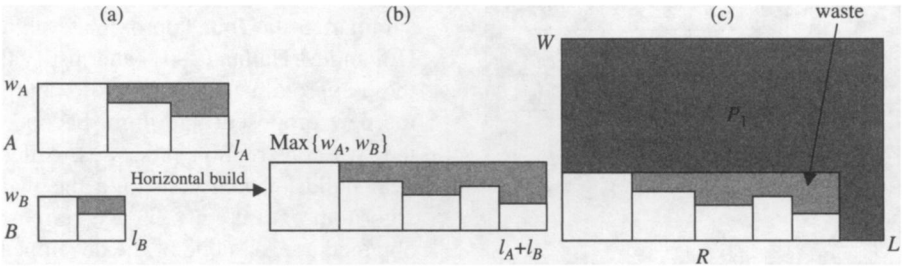
BLP, or *bottom left procedure*, is a simple constructive procedure that packs pieces on the first position available on the bottom left edge of the strip.

SGA, or *strip generation algorithm*, is a two-stage algorithm. The first stage constructs a set of different *uniform* and *general horizontal strips*. Each of these strips is obtained when a special single-bounded knapsack problem is solved via dynamic programming. The second stage combines a few of the generated strips to construct an *approximate horizontal cutting pattern*. Two cutting patterns are particularly considered: one that only combines uniform strips using a single bounded knapsack problem, and one that combines both uniform and general strips using a greedy algorithm.

The ESGA (*extended substrip generation algorithm*) is based on three observations relative to the structure of a(n) (near)optimal pattern. Generally, a(n) (near)optimal pattern has few *uniform horizontal strips*; few *general horizontal strips*; and *general* and *uniform strips* produced by applying some cuts on the W axis of the stock rectangle S. ESGA creates *some uniform* (respectively *general*) horizontal strips of length L while accounting for the width of the different pieces; selects the *best* of these strips for *filling* S; and deals with a *finite number of subrectangles*, which are also filled by some *uniform and/or general horizontal strips*. The last step of ESGA is accelerated when an upper bound that eliminates unnecessary branching is used.

Finally, HESGA combines ESGA with some hill-climbing strategies. Each step of ESGA is a tree stopped at the first level. For instance, a level corresponds to the subrectangles obtained using β-cuts, with $\beta \in \mathcal{W}_{(L,W)}$,

Figure 2. 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.



where $\mathcal{W}_{(L,W)}$ denotes the set of linear combinations of widths. Few HC strategies were reinforced to reduce the computational time of HESGA.

Any of these four strategies provides a lower bound to the 2TDC problem. However, we opted for the use of the bounds provided by the SGA and ESGA approximate algorithms and do not consider those obtained by BLP and HESGA. The BLP-based bound is not as tight as the bounds obtained by any of the remaining three approximate algorithms, whereas the quality improvement provided by HESGA does not warrant the additional computational time required. However, SGA produces good quality results within a very short computational time, and ESGA performs generally well. For the tested problems, SGA is on average 3.22% from the optimum within an average computational time of 0.12 seconds (the largest observed runtime equals 0.4 seconds). For the same instances, ESGA is only 0.8% from the optimum but requires 0.9 seconds on average with the largest runtime being 7.9 seconds. Subsequently, the SGA and ESGA approximate approaches are used to provide lower bounds for the problem at hand.

2.2. Constructing Substrips

The construction of each horizontal (sub)strip is obtained by combining all pieces and their duplicates through horizontal builds. In the following, we define the *general horizontal construction procedure*, which combines different piece types on the same strip.

DEFINITION 1. Let (l_A, w_A) and (l_B, w_B) be the dimensions of two substrips A and B (see Figure 2a). Let $R = (l_R, w_R) = (l_A + l_B, \max\{w_A, w_B\})$ be a new subrectangle obtained by joining A and B using a horizontal build (see Figure 2b). Then R is an *internal horizontal rectangle* if $(l_R, w_R) \leqslant (L, W)$ and $d_i(R) \leqslant b_i$, $i \in I$ where $d_i(R)$ denotes the number of occurrences of the ith piece on R.

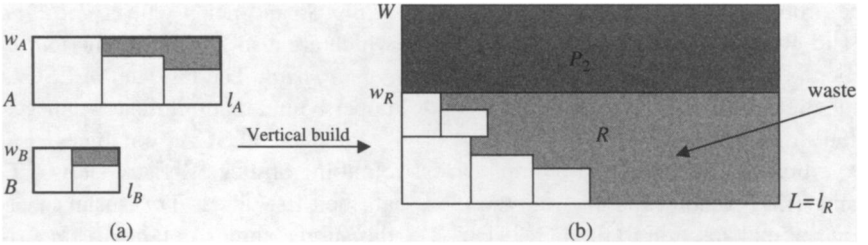
The optimal horizontal solution to the FC_2TDC problem is obtained by combining some horizontal substrips and/or some internal horizontal rectangles having the strip structure. Each of these rectangles is obtained by applying a vertical build. The vertical construction proceeds as defined below.

DEFINITION 2. Let (l_A, w_A) and (l_B, w_B) be the dimensions of two internal horizontal rectangles A and B, respectively, (see Figure 3a). Let $R = (l_R, w_R) = (\max\{l_A, l_B\}, w_A + w_B)$ be a new subrectangle obtained by joining A and B using a vertical build (see Figure 3b). Then R is an *internal vertical rectangle* if $(l_R, w_R) \leqslant (L, W)$ and $d_i(R) \leqslant b_i$, $i \in I$.

Here, each *internal horizontal rectangle* consists of (i) a set of initial pieces displayed according to Definition 1 and (ii) a set of horizontal (sub)strips displayed according to Definition 2. It is noteworthy that the above constructions are also applicable to vertical strips.

Given a stock rectangle S, a set of pieces and an internal (horizontal) rectangle R, we need to find an *upper bound* for the value of the best cutting pattern on S, that is constrained to include R. Let $g(R)$ be the *internal value* of R; that is, $g(R)$ is the sum of the weights of the pieces contained in R. Let $h(R)$ be the *maximum profit* generated by region P_1 of Figure 2c (respectively region P_2 of Figure 3b) where at most $b_i - d_i(R)$ items of piece i can

Figure 3. Representation of (a) two (sub)strips and (b) a resulting internal rectangle using a vertical build, placed in the initial stock rectangle.



be used in P_1 (respectively P_2). Then,

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

represents the *maximum profit that can be generated by a feasible cutting pattern* that is constrained to include R. Generally, finding the exact value of $h(R)$ without much computational effort is a difficult task. Thus, we estimate $h(R)$ with a new valid estimator $h'(R)$, where $h'(R) \geqslant h(R)$, and use $h'(R)$ to compute $f'(R)$, a new valid estimator of $f(R)$, as explained in the following.

2.3. An Upper Bound for a Subrectangle (α, β)

Herein, we develop a new upper bound $f'(R)$ for the maximum attainable value of a feasible cutting pattern $f(R)$. The value $f'(R)$ depends on the type of build being considered. For instance, when the build is horizontal, the estimation of $f(R)$ is based on the region P_1 of Figure 2c. However, when the build is vertical, the estimation of $f(R)$ is based on the region P_2 of Figure 3b. In our study, we focus on the horizontal FC_2TDC problem, because the vertical version of the problem simply permutes the lengths and widths of the stock rectangle and those of the piece types.

Let (α, β) $(\leqslant (L, W))$ be a subrectangle (or strip) and let $S_{\alpha,\beta}$ be a (sub)set of the initial pieces entering in the subrectangle (α, β), where $S_{\alpha,\beta} = \{i \in I\colon l_i \leqslant \alpha, w_i \leqslant \beta\}$. Hifi and Roucairol (2001) introduced several upper bounds for $f(R)$. These upper bounds were obtained by combining the solutions of some unconstrained and constrained single knapsack problems. The unconstrained single knapsack problems were:

- $UK_{\alpha,\bar{w}_j}$, $j = 1, \ldots, r$, which generate horizontal strips of widths $\bar{w}_j \leqslant \beta$ and optimal solution values

$$F^{\text{hor}}_{\bar{w}_j}(\alpha) = \max\left\{ \sum_{i \in S_{\alpha,\bar{w}_j}} c_i x_i\colon \sum_{i \in S_{\alpha,\bar{w}_j}} l_i x_i \leqslant \alpha;\ x_i \in \mathbb{N}, i \in S_{\alpha,\bar{w}_j} \right\};$$

and

- $UK^{\text{hor}}_{(\alpha,\beta)}$, which finds the optimal solution to the FU_2TDC problem, with value

$$G^{\text{hor}}_{\alpha}(\beta) = \max\left\{ \sum_{j=1}^{r} F^{\text{hor}}_{\bar{w}_j}(\alpha) y_j\colon \sum_{j=1}^{r} \bar{w}_j y_j \leqslant \beta;\ y_j \in \mathbb{N}, j = 1, \ldots, r \right\}.$$

x_i denotes the number of times piece i appears in the horizontal strip of width $\bar{w}_j$. y_j is the number of occurrences of the strip of width $\bar{w}_j$ in the subrectangle (α, β). The value $G^{\text{hor}}_{\alpha}(\beta)$ is an upper bound to the FC_2TDC problem. At the root node of the developed tree, (α, β) is simply (L, W), and the yielded estimator is the first upper bound.

The main results found in Hifi and Roucairol (2001) can be summarized as follows. Let A and B be two internal rectangles and let R be the resulting valid construction obtained by combining A and B either horizontally or vertically. When only unconstrained knapsack problems are used, the estimator of $f'(R)$ is obtained as follows: let k, $k \in \{1, \ldots, r\}$, be the index realizing $\bar{w}_k = w_R$. Then,

(i) When R is obtained by applying a *horizontal construction*,

$$f'(R) = g(R) + G^{\text{hor}}_{L}(W - w_R) + F^{\text{hor}}_{\bar{w}_k}(L - l_R).$$

(ii) When R is obtained by applying a *vertical construction*,

$$f'(R) = g(R) + G^{\text{hor}}_{L}(W - w_R).$$

When the constrained knapsack problems are used, the estimator of $f'(R)$ (for a horizontal construction) becomes $f'(R) = g(R) + G^{\text{hor}}_{L}(W - w_R) + f^{\text{hor}}_{\bar{w}_k}(L - l_R)$.

Herein, the main idea is to associate to each piece type i, $i \in I$, a strip of width w_i, $w_i \in \{w_1, \ldots, w_n\}$. The computed upper bounds are solely obtained by considering the modified constrained single knapsacks. Indeed, the *unconstrained single-knapsack* problems $UK_{\alpha,\bar{w}_j}$, $j = 1, \ldots, r$, and $UK^{\text{hor}}_{(\alpha,\beta)}$ are substituted with new *constrained single-knapsack* problems CK_{α,w_j}, $j \in I$, whose optimal solution values are

$$\phi^{\text{hor}}_{w_j}(\alpha) = \max\left\{ \sum_{i \in S_{\alpha,w_j}} c_i x_i\colon \sum_{i \in S_{\alpha,w_j}} l_i x_i \leqslant \alpha,\ x_i \leqslant b_i,\ x_i \in \mathbb{N}, i \in S_{\alpha,w_j} \right\},$$

and $K^{\text{hor}}_{(\alpha,\beta)}$, whose optimal solution value is

$$\Phi^{\text{hor}}_{\alpha}(\beta) = \max\left\{ \sum_{j \in I} \phi^{\text{hor}}_{w_j}(\alpha) y_j\colon \sum_{j \in I} w_j y_j \leqslant \beta,\ y_j \leqslant b_j,\ y_j \in \mathbb{N}, j \in I \right\}.$$

Applying a dynamic programming procedure on the problem with the largest width CK_{L,w_n} generates all the solution values corresponding to each subproblem CK_{L,w_j}, $j = 1, \ldots, n-1$. Subsequently, solving $K^{\text{hor}}_{(L,W)}$ yields the solution value at the root node. Similarly, solving $CK_{(L,W)}$ yields all solution values $\phi^{\text{hor}}_{w_j}(\alpha)$, for $\alpha = 0, \ldots, L$, $j \in I$. $\phi^{\text{hor}}_{w_j}(\alpha)$ is an upper bound to the solution value generated by the subrectangle (α, w_j), $j \in I$.

The following results show that the *constrained single-knapsack* problems yield a tighter upper bound to the FC_2TDC problem.

U1. Given an internal rectangle $R = (l_R, w_R)$, a valid and a not worse upper bound for $f(R)$ when a horizontal build is applied is

$$f'(R) = g(R) + \Phi^{\text{hor}}_{L}(W - w_R) + \phi^{\text{hor}}_{w_R}(L - l_R). \tag{1}$$

That is, a tighter bound can be obtained by considering nearly the same knapsack problems but with smaller feasible regions.

U2. Given an internal rectangle R with dimensions (l_R, w_R), a valid upper bound for $f(R)$ when a vertical build is applied is

$$f'(R) = g(R) + \Phi_L^{\text{hor}}(W - w_R). \tag{2}$$

Equation (2) simply states that the right part of an internal rectangle R can be reduced to zero given a certain arrangement of the pieces.

Sometimes, the subrectangle $(L, W - w_R)$ (i.e., the region P_2 of Figure 3) has a strip-structure. In this case, the construction step, explained in Definition 1, generates several nodes prior to reaching the optimal solution when R is part of this solution. However, there are cases when applying the construction step is not necessary, as shown in the following result.

U3. If an internal rectangle R is part of an optimal solution such that $W - w_R < 2w_1$, then only one more strip can be in this optimal solution. This strip is obtained by solving $CK_{L, W-w_R}$ involving all pieces $i \in I$ whose $w_i \leqslant W - w_R$ and $x_i \leqslant b_i - d_i(R)$.

The following result considers a complementary upper bound that can be obtained by solving another constrained knapsack problem.

U4. Let $P(R)$ be the area representing either P_1 of Figure 2 or P_2 of Figure 3. Then, we associate to $P(R)$ the linear problem $K^u_{P(R)}$ whose optimal objective value is

$$\mathcal{U}(P(R)) = \max\left\{ \sum_{j \in R_{P(R)}} c_j z_j : \sum_{j \in R_{P(R)}} (l_j w_j) z_j \leqslant \text{Area}(P(R)), \right.$$
$$\left. z_j \leqslant b_j - d_j(R_{P(R)}), z_j \in \mathbb{N}, j \in R_{P(R)} \right\},$$

where $R_{P(R)}$ is the set of pieces entering in $P(R)$, z_j is the number of times piece j occurs, where $j \in R_{P(R)}$. Evidently, we need to distinguish two cases: (i) $P(R)$ represents the area P_1, and (ii) $P(R)$ represents the area P_2. On the one hand, if $P(R) = P_1$, then the set of pieces $R_{P(R)}$ contains the pieces entering the part $(L - l_R, w_R)$ and the second part $(L, W - w_R)$, where R is the current internal rectangle. On the other hand, if $P = P_2$, then $R_{P(R)}$ contains the pieces entering the part $(L, W - w_R)$.

The problem $K^u_{P(R)}$ is a large single-bounded knapsack when $z_j \in \mathbb{N}$. Because the variables z_j are relaxed, then, initially, we reorder the pieces in decreasing order of profit to area ratios with ties broken in favor of the largest profit and, if $c_k/\pi_k = c_p/\pi_p$, then $k > p$ implies that $c_k > c_p$. We then apply a greedy procedure (see Martello and Toth 1990, 1997; Pisinger 1997) to $K^u_{P(R)}$ (and we estimate the upper bound for $P(R)$ with $\lfloor \mathcal{U}(P(R)) \rfloor$).

The area $P(R)$ can be normalized using linear combinations of the lengths and widths. For example, if $P(R)$ represents the area P_2, then a normalized area can be computed as follows: Let (X, Y) be the dimensions of the part $(L, W - w_R)$. Its normalized area is (X_0, Y_0), where

$$X_0 = \max_{x \in \mathcal{L}_{(X,Y)}} \{x\} \quad \text{and} \quad Y_0 = \max_{y \in \mathcal{W}_{(X,Y)}} \{y\}.$$

$\mathcal{L}_{(X,Y)}$ (respectively $\mathcal{W}_{(X,Y)}$) denotes the set of linear combinations of the lengths (respectively the widths) of the pieces that enter (X, Y). If $P(R)$ equals the area P_1, then we need to proceed in two stages: a first stage for the part $(L, W - w_R)$ and a second stage for the part $(L - l_R, w_R)$. Adding these bounds produces an upper bound for the region P_1.

However, limited computational results showed that the new algorithm performs well without this upper bound. When this upper bound is introduced, the algorithm reduces the number of generated nodes but, in some instances, at a cost of a larger computational time. It is therefore best to reserve the use of this upper bound to more complex instances. In our study, we use the following strategies:

(i) If the construction is vertical, we retrieve the structure of the upper bound estimating the area P_2. If the obtained structure satisfies the demand constraints, then the node is fathomed and the current lower bound is updated if necessary.

(ii) If the construction is horizontal, we estimate the upper bound of $(L, W - w_R)$ of P_1 and, for the part $(L - l_R, w_R)$ we proceed as follows: We approximately solve the problem CK_{L-l_R, w_R}, where the pieces placed in (l_R, w_R) are deduced from the set S_{L-l_R, w_R} and x_i is a nonnegative real, with i in S_{L-l_R, w_R}.

2.4. Combining Substrips

Box 1 describes the main steps of the exact algorithm, noted ALG01. First, ALG01 solves the knapsack problem CK_{L, w_n}; thus, generating all bounded (sub)strips. Second, it solves $K^{\text{hor}}_{L,W}$ generating all upper bounds corresponding to the (sub)rectangles $(L, y), 0 \leqslant y \leqslant W$. At the root node, ALG01 exits if the structure of the upper bound satisfies the demand constraints; otherwise, it computes an initial lower bound using either SGA or ESGA. If the approximate solution coincides with the initial upper bound, then ALG01 stops with the current feasible solution; otherwise, it proceeds with the branching process.

ALG01 combines substrips using two main lists: *Open* and *Closed*. The *Open* list is used to store all configurations for which the upper bound is larger than $\text{Opt}^{\text{hor}}(Best)$, where $\text{Opt}^{\text{hor}}(Best)$ is the current best (feasible) solution value (obviously, the configurations in *Open* cannot be discarded). The *Open* list is initialized to the different piece types R_i. R_i is characterized by the dimensions of the ith piece $(l_{R_i}, w_{R_i}) = (l_i, w_i)$, the *internal value* $g(R_i) = c_i$, the *estimator* $h'(R_i)$, the *order search* (cf. §2.5.2), and a vector $d = d(R_i)$ of dimension n, where $d_j(R_i) \leqslant b_j$, $j \in I$.

The *Closed* list is used to save all best solutions obtained with the construction processes of Definitions 1 and 2. The *Closed* list is initialized to the empty set.

Box 1 Main steps of the exact algorithm ALG01.

Input: an instance of the FC_2TDC problem.
Output: an optimal (horizontal) solution *Best* with value $Opt^{hor}(Best)$.
- Solve the problems CK_{L,w_n} and $K^{hor}_{L,W}$;
- Let *Sol* be the solution of $K^{hor}_{L,W}$, $\Phi^{hor}_L(W)$ its value and $d(Sol)$ its structure;
  If $\forall i \in I,\ d_i(Sol) \leqslant b_i$, then **exit** with the optimal solution *Sol* with value $\Phi^{hor}_L(W)$;
- Let *Best* be the feasible solution obtained by applying SGA
  (or ESGA) and $SGA(Best)$ (or $SGA(Best)$) denotes its solution value;
- Set $Opt^{hor}(Best) = SGA(Best)$ or $ESGA(Best)$;
- If $Opt^{hor}(Best) = \Phi^{hor}_L(W)$ then **exit** with the optimal solution *Best* with value $Opt^{hor}(Best)$;
- Set $Open = \{R_1, R_2, \ldots, R_n\}$, $Closed = \{\}$ and $Stop = false$;

Repeat
Let $R \in Open$ be the internal rectangle with the highest value f';
If $f'(R) - Opt^{hor}(Best) \leqslant 0$, 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 *general horizontal build* (see Definition 1) or a vertical build (see Definition 2) of R with some internal rectangles of *Closed*;
   (b) the dimensions (l_q, w_q) are less than or equal to (L, W);
   (c) $\forall q \in Q, d_i(q) \leqslant b_i,\ i \in I$;
3. for each element q of Q, do:
   (a) compute $g(q)$ and $h'(q)$;
   (b) If $g(q) > Opt^{hor}(Best)$ then set $Best = q$ and $Opt^{hor}(Best) = g(q)$;
   (c) If $g(q) + h'(q) > Opt^{hor}(Best)$ then
      (c.1) q is labeled with a new order search if it is obtained with a horizontal and/or vertical build;
      (c.2) set $Open = Open \cup \{q\}$;
4. If $Open = \{\}$ then $Stop := true$;

end;
Until *Stop*;
- Exit with the optimal solution *Best* with value $Opt^{hor}(Best)$.

At each iteration, an element R is chosen from *Open* and transferred to *Closed*. A set Q of new *internal rectangles* is created by combining R with some elements of *Closed*, using a horizontal or a vertical build, or both. In this case, if R coincides with a piece R_i, $i \in I$, then it is combined with all elements of *Closed*; otherwise, R is combined with each element, say R', where R' is a piece and its order search satisfies certain conditions (as discussed in §2.5.2).

Moreover, if a horizontal build is made, then the obtained construction is transferred to *Open* and labeled by using the order search. Note that each new construction obtained by a horizontal construction is considered as a new piece (substrip) for the vertical construction. Finally, the algorithm stops when *Open* is reduced to the empty set, or, differently stated, when it is no longer possible to construct a better candidate internal rectangle.

2.5. Avoiding Symmetries or Duplicate Patterns

Avoiding symmetries or duplicate patterns is critical when dealing with an exact algorithm for cutting stock problems because it can substantially reduce the computational time. In this case, we avoid duplicate patterns using two strategies: **S1** (the *dominated patterns*) and **S2** (the *order search function*).

2.5.1. Dominated Patterns S1. Let A and B be two substrips and let $C := (A/B)$ be the pattern obtained by vertically combining A and B. If C is valid, then C is a dominated pattern if one of the following two propositions holds true:

(a) $\exists k \in I$ such that $(l_k, w_k) \leqslant (L - l_A, w_A)$ and $b_k - d_k(C) > 0$;

(b) $\exists k' \in I$ such that $(l_{k'}, w_{k'}) \leqslant (L - l_B, w_B)$ and $b_{k'} - d_{k'}(C) > 0$.

Intuitively, if two strips A and B are vertically combined (resulting in C), and if either of the two strips can be enlarged at least by one piece, then C should not be constructed because a better strip can be obtained.

Recall that we use two types of builds: horizontal and vertical. These two types of builds are integrated in the search process of Hifi and Roucairol's (2001) algorithm. The principle can be summarized as follows: On the one hand, if A is taken from *Open* and comprises a unique piece, then A is combined with all elements of *Closed*. On the other hand, if A is composed by several pieces,

then A is combined with the elements of *Closed* that comprise a single piece. On the other hand, the search process can be accelerated using the order search.

2.5.2. The Order Search S2. In order to eliminate some duplicate patterns, we establish an *order search* that we apply to each piece or (sub)strip. All elements of *Open* are initially sequenced in nondecreasing order, i.e., each piece's type R_i, $i \in I$, is replaced by the element R_i^θ, $i = \theta = 1, \ldots, n$ (θ denotes the order search of a certain (sub)pattern). If A coincides with an element composed of a unique piece R_k^θ, then the order search of A, denoted θ_A, is equal to $\theta_{R_k^\theta} = A_{\theta_k}$. On the other hand, if A comprises at least two pieces, say R_k^θ and R_s^θ, then the order search of A is set equal to $\min\{\theta_{R_k^\theta}, \theta_{R_s^\theta}\}$. In this case, if $\theta_{R_k^\theta} \leqslant \theta_{R_s^\theta}$, then R_s is placed on the right part of A (denoted $R_k \,|\, R_s$); otherwise, it is placed on the left part of A (denoted $R_s \,|\, R_k$). Using the above notions, we can apply the following result:

Let A and B be two internal rectangles, where $A \in$ *Open* and comprises at least two pieces, and $B \in$ *Closed* and comprises a unique piece type. Assume that the combination of A and B produces an internal rectangle. Then, the horizontal build between A and B is a duplicate internal rectangle if $\theta_A < \theta_B$, where θ_A (resp. θ_B) is the order of A (resp. B).

Note that the aim is to show how the used process can produce an equivalent pattern composed of A and B (with some shifting). Indeed, consider the two following cases: (a) A comprises two pieces, and (b) A comprises a series of different (or equal) pieces.

(a) Let $A = C \,|\, D$ and B is a piece.

(i) If $B \in$ *Closed* and $A \in$ *Open*, then C and D are also included in *Closed*.

(ii) Because B, C, and D are pieces, then the patterns $C \,|\, B$ and $D \,|\, B$ (or $B \,|\, D$) necessarily exist, if B, C and D are in *Closed*.

(iii) If $D \,|\, B \in$ *Open* (or $B \,|\, D \in$ *Open*) and $C \in$ *Closed*, then $C \,|\, D \,|\, B$ (or $C \,|\, B \,|\, D$) is constructed, because $\min\{\theta_D, \theta_B\} \geqslant \theta_C$; otherwise, if $C \in$ *Open* and $D \,|\, B \in$ *Closed* (or $B \,|\, D \in$ *Closed*), then $C \,|\, D \,|\, B$ (or $C \,|\, B \,|\, D$) is constructed, because C is a piece.

(b) Consider the general case, where $A = A_1 | A_2 | \cdots | A_{m-1} | A_m$ (following this order from the left to right) and B comprises a single piece.

Because A is selected from *Open*, then all the pieces contributing to construct A are necessarily in the *Closed* list, i.e., $A_1, A_2, \ldots, A_{m-1}$, and A_m are in *Closed*. So, $\theta_A < \theta_B$ and $\theta_{A_1} \leqslant \theta_{A_2} \leqslant \cdots \leqslant \theta_{A_{m-1}} \leqslant \theta_{A_m}$. For the general case, we can distinguish two cases:

Case (b1) $\theta_{A_m} \leqslant \theta_B$, and

Case (b2) $\theta_{A_1} \leqslant \cdots \leqslant \theta_{A_{k-1}} \leqslant \theta_B \leqslant \theta_{A_k} \leqslant \cdots \leqslant \theta_{A_m}$.

Case (b1) is a special case of (b2). Subsequently, we only need to show how an equivalent pattern can be realized by using a simple process.

(i) Initially, the A_m and A_{m-1} which are single pieces are combined into a pattern $H = A_{m-1} \,|\, A_m$; so, $\theta_{A_{m-1} \,|\, A_m} = \min\{\theta_{A_{m-1}}, \theta_{A_m}\} = \theta_{A_{m-1}}$.

(ii) The pattern $H = A_{m-1} \,|\, A_m$ exists and A_{m-2} is a piece. If $A_{m-2} \in$ *Open* and $H \in$ *Closed* then $A_{m-2} \,|\, H = A_{m-2} \,|\, A_{m-1} \,|\, A_m$ is constructed; otherwise, if $H \in$ *Open* and $A_{m-2} \in$ *Closed*, and because $\theta_H = \theta_{A_{m-1}} \geqslant \theta_{A_{m-2}}$, then the construction $A_{m-2} \,|\, H$ is also realized.

(iii) The same process is repeated until reaching the internal rectangle $G = A_k | \cdots | A_m$. Now, for the pattern B, we use the same process. Indeed, if $B \in$ *Open* and $G \in$ *Closed*, then $B \,|\, G$ is realized; otherwise, $B \in$ *Closed* and $G \in$ *Open*, and $B \,|\, G$ is constructed because $\theta_G = \theta_{A_k} \geqslant \theta_B$.

(iv) Finally, by repeating the same process until the last pattern A_1, the construction $A_1 | \cdots | A_{k-1} | B | A_k | \cdots | A_m$ is realized, which is equivalent to the pattern $A \,|\, B$ by shifting B from the middle to the end.

We use the same process when vertical constructions are considered. Indeed, another order search is applied for the vertical construction and all constructions obtained by a horizontal build are labeled starting from $n + 1$. Each (sub)strip obtained from a horizontal build can be viewed as a new piece with a new order.

3. Computational Results

In this section, we undertake an extensive computational experiment whose purpose is twofold. First, in §3.1, we evaluate the quality of the proposed bounds and compare them to those existing in the literature. Second, in §3.2, we evaluate the performance of the proposed algorithm and compare it to a recent exact algorithm.

We coded the different versions of the algorithm in C and ran them on an UltraSparc10 (250 MHz and 128 Mb of RAM). We studied their performance on benchmark problems from the literature. These benchmark problems[1] represent 38 instances extracted from Hifi and Roucairol (2001) and Lodi and Monaci (2003). The first 14 of these problems are *weighted* whereas the last 24 problems are *unweighted* with $c_i = l_i w_i$, $i \in I$.

3.1. Upper Bound Comparison

The purpose of this section is to evaluate the quality of the proposed upper bound. Indeed, we compare the proposed bound *UCK*, using the constrained knapsack structure, to those existing in the literature, in particular to those of Hifi and Roucairol (2001), Lodi and Monaci (2003), Gilmore and Gomory (1961), and to the optimal solution value they are estimating. Hifi and Roucairol (2001) computed their upper bound *UUK* using unconstrained knapsack problems. Lodi and Monaci (2003) solved two integer linear models to get their upper bound *UM*. Gilmore and Gomory (1961) generated *UCG* using column generation.

When evaluating the quality of the bound, we distinguish the two possible cases: (a) the case where the first cut is horizontal, and (b) the case where the first cut is vertical. Columns 2 to 5 of Table 1 display the results for the case where the first cut is horizontal, whereas columns 6 to 9 display the same results for the case where the first cut

Table 1. Quality of the proposed upper bound.

	The first cut is horizontal				The first cut is vertical			
Instance	*UCK*/Opt	*UUK*/Opt	*UM*/Opt	*UCG*/Opt	*UUK*/Opt	*UCK*/Opt	*UM*/Opt	*UCG*/Opt
HH	1.000	1.135	1.164	1.017	1.131	1.254	1.333	1.080
2	1.137	1.145	1.135	1.045	1.101	1.259	1.141	1.051
3	1.151	1.302	1.166	1.101	1.103	1.287	1.145	1.088
A1	1.000	1.209	1.180	1.032	1.000	1.187	1.170	1.094
A2	1.121	1.458	1.171	1.033	1.048	1.461	1.161	1.057
STS2	1.022	1.097	1.073	1.018	1.000	1.019	1.027	1.000
STS4	1.000	1.035	1.057	1.018	1.013	1.030	1.048	1.030
CHL1	1.006	1.069	1.091	1.049	1.034	1.064	1.114	1.007
CHL2	1.006	1.088	1.106	1.017	1.028	1.086	1.187	1.016
CW1	1.000	1.053	1.117	1.072	1.000	1.000	1.123	1.007
CW2	1.049	1.060	1.123	1.034	1.055	1.090	1.162	1.091
CW3	1.092	1.123	1.128	1.064	1.043	1.043	1.060	1.028
Hchl2	1.037	1.056	1.056	1.012	1.035	1.077	1.069	1.008
Hchl9	1.053	1.171	1.040	1.019	1.071	1.180	1.049	1.003
2s	1.106	1.152	1.152	1.046	1.101	1.143	1.143	1.051
3s	1.018	1.031	1.077	1.033	1.000	1.023	1.067	1.066
A1s	1.000	1.000	1.017	1.014	1.000	1.001	1.031	1.009
A2s	1.000	1.017	1.052	1.015	1.000	1.033	1.043	1.023
STS2s	1.006	1.016	1.023	1.010	1.006	1.006	1.011	1.009
STS4s	1.016	1.024	1.034	1.017	1.011	1.012	1.034	1.011
OF1	1.000	1.000	1.032	1.000	1.020	1.031	1.053	1.006
OF2	1.027	1.089	1.113	1.081	1.047	1.101	1.110	1.071
W	1.000	1.023	1.067	1.066	1.018	1.031	1.077	1.033
CHL1s	1.000	1.013	1.013	1.012	1.008	1.042	1.047	1.006
CHL2s	1.053	1.078	1.078	1.046	1.008	1.066	1.066	1.000
A3	1.000	1.020	1.041	1.040	1.000	1.009	1.036	1.009
A4	1.024	1.058	1.071	1.000	1.037	1.059	1.067	1.010
A5	1.000	1.038	1.052	1.050	1.017	1.055	1.060	1.015
CHL5	1.000	1.102	1.102	1.044	1.049	1.163	1.163	1.108
CHL6	1.012	1.020	1.020	1.020	1.011	1.032	1.038	1.012
CHL7	1.003	1.009	1.010	1.005	1.008	1.017	1.018	1.005
CU1	1.000	1.000	1.015	1.015	1.000	1.000	1.025	1.012
CU2	1.000	1.000	1.006	1.006	1.011	1.011	1.039	1.038
Hchl3s	1.010	1.031	1.041	1.015	1.025	1.025	1.052	1.018
Hchl4s	1.041	1.081	1.091	1.037	1.031	1.077	1.106	1.015
Hchl6s	1.009	1.010	1.026	1.016	1.006	1.014	1.031	1.015
Hchl7s	1.003	1.009	1.015	1.012	1.002	1.006	1.009	1.004
Hchl8s	1.136	1.344	1.241	1.051	1.200	1.239	1.200	1.037
Average	1.030	1.083	1.079	1.032	1.032	1.085	1.087	1.030
Nb. Opt	14	4	0	1	8	2	0	2

Note. Effectiveness of the new upper bound, at the root node, compared to the bounds published by Hifi and Roucairol (2001) and Lodi and Monaci (2003).

is vertical. Table 1 reports the distance of the new upper bound *UCK* from the optimal solution measured by the ratio *UCK*/Opt, the distance of Hifi and Roucairol (2001), the bound of *UUK* from the optimal given by the ratio *UUK*/Opt, as well as the ratios *UM*/Opt and *UCG*/Opt as reported in Lodi and Monaci (2003).

Table 1 shows that the new upper bound *UCK* performs better than both *UUK* and *UM*. In addition, *UCK* is competitive with the column generation bound *UCG*. Indeed, the 3.13% average gap of *UCK* from the optimal solution is very close to the 3.11% gap of *UCG*. For weighted problems, these average gaps are, respectively, 4.82% and 3.79% for *UCK* and *UCG* for the horizontal case, and become 4.73% and 4.00% for the vertical case. For unweighted problems, the gaps are 1.94% and 2.87% for the horizontal case and 2.41% and 2.43% for the vertical case. *UCK* does, in addition, coincide with the optimal solution in more occasions than *UCG*. In fact, *UCK* produces 29% of the optimal solutions whereas its counterpart *UCG* reaches the optimal solution only on 5% of the cases.

As will be discussed in §3.2, the *UCK* initial and internal upper bounds are very efficient within a branch and bound algorithm. At each node of the developed tree, each upper bound is generally obtained in a constant time, which makes the algorithm more efficient.

The average percent gaps, displayed in Table 1 for the horizontal case, are illustrated in Figure 4. The figure compares the quality of the four bounds. It shows that the *UCK* and *UCG* curves are lower than the *UUK* and *UM* corresponding curves.

Figure 4. Comparison of the quality of the four upper bounds when the first cut is horizontal.

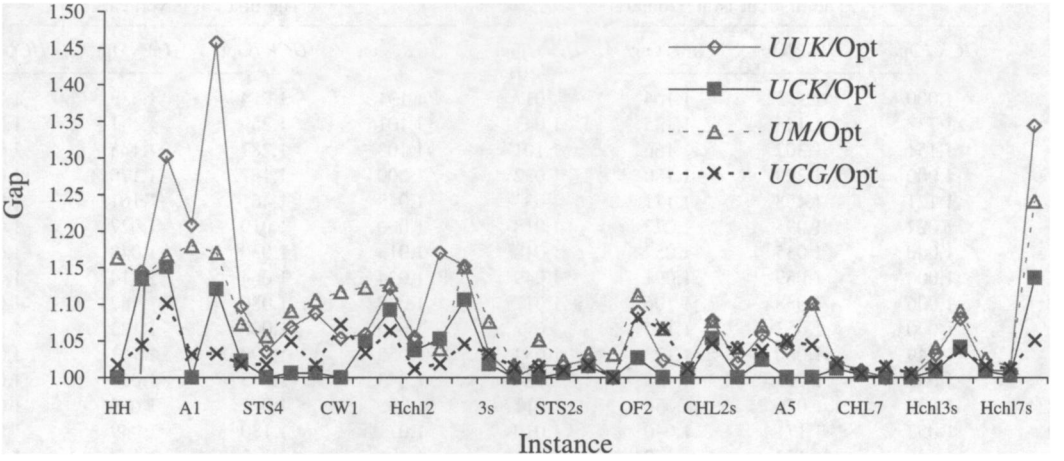


Table 2. Performance of the new algorithm.

		ALG0		ALG01_ESGA				ALG01_SGA			
Instance	Opt	Node	Time	Node	Time	%Node	%Time	Node	Time	%Node	%Time
HH	10,689	79	0.20	0	0.01	100	95.00	0	0.01	100	95.00
2	2,535	1,789	2.90	831	1.51	53.55	48.64	941	1.50	47.40	48.98
3	1,720	258	0.20	74	0.11	71.32	45.00	79	0.05	69.38	75.00
A1	1,820	254	0.20	0	0.15	100.00	25.00	0	0.06	100	70.00
A2	2,315	820	0.80	81	0.26	90.12	67.50	97	0.11	88.17	86.25
STS2	4,450	2,218	5.60	161	0.95	92.74	83.10	368	0.36	83.41	93.59
STS4	9,409	3,411	9.20	0	0.03	100	99.67	0	0.03	100	99.67
CHL1	8,360	18,643	610.10	199	1.45	98.93	99.76	1,752	3.63	90.60	99.41
CHL2	2,235	133	0.10	19	0.33	85.71	−230.00	53	0.07	60.15	30.00
CW1	6,402	44	1.00	0	0.46	100.00	54.90	0	0.46	100	54.90
CW2	5,354	87	2.10	52	0.67	40.23	68.40	52	0.13	40.23	93.87
CW3	5,287	205	6.40	65	2.95	68.29	54.19	103	1.27	49.76	80.28
Hchl2	9,630	87,202	N/A	2,186	5.15	97.49	99.95	5,493	26.82	93.70	99.75
Hchl9	5,100	23,428	858.00	2,193	4.10	90.64	99.52	2,862	6.83	87.78	99.20
2s	2,430	1,923	4.60	1,006	1.86	47.69	59.74	1,312	2.05	31.77	55.63
3s	2,599	148	0.10	44	0.15	70.27	−50.00	55	0.11	62.84	−10.00
A1s	2,950	3	0.10	0	0.01	100	90.00	0	0.01	100	90.00
A2s	3,423	53	0.20	0	0.04	100	80.00	0	0.04	100	80.00
STS2s	4,569	1,015	1.10	6	0.66	99.41	40.00	98	0.21	90.34	80.91
STS4s	9,481	3,710	8.90	588	0.75	84.15	91.57	1,158	1.38	68.79	84.49
OF1	2,713	63	0.10	0	0.01	100	91.67	0	0.01	100	91.67
OF2	2,515	145	0.10	8	0.05	94.48	50.00	34	0.06	76.55	40.00
W	2,623	231	0.20	0	0.01	100	95.00	0	0.01	100	95.00
CHL1s	13,036	2,723	6.10	0	0.34	100	94.44	0	0.34	100	94.44
CHL2s	3,162	93	0.10	41	0.15	55.91	−7.14	41	0.04	55.91	71.43
A3	5,380	110	0.30	0	0.25	100	16.67	0	0.05	100	83.33
A4	5,885	958	1.60	102	0.45	89.35	71.88	102	0.07	89.35	95.63
A5	12,553	1264	2.40	0	0.06	100	97.50	0	0.06	100	97.50
CHL5	363	151	0.10	0	0.01	100	91.67	0	0.01	100	91.67
CHL6	16,572	5,515	38.20	548	1.66	90.06	95.65	553	0.57	89.97	98.51
CHL7	16,728	7,159	44.60	212	1.93	97.04	95.67	1,004	1.67	85.98	96.26
CU1	12,312	31	1.00	0	0.18	100	82.00	0	0.18	100	82.00
CU2	26,100	2	2.70	0	0.49	100	81.92	0	0.49	100	81.92
Hchl3s	11,961	5,251	15.30	447	0.69	91.49	95.49	523	0.32	90.04	97.91
Hchl4s	11,408	18,317	507.40	1,538	3.65	91.60	99.28	3,168	24.47	82.70	95.18
Hchl6s	60,170	3,937	14.80	78	3.02	98.02	79.59	581	0.39	85.24	97.36
Hchl7s	62,459	10,184	96.20	483	4.42	95.26	95.41	955	2.47	90.62	97.43
Hchl8s	729	4,204	26.40	361	0.19	91.41	99.28	958	0.63	77.21	97.61
Average		5,414.76	61.30	297.97	1.07	89.08	64.42	587.95	2.03	83.89	81.89

Note. The first cut is horizontal. N/A means that the algorithm exceeds 30 minutes (the limited computing time).

3.2. Performance of the New Algorithm

In this section, we compare the performance of the proposed new exact algorithm ALG01 to that of Hifi and Roucairol (2001), called herein ALG0. Specifically, we consider two versions of ALG01. The first version, denoted ALG01_ESGA, starts with an initial lower bound obtained with ESGA. The second version, denoted ALG01_SGA, starts with the approximate solution obtained from SGA. (Recall that ESGA performs, in general, better than SGA whereas SGA produces good results within smaller computational times.)

The performance of ALG01 is displayed, for each of the 38 problems, in Table 2 when the first cut is horizontal. Table 2 shows that ALG01, using ESGA or SGA, performs better than ALG0.

The analysis of Table 2 reveals that ALG01_ESGA saves on average 64.42% computational time. This time gain is particularly sizeable for some instances. For example, the optimal solution of Hchl2 was not reached by ALG0, whereas ALG01_ESGA produced it in 5.15 seconds. In addition, both versions of the algorithm reduce the number of generated nodes. Indeed, the reduction varies between 40.23% and 100%, with an average reduction of 89.08%.

Similarly, the second version of ALG01 using SGA is efficient. Indeed, ALG01_SGA is faster than ALG0 with an average computational time gain of 81.89% and an average reduction of generated nodes of 83.89%. The computational gain—when a gain is obtained—is at least 30% and the node reduction is at least 31.77%. These gains can be very sizeable in many instances, as observed for instances Hchl2, Hchl9, and CHL1s.

In some instances ALG01_SGA performs better than ALG01_ESGA. However, for the more complex instances, ALG01_ESGA is generally more efficient—for example, instances Hchl2 and Hchl4s. The additional computational time required by ALG01_ESGA is due to the computational time of the ESGA algorithm. Although, for the same instances, SGA needs on average 0.12 seconds with the largest observed value equal to 0.40 seconds, ESGA requires an average run time of 0.90 seconds with the largest observed value equal to 7.90 seconds (see Hifi and M'Hallah 2001).

Having discussed the performance of ALG01 using both approaches SGA and ESGA, we now evaluate the impact of (i) lower bounds, upper bounds U1–U2, and the optimality criterion U3, and (ii) adding symmetry or duplicate pattern detection S1 and S2.

First, we run ALG01_ESGA without the new bounds and without the symmetry or duplicate pattern detection, and tally the number of nodes in Column 3 of Table 2. Second, we run ALG01_ESGA with the new bounds and without the symmetry or duplicate pattern detection and record the number of generated nodes, denoted *Node* 1. We then compute *%Node* 1 the percent gain induced by the new bounds on the number of generated nodes. Finally, we compare *Node* 1 to *Node* 2 the number of generated nodes of ALG01_ESGA when both the new bounds and the symmetry or duplicate pattern detection are applied. We, subsequently, compute *%Node* 2 the percent improvement induced by the use of the news bounds and of the symmetry or duplicate pattern detection in comparison to the case where neither is used.

Table 3. Effect of lower or upper bounds and duplicate patterns at internal nodes.

	Lower and upper bounds		Duplicate patterns	
Instance	*Node* 1	*%Node* 1	*Node* 2	*%Node* 2
STS4	0	100.00	—	—
CHL1	3,982	78.64	199	98.93
Hchl2	5,268	93.96	2,186	97.49
Hchl9	6,020	74.30	2,193	90.64
CHL1s	0	100.00	—	—
CHL6	1,345	75.61	548	90.06
CHL7	2,253	68.53	212	97.04
Hchl4s	5,022	72.58	1,538	91.60
Hchl7s	3,856	62.14	483	95.26
Hchl8s	1,452	65.46	361	91.41

Note. The first cut is horizontal.

Table 3 reports *Node* 1, *%Node* 1, *Node* 2, and *%Node* 2 for 10 instances. These instances reflect the general behavior of the problems and are considered significant in terms of the number of generated nodes. It is noteworthy that *%Node* 2 for each of these 10 instances is that displayed in Table 2 for ALG01_ESGA.

The use of the new bounds definitely improves the performance of ALG01_ESGA. The percent gain on the number of generated nodes varies from 62.14% to 93.96%. These percentages further increase to attain 98.93%, when, in addition to the new bounds, the symmetry and duplicate pattern detection are used.

4. Conclusion

The constrained two-staged two-dimensional cutting problem (2TDC) is a classical variant of the well-known family of cutting and packing problems. In this paper, we have proposed an exact algorithm for solving the 2TDC problem. We have introduced a new upper bound and compared its performance to a number of existing bounds, including the best-known column generation bound. Moreover, we have introduced a new exact algorithm based on a branch-and-bound procedure. To curtail the search of the branch-and-bound procedure, we have proposed new pruning strategies. We have studied two versions of the proposed algorithm. The first version starts by using the strip generation procedure, and the second uses an extended version of the strip generation algorithm. We have evaluated the performance of both versions of the algorithm through a number of experiments that we conducted on problem instances from the literature with different sizes and densities. The computational results indicate that the proposed approaches perform well and show that the algorithms solve the different problem instances efficiently within short computing times.

Endnote

1. Publicly available from ftp://panoramix.univ-paris1.fr/pub/CERMSEM/hifi/OR-Benchmark.html.

Acknowledgments

The authors thank two anonymous referees for their helpful comments and pertinent suggestions that contributed to the improvement of the presentation and the contents of this paper.

References

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

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

Dyckhoff, H., G. Scheithauer, J. Terno. 1997. Cutting and packing (C&P). M. Dell'Amico, F. Maffioli, S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization*. John Wiley and Sons, Chichester, U.K.

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

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

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

Hifi, M. 1999. Contribution à la résolution de quelques problèmes difficiles de l'optimisation combinatoire. [A contribution to the solution of some difficult problems of combinatorial optimization] Habilitation thesis, PR*i*SM, Université de Versailles, Saint-Quentin-en-Yvelines, France.

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

Hifi, M., R. M'Hallah. 2001. Strip generation algorithms for constrained two-dimensional two-staged cutting stock problems. Technical report 2001-Cahiers Bleu, Maison des Sciences Economiques, Université Paris 1 Panthéon-Sorbonne, France.

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

Kantorovich, L. V. 1960. Mathematical methods of organizing and planning production. *Management Sci.* **6** 363–422.

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

Martello, S., P. Toth. 1990. An exact algorithm for large unbounded knapsack problems. *Oper. Res. Lett.* **9** 15–20.

Martello, S., P. Toth. 1997. Upper bounds and algorithms for hard 0–1 knapsack problems. *Oper. Res.* **45** 768–778.

Morabito, R., M. Arenales. 1996. Staged and constrained two-dimensional guillotine cutting problems: An and-or-graph approach. *Eur. J. Oper. Res.* **94/3** 548–560.

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

Pisinger, D. 1997. A minimal algorithm for the 0–1 knapsack problem. *Oper. Res.* **45** 758–767.

Vanderbeck, F. 2001. A nested decomposition approach to a 3-stage 2-dimensional cutting stock problem. *Management Sci.* **47** 864–879.