Integer Linear Programming Models for 2-staged Two-Dimensional Knapsack Problems

Andrea Lodi, Michele Monaci

Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna
Viale Risorgimento, 2 - 40136 - Bologna (Italy)

E-mails: alodi@deis.unibo.it, mmonaci@deis.unibo.it

Submitted October 2000, Revised September 2001

Abstract

We are given a unique rectangular stock of material S, with height H and width W, and a list of m rectangular shapes to be cut from S. Each shape's type i ($i = 1, \ldots, m$) is characterized by a height $\bar{h}_i$, a width $\bar{w}_i$, a profit $\bar{p}_i$, and an upper bound ub_i indicating the maximum number of items of type i which can be cut. We refer to the *Two-Dimensional Knapsack* **(TDK)** as the problem of determining a cutting pattern of S maximizing the sum of the profits of the cut items. In particular, we consider the classical variant of TDK in which the maximum number of cuts allowed to obtain each item is fixed to 2, and we refer to this problem as 2-staged TDK (2TDK). **For 2TDK we present two new Integer Linear Programming models, we discuss their properties, and we compare them with other formulations in terms of bound. Finally, both models are computationally tested within a standard branch-and-bound framework on a large set of instances from the literature by reinforcing them with the addition of linear inequalities which avoid symmetries.**

Keywords: Packing, Cutting, Integer Linear Programming

1 Introduction

The problem of cutting a given set of small rectangles (*items*) from large identical *stock rectangles* of material has been regarded as a prototypical problem in the field of Cutting & Packing (see Dyckhoff, Scheithauer, and Terno [2] for an annotated bibliography) ever since the seminal work of Gilmore and Gomory [7]. In [7] **these** authors discussed a large variety of multi-dimensional cutting problems, moving from the definition of the *Two-Dimensional Cutting Stock Problem* in which the objective function is to minimize the number of large rectangles used. In an earlier work, Gilmore and Gomory [6] proposed a column generation approach to solve the One-Dimensional version of the above problem, which calls for cutting a set of one-dimensional items from the minimum number of identical

bars of material. In [7] this approach is extended **to solve the two-dimensional version of the problem**, and each *slave* problem is as follows: a profit is associated to each item, and a unique large rectangle has to be cut so as to obtain a subset of the items whose sum of the profits is a maximum. This latter problem is referred in [7] as the *Cutting Knapsack Problem* to emphasize that a unique stock rectangle and a set of profits are considered. **In this paper we refer to the former problem as** *Two-Dimensional Cutting Stock* **(or** *Two-Dimensional Bin Packing*[1]**, see Lodi, Martello and Monaci [14] for a recent survey), and to the latter as** *Two-Dimensional Knapsack* **(TDK).**

More formally, in TDK we are given a unique rectangular stock of material S, with height H and width W, and a list of m rectangular shapes to be cut from S. Each shape's type i ($i = 1, \ldots, m$) is characterized by a height $\bar{h}_i$, a width $\bar{w}_i$, a profit $\bar{p}_i$, and an upper bound ub_i indicating the maximum number of items of type i which can be cut. The problem calls for the determination of a cutting pattern of S maximizing the sum of the profits of the cut items (see, Figure 1(a)).

TDK can be found in the literature in many variants deriving from additional requirements or extensions. One of the most common of these variants is determined by the requirement of producing cutting patterns of *guillotine type*, i.e., in which each item must be cut with a sequence of edge to edge cuts parallel to the edges of S (see Figure 1(b)). A special case of this class of problems is the so called *d-staged* **Two-Dimensional Knapsack**, in which the maximum number of guillotine cuts allowed to obtain each item is fixed to d. This latter class of problems was introduced by Gilmore and Gomory [7], and has received considerable attention due to relevant real-world applications.

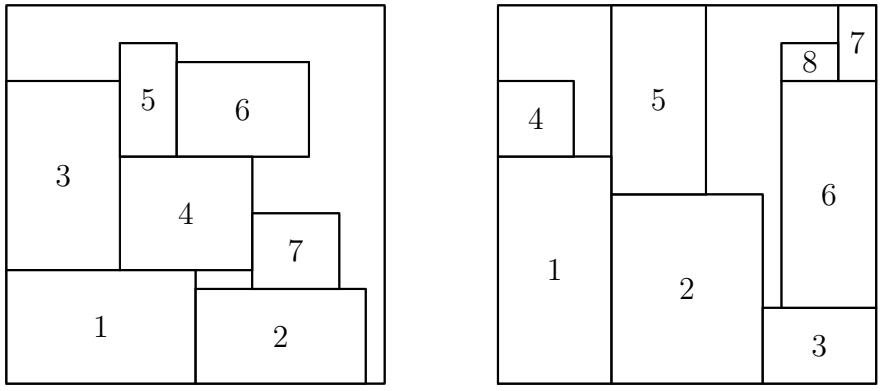

Figure 1: Examples of non-guillotine cutting (a), and guillotine cutting (b) patterns.

In this paper we consider the case of d-staged cutting with $d = 2$ (see Figure 2(a)), and we denote it as *2-staged* Two-Dimensional Knapsack (2TDK). **Note that, if a third**

[1]The name Cutting Stock is used when multiple copies of each item have to be cut off, while Bin Packing is the version in which each item is considered somehow separately.

stage of cutting is allowed only to separate an item from a waste area, we call this the *non-exact case* **of 2TDK or 2TDK** *with trimming* **(see, Gilmore and Gomory [7], and again Figure 2(a)). Otherwise, we have the** *exact case* **of 2TDK, or 2TDK** *without trimming* **(see, Figure 2(b)).**

It is easy to see (but it will be shown in detail in Section 3.2) that the restriction introduced takes 2TDK to be suited for the column generation approach of Gilmore and Gomory since now the slave problem turns out to be a One-Dimensional Knapsack. These problems have been addressed in the literature by several authors, and both exact and heuristic algorithms have been proposed for some different variants. In particular, the following problems may occur:

- 2TDK is said to be *Unconstrained* (U-2TDK) if there is no limit to the number of items of each type which can be cut off (apart from the obvious geometric limit). Otherwise, the problem is said to be *Constrained* (C-2TDK);
- if a 90° degree *Rotation* of the items is allowed we refer to R-2TDK; otherwise, we consider the orientation of the items to be *Fixed* (F-2TDK);
- 2TDK is said to be *unweighted* if $\bar{p}_i = \bar{h}_i \bar{w}_i$ $(i = 1, \ldots, m)$; otherwise, if the profit of an item is not equal to its area, the problem is said to be *weighted*.

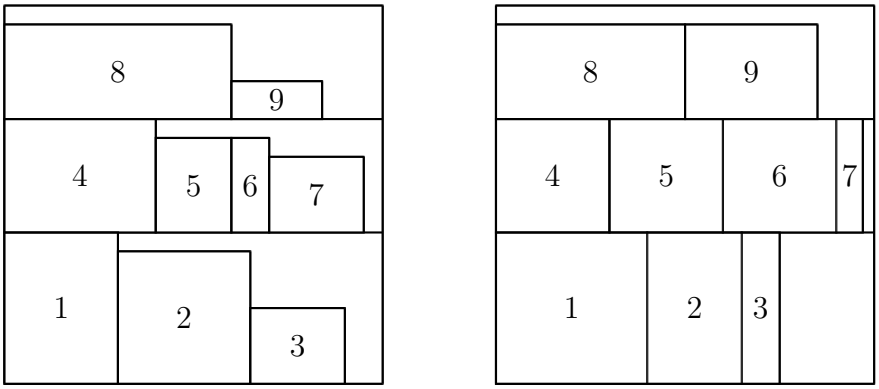

Figure 2: Examples of 2-staged patterns: non-exact (a) and exact (b) cases.

Extensive studies on 2TDK problems have been performed by Hifi [9]. In particular, Hifi and Zissimopoulos [12], Morabito and Garcia [19], and Hifi [10] adapted and extended the approaches proposed by Gilmore and Gomory [7] (i.e., dynamic programming and the already mentioned column generation) to solve many of the variants of 2TDK. Hifi and Roucairol [11] proposed both exact and heuristic algorithms for the specific case FC-2TDK.

The case in which $d = 3$ has been recently faced by Vanderbeck [21] by using nested decomposition within the classical column generation formulation. The

general TDK has been considered by Fekete and Schepers [3, 4] who proposed an elegant graph theory formulation, and a bounding technique based on the definition of "dual-feasible" functions. Both the formulation and the bounding technique can be applied to multi-dimensional cutting problems in general.

In this paper we propose two Integer Linear Programming (ILP) models for 2TDK involving a polynomial number of variables and constraints, which can be easily adapted to all the variants of 2TDK originated by the cases above. In the next section the models are introduced and discussed by considering, for simplicity, the Fixed Constrained version of the problem. **In Section 3 the quality of the continuous relaxations of these models are compared with the classical column generation formulation by Gilmore and Gomory, and with bounds obtained by exploiting appropriate dual-feasible functions. In Section 4 the models are computationally tested on instances from the literature by using the branch-and-bound framework provided by Cplex 6.5.3 enriched by specific linear inequalities aimed at avoiding symmetries and by a simple problem-oriented branching rule.** In Section 5 the extensions to the other cases are presented along with additional computational results on each of them.

In the following we assume that all input data are positive integers.

2 ILP Models for FC-2TDK

Many of the classical heuristic results on Two-Dimensional Packing Problems are obtained by considering the restriction of packing (cutting) the items into *shelves*, i.e., rows forming levels. More precisely, a shelf is a slice of the stock rectangle with width W, and height coincident with the height of the tallest item cut off from it. In addition, each item cut off from the shelf has its bottom edge on a line, the basis of the shelf, and the top of the shelf determines the basis of a following shelf (see, e.g., the slice containing items 4, 5, 6, 7 in Figure 2(a)).

The following simple observation holds.

Observation 1 *Each feasible solution of 2TDK with trimming is composed of shelves, and, vice-versa, each item packed into a shelf can be cut off in at most two stages* **(plus trimming)**.

Recently Lodi, Martello and Vigo [15] and Lodi [13] introduced new models for Two-Dimensional Packing problems in which the restriction of packing into shelves is explicitly considered. In the light of the previous observation, some of the results and the terminology introduced in [13] can be used for 2TDK. In particular, it is also true for 2TDK that for any optimal solution there exists an equivalent solution in which the first (leftmost) item cut in each shelf is the tallest item of the shelf (see again Figure 2(a)). This allows us to consider only solutions which satisfy this condition, and this first item is said to *initialize* the shelf.

2.1 Model 1

For the first model we consider each item to be distinct, i.e., for each shape's type i $(i = 1, \ldots, m)$, we define ub_i identical items j such that $h_j = \bar{h}_i$, $w_j = \bar{w}_i$, and $p_j = \bar{p}_i$. Let $n = \sum_{i=1}^{m} ub_i$ indicate the overall number of items, and consider the items ordered in such a way that $h_1 \geq h_2 \geq \ldots \geq h_n$. The model assumes that n potential shelves may be initialized: **shelf k, if used, must be initialized by item k $(k = 1, \ldots, n)$.** Then, the possible cutting of the n items from the potential shelves is described by the following binary variables:

$$x_{jk} = \begin{cases} 1 & \text{if item } j \text{ is cut from shelf } k \\ 0 & \text{otherwise} \end{cases} \qquad (i = 1, \ldots, m;\ j = 1, \ldots, n) \tag{1}$$

The model is then as follows:

$$\text{M1} \qquad \max \sum_{j=1}^{n} p_j \sum_{k=1}^{j} x_{jk} \tag{2}$$

$$\text{subject to} \qquad \sum_{k=1}^{j} x_{jk} \leq 1 \qquad (j = 1, \ldots, n) \tag{3}$$

$$\sum_{j=k+1}^{n} w_j x_{jk} \leq (W - w_k)\, x_{kk} \quad (k = 1, \ldots, n-1) \tag{4}$$

$$\sum_{k=1}^{n} h_k x_{kk} \leq H \tag{5}$$

$$x_{jk} \in \{0, 1\} \qquad (k = 1, \ldots, n;\ j = k, \ldots, n) \tag{6}$$

The objective function (2) maximizes the sum of the profits of the cut items. Inequalities (3) guarantee that each item is cut at most once, and only from shelves whose height is at least equal to the height of the item. Inequalities (4) assure that the width constraint for each shelf is satisfied, **and that either item k is on shelf k or shelf k is empty**, whereas inequality (5) imposes the height constraint. Note that the meaning of each variable x_{kk} $(k = 1, \ldots, n)$ is twofold: $x_{kk} = 1$ implies that item k is cut from shelf k, i.e., shelf k is used and initialized by its corresponding item.

2.2 Model 2

In the second model the decomposition of the sets of shapes into single items is done only in terms of shelves, i.e., we consider the items with the same shape's type together, whereas we separate them with respect to the initialization of the shelves. Hence, we need to define a mapping between shape's types i $(i = 1, \ldots, m)$, and potential shelves k $(k = 1, \ldots, n)$. In fact, any item of type i may be cut from shelves in the range $[1, \sum_{s=1}^{i} ub_s]$, and we define $\alpha_i = \sum_{s=1}^{i} ub_s$ $(i = 1, \ldots, m)$ with $\alpha_0 = 0$. On the other hand, any shelf k can be used to obtain items whose type is in the range $[\beta_k, m]$, with $\beta_k = \min\{r : 1 \leq r \leq m, \alpha_r \geq k\}$

$(k = 1, \ldots, n)$. **Thus, β_k $(k = 1, \ldots, n)$ denotes the shape's type of the item which must initialize shelf k.** By assuming again $\bar{h}_1 \geq \bar{h}_2 \geq \ldots \geq \bar{h}_m$, we have two separate sets of variables. The first set is composed of the following integer (non-binary) variables:

$$x_{ik} = \begin{cases} \text{number of items of type } i \text{ cut from shelf } k & \text{if } i \neq \beta_k \\ \text{number of } \textbf{additional} \text{ items of type } i \text{ cut from shelf } k & \text{if } i = \beta_k \end{cases} \tag{7}$$

where $i = 1, \ldots, m;\ k \in [1, \alpha_i]$, and the term "additional" indicates that the item of type i initializing shelf k is separately considered (if the shelf corresponds to this type of items).

The second set involves the following binary variables:

$$q_k = \begin{cases} 1 & \text{if shelf } k \text{ is used} \\ 0 & \text{otherwise} \end{cases} \qquad (k = 1, \ldots, n) \tag{8}$$

The model is then as follows:

$$\text{M2} \quad \max \sum_{i=1}^{m} \bar{p}_i \Big(\sum_{k=1}^{\alpha_i} x_{ik} + \sum_{k=\alpha_{i-1}+1}^{\alpha_i} q_k\Big) \tag{9}$$

$$\text{subject to} \quad \sum_{k=1}^{\alpha_i} x_{ik} + \sum_{k=\alpha_{i-1}+1}^{\alpha_i} q_k \leq ub_i \qquad (i = 1, \ldots, m) \tag{10}$$

$$\sum_{i=\beta_k}^{m} \bar{w}_i x_{ik} \leq (W - \bar{w}_{\beta_k}) q_k \qquad (k = 1, \ldots, n) \tag{11}$$

$$\sum_{k=1}^{n} \bar{h}_{\beta_k} q_k \leq H \tag{12}$$

$$\sum_{s=k}^{\alpha_i} x_{is} \leq ub_i - (k - \alpha_{i-1}) \qquad (i = 1, \ldots, m;\ k \in [\alpha_{i-1} + 1, \alpha_i]) \tag{13}$$

$$0 \leq x_{ik} \leq ub_i \quad \text{integer} \qquad (i = 1, \ldots, m;\ k \in [1, \alpha_i]) \tag{14}$$

$$q_k \in \{0, 1\} \qquad (k = 1, \ldots, n) \tag{15}$$

The objective function (9) corresponds to the one of Model 1, so as inequalities (10), (11), and (12) which impose the cardinality constraints, the width constraints, and the height constraint, respectively. **The purpose of inequalities (13), which are redundant in terms of ILP formulation, is to strengthen the bound on the x_{ik} variables (given by inequalities (14)). Indeed, it is quite easy to see that without inequalities (13) part of the structure obtained in M1 by considering the items having the same shape separately would be lost in M2. For any shape's type i such that $ub_i > 1$, all items of type i can be packed in shelves $k \in [\alpha_{i-1} + 1, \alpha_i]$, whereas in M1 an item j can be packed only in shelves k such that $k \leq j$. (Note that we have a constraint (13) for each potential shelf k $(k = 1, \ldots, n)$, and that these inequalities are useful for improving the LP relaxation of M2, see Section 3.1.)**

Immediate correspondence exists between the q_k variables of Model 2 and the x_{kk} variables $(k = 1, \ldots, n)$ in Model 1, whereas each variable x_{ik} of Model 2 **"cumulates" a**

set of $x_{\ell k}$ variables of Model 1. Formally, by denoting with the apex "^{I}" (resp. "^{II}") the x variables in M1 (resp. M2), a straightforward mapping of the x variables of the two models is:

$$x^{II}_{ik} = \sum_{\ell=\alpha_{i-1}+1}^{\alpha_i} x^I_{\ell k} \quad (\textbf{if } i \neq \beta_k), \qquad x^{II}_{ik} = \sum_{\ell=k+1}^{\alpha_i} x^I_{\ell k} \quad (\textbf{if } i = \beta_k) \tag{16}$$

We conclude Section 2 by briefly discussing the size of M1 and M2 above. It is immediate that M1 involves $n(n+1)/2$ binary variables and $2n$ constraints, whereas M2 involves $2n + m + 1$ constraints, n binary variables, and $\sum_{i=1}^{m}\sum_{s=1}^{i} ub_s$ integer variables. This means that the number of integer variables of M2 depends on the structure of the instance, and, in particular, belongs to the range $[n, n(n+1)/2]$, where the lower bound of n variables corresponds to the case in which all the items are identical ($m = 1$), whereas the upper bound is given by the case in which all items are different ($m = n$).

3 Upper Bounds

In this section we consider upper bound procedures for 2TDK and we computationally compare them in Section 3.4.

3.1 Upper Bounds from the LP relaxations of the Models

Valid upper bounds for 2TDK can be obtained by solving the LP relaxation of both M1 and M2, i.e., by replacing each constraint (6) (resp. (15)) with $0 \leq x_{jk} \leq 1$ (resp. $0 \leq q_k \leq 1$), and by relaxing for M2 the integrality requirement in constraints (14).

As anticipated in Section 2.2, inequalities (13) are useful in terms of LP relaxation. Indeed, M2 without inequalities (13) is allowed to split an item into one part initializing the shelf (q part) and some others which can be packed as "additional" parts (x parts) in the same shelf or in the following shelves of that type (if any). Hence, the profit of the item is possibly taken into account completely (see the objective function (9)), while the height of the corresponding shelf is only partially paid (see constraint (12)). This is not the case for M1, since if an item initializes a shelf it cannot be packed as "additional" neither in it nor in the following shelves.

The following example shows the effectiveness of inequalities (13) in overcoming the above drawback.

Example 1
We are given the simple instance: $H = W = 10$, $m = 2$, $\bar{h}_1 = 10$, $\bar{w}_1 = 1$, $\bar{p}_1 = 1000$, $ub_1 = 3$, $\bar{h}_2 = \bar{w}_2 = 1$, $\bar{p}_2 = 1$, $ub_2 = 100$.
The LP relaxation of M1 gives an upper bound $U^I_M = 3007$, whereas, without inequalities (13), $U^{II}_M = 3070$ corresponding to the solution $q_1 = 0.3$, $x_{11} = 2.7$, plus 7 shelves full of

items of shape's type 2. This solution could be forbidden by using an obvious strengthening of (14): $0 \leq x_{ik} \leq ub_i - 1$ if $\beta_i = k$, which corresponds, in this case, to $x_{11} \leq 2$. However, this could be not enough: an equivalent solution is $q_1 = 0.2$, $x_{11} = 1.8$, $q_2 = 0.1$, $x_{12} = 0.9$ plus 7 shelves full of items of shape's type 2. By imposing the three inequalities (13) (which dominate by the lifting operation the strengthening above) $x_{11} + x_{12} + x_{13} \leq 2$, $x_{12} + x_{13} \leq 1$, $x_{13} \leq 0$, we obtain $U^{II}_{M2} = 3007$. □

Due to the above discussion, the following theorem holds.

Theorem 1 *M1 and M2 are equivalent in terms of continuous relaxation.*

Proof. It is enough to show that to each solution of the continuous relaxation of M1 ($M1_c$ in the following) corresponds a solution of the continuous relaxation of M2 ($M2_c$ in the following) with the same value, and vice-versa. **In order to distinguish the x variables of M1 and M2, we use the same notation introduced in Section 2.2.**
M1 $\rightarrow$ M2. Given a feasible solution $M1_c$, a corresponding solution $M2_c$ is built as follows:

for $k := 1$ **to** n $\quad q_k := x^I_{kk}$;
for $i := 1$ **to** m
  for $k := 1$ **to** α_i
    if $(i \neq \beta_k)$ **then** $x^{II}_{ik} := \sum_{j=\alpha_{i-1}+1}^{\alpha_i} x^I_{jk}$
    else $x^{II}_{ik} := \sum_{j=k+1}^{\alpha_i} x^I_{jk}$

and this solution, with exactly the same value, **satisfies inequalities (13) by construction. Hence, recalling the discussion of Section 2.2, in $M2_c$ the "additional" items of shape's type i cut off from a shelf $k \in [\alpha_{i-1}+1, \alpha_i]$ (i.e., $\beta_k = i$) are those items $j > k$ (and, obviously, such that $\beta_j = i$) cut off from shelf k in $M1_c$.**
M2 $\rightarrow$ M1. Given a feasible solution $M2_c$, a corresponding solution $M1_c$ is built as follows:

for $k := 1$ **to** n
  $x^I_{kk} := q_k$;
  for $i := \beta_k$ **to** m $\quad z_i := x^{II}_{ik}$
  for $j := k + 1$ **to** n
    $x^I_{jk} := \min\{1, z_{\beta_j}\}$
    $z_{\beta_j} := z_{\beta_j} - x^I_{jk}$

where z_i $(i = 1, \ldots, m)$ is, by construction, the number of "additional" items of shape's type i cut from the current shelf (the one defined by the outer loop). The solution obtained with the algorithm above has the same value of the original $M2_c$ and is feasible for the continuous relaxation of M1 since inequalities (13) assure that, given a shelf k such that $i = \beta_k$ $(k = 1, \ldots, n)$, no item j of shape's type i can be cut off from k

if $j < k$. $\square$

A final remark concerns the *geometric bound*, i.e., the most obvious upper bound for general Two-Dimensional Knapsack problems obtained by solving the *continuous relaxation* of the *0-1 Knapsack Problem* in which the capacity of the knapsack is $W \cdot H$, and to each item j ($j = 1, \ldots, n$) corresponds an item of KP01 whose profit is p_j and whose weight is $w_j \cdot h_j$. Not surprisingly, by denoting the value of this continuous relaxation bound as U_g, and the value of the continuous relaxation of the models as U_M, the following simple result can be proved with algebraic arguments (see Monaci [18]):

Proposition 1 ***For any instance of FC-2TDK,*** $U_M \leq U_g$***.***

3.2 The Column Generation Upper Bound *** new ***

A classical upper bound for FC-2TDK can be derived from the column generation approach proposed by Gilmore and Gomory [6]. Let $\mathcal{S} := \{1, \ldots, NS\}$ be the set of all "feasible" shelves, i.e., those shelves s satisfying the condition:

$$\sum_{i=1}^{m} \bar{w}_i r_i^s \leq W$$

where r_i^s ($0 \leq r_i^s \leq ub_i,\ i = 1, \ldots, m$) denotes the number of items with shape's type i which are cut from shelf s. Let A_s and P_s be the height and the profit of shelf s, i.e., formally:

$$A_s := \max\{\bar{h}_i : r_i^s > 0,\ i = 1, \ldots, m\} \quad \text{and} \quad P_s := \sum_{i=1}^{m} \bar{p}_i r_i^s$$

Thus, the LP relaxation of the column generation formulation for 2TDK (the so-called *master* problem) is as follows:

$$U_{CG} := \max \sum_{s \in \mathcal{S}} P_s y_s \tag{17}$$

$$\text{subject to} \quad \sum_{s \in \mathcal{S}} r_i^s y_s \leq ub_i \quad (i = 1, \ldots, m) \tag{18}$$

$$\sum_{s \in \mathcal{S}} A_s y_s \leq H \tag{19}$$

$$y_s \geq 0 \qquad (s \in \mathcal{S}) \tag{20}$$

Since the number NS of variables (columns) is in general huge, only a subset of them is considered, thus "restricted" LP relaxations are iteratively solved until no variable with negative reduced cost exists, i.e.

$$P_s \leq \rho A_s + \sum_{i=1}^{m} \pi_i r_i^s \quad (\forall s \in \mathcal{S}) \tag{21}$$

where π_i $(i = 1, \ldots, m)$ and ρ are the dual variables associated with constraints (18) and (19), respectively. Columns with negative reduced cost, violating condition (21), can be detected by solving the so-called *slave* problem, i.e., the following *Bounded Knapsack Problem* (BKP):

$$Z(BKP) := \max \sum_{i=1}^{m} (\bar{p}_i - \pi_i) r_i \tag{22}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \bar{w}_i r_i \leq W \tag{23}$$

$$0 \leq r_i \leq ub_i \quad \text{integer} \quad (i = 1, \ldots, m) \tag{24}$$

If there exists a feasible shelf s such that $Z(BKP) > \rho A_s$, then s is added to the formulation (17)–(20), otherwise U_{CG} represents the value of the upper bound of the column generation formulation.

Since we are just interested at the quality of the bound, we implemented the above column generation approach by using Cplex for solving both the master problem and the slave problem. For the description of an efficient implementation of column generation techniques for cutting and packing, the reader is referred to Vanderbeck [20, 21].

Finally, it is easy to see that the presented technique can be adapted to all the variants of 2TDK by modifying the slave problem, i.e., the way the columns are generated.

3.3 Upper Bounds from dual-feasible functions *** new ***

Recently, Fekete and Schepers [4] proposed a general bounding technique for cutting and packing problems by using the concept of *dual-feasible functions* (see also Lueker [16]), and exploited it for solving multi-dimensional knapsack problems.

Roughly speaking, a dual-feasible function u is a function mapping a given instance I of the problem into a new instance I' such that any feasible solution for I is also feasible for I': hence, any bound for I' is also valid for I. Fekete and Schepers [4] proposed a set of mapping functions which can be used to produce different I', thus different valid bounds: after this mapping, they simply compute the *geometric bound* (see Section 3.1) on I'.

It is known that any function u described above is a *feasible* solution of the *dual* of the column generation formulation of the problem (see the previous section). This fact implies the name "dual-feasible" function, and the dominance of U_{CG} with respect to the bounds which can be obtained from any function u.

We used the three dual-feasible functions proposed in [4] to produce modified instances of FC-2TDK. However, since the bound given by the LP relaxation of M1 and M2 dominates the geometric bound (recall Proposition 1), we applied the LP relaxation of our models as bounding procedure on the modified instances. The results were not satisfying since we were not able to improve the bound given by M1 (and M2) on the original instances. This behavior seems to be due to the structure of the instances (see Section 3.4) in which there is a large number of "small" items, i.e., items whose height (resp. width) is small with respect to the height (resp. width) of the stock unit. The dual-feasible functions in

[4] are known to be effective in case of "large" items. However, better results could be in principle obtained by using specific dual-feasible functions taking into account the 2-stage constraint.

Since no improvement has been obtained, detailed computational results using dual-feasible functions have been omitted in the next section.

3.4 Upper Bound comparison *** new ***

We compared the column generation bound and the bound given by the Models described in Section 2 by considering the set of 38 FC-2TDK instances[2] proposed by Hifi and Roucairol [11] and composed of 14 weighted instances and 24 unweighted ones. The results are given, separately for weighted and unweighted instances, in Table 1. We need to note that the literature of 2TDK distinguishes two cases: (i) the case in which the first cut is horizontal, i.e., the case we referred to in the entire paper, and (ii) the case in which the first cut is vertical. In the latter case, we can simply extend the terminology of shelf patterns by referring to "shelves" also as "columns" forming levels (see the discussion at the beginning of Section 2).

In particular, for each instance of the discussed sets, Table 1 indicates the identifier (ID), the number of shape's types (m), and the number of items (n). Moreover we report, for both horizontal and vertical patterns (cases (i) and (ii), respectively), the optimal solution value (Opt), the ratio between the bound produced by the continuous relaxation of the models (U_M) and the optimal solution value, and the ratio between the bound produced by the column generation (U_{CG}) and the optimal solution value[3].

The computational results show that the bound produced by the column generation is better than the one produced by the LP relaxations of our models on almost all the instances. This fact is not surprising since it descends from a general result by Geoffrion [5]. Specifically, the average percentage gap of U_M for the weighted instances is 11.5% (12.8% resp.) for the horizontal (vertical resp.) pattern with respect to 3.8% (4.0% resp.) of U_{CG}. For the unweighted instances, instead, these gaps are 5.8% (6.5% resp.) with respect to 2.8% (2.1% resp.).

These results show a quite large difference between U_M and U_{CG}, mainly in the weighted case. However, as will be shown in Section 4.2, these initial gaps of M1 and M2 are efficiently closed within a branch-and-bound framework. Moreover, we have to point out that the computation of U_{CG} requires the solution of a set of (NP-hard) BKPs, thus the elapsed time is exponential in m.

[2]Available at `ftp://panoramix.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/`.

[3]Both bounds are rounded down to the closest integer before computing their ratio with respect to the optimal value.

Table 1: FC-2TDK: Models' bound vs. Column Generation bound.

Instance			First Cut is Horizontal			First Cut is Vertical		
ID	m	n	Opt	$\frac{U_M}{Opt}$	$\frac{U_{CG}}{Opt}$	Opt	$\frac{U_M}{Opt}$	$\frac{U_{CG}}{Opt}$
HH	5	18	10689	1.164	1.017	9246	1.333	1.080
2	10	23	2535	1.135	1.045	2444	1.141	1.051
3	20	62	1720	1.166	1.101	1740	1.145	1.088
A1	20	62	1820	1.180	1.032	1820	1.170	1.094
A2	20	53	2315	1.171	1.033	2310	1.161	1.057
STS2	30	78	4450	1.073	1.018	4620	1.027	1.000
STS4	20	50	9409	1.057	1.018	9468	1.048	1.030
CHL1	30	63	8360	1.091	1.049	8208	1.114	1.007
CHL2	10	19	2235	1.106	1.017	2086	1.187	1.016
CW1	25	67	6402	1.117	1.072	6402	1.123	1.007
CW2	35	63	5354	1.123	1.034	5159	1.162	1.091
CW3	40	96	5287	1.128	1.064	5689	1.060	1.028
Hchl2	35	75	9630	1.056	1.012	9528	1.069	1.008
Hchl9	35	76	5100	1.040	1.019	5060	1.049	1.003
2s	10	23	2430	1.152	1.046	2450	1.143	1.051
3s	20	62	2599	1.077	1.033	2623	1.067	1.066
A1s	20	62	2950	1.017	1.014	2910	1.031	1.009
A2s	20	53	3423	1.052	1.015	3451	1.043	1.023
STS2s	30	78	4569	1.023	1.010	4625	1.011	1.009
STS4s	20	50	9481	1.034	1.017	9481	1.034	1.011
OF1	10	23	2713	1.032	1.000	2660	1.053	1.006
OF2	10	24	2515	1.113	1.081	2522	1.110	1.071
W	20	62	2623	1.067	1.066	2599	1.077	1.033
CHL1s	30	63	13036	1.013	1.012	12602	1.047	1.006
CHL2s	10	19	3162	1.078	1.046	3198	1.066	1.000
A3	20	46	5380	1.041	1.040	5403	1.036	1.009
A4	20	35	5885	1.071	1.037	5905	1.067	1.010
A5	20	45	12553	1.052	1.050	12449	1.060	1.015
CHL5	10	18	363	1.102	1.044	344	1.163	1.108
CHL6	30	65	16572	1.020	1.020	16281	1.038	1.012
CHL7	35	75	16728	1.010	1.005	16602	1.018	1.005
CU1	25	82	12312	1.015	1.015	12200	1.025	1.012
CU2	35	90	26100	1.006	1.006	25260	1.039	1.038
Hchl3s	10	51	11961	1.041	1.015	11829	1.052	1.018
Hchl4s	10	32	11408	1.091	1.037	11258	1.106	1.015
Hchl6s	22	60	60170	1.026	1.016	59853	1.031	1.015
Hchl7s	40	90	62459	1.015	1.012	62845	1.009	1.004
Hchl8s	10	18	729	1.241	1.051	791	1.200	1.037

4 Computational Experiments

We tested models M1 and M2 on the instances introduced in Section 3.4 by using the standard branch-and-bound (without specific tuning of the parameters) of the commercial ILP solver Cplex version 6.5.3. The code runs on a Digital Alpha 533 Mhz.

We performed two sets of computational experiments: **the first is aimed at improving the performance of the models within the branch-and-bound framework, whereas the second compares the models with the existing literature. In Section 4.1 we propose linear inequalities aimed at removing symmetric solutions from the search space, and their effectiveness is computationally shown. Moreover, a simple improvement to the standard branching rule of Cplex 6.5.3 is presented and tested in the same section.**

In the second set of experiments (see Section 4.2) we compare the results obtained by M1 and M2 (plus the effective linear inequalities and the branching rule) with those of the best exact algorithm in the literature by Hifi and Roucairol [11].

4.1 Avoiding Symmetries by Linear Inequalities

Symmetries are one of the main problems when dealing with exact methods for cutting problems. **This is true in general for ILP models (e.g., Hadjiconstantinou and Christofides [8]) when they are not based on the column generation framework (see, instead, Vanderbeck [20]), and also for branch-and-bound methods exploiting combinatorial bounds (e.g., Martello and Vigo [17]), the only exception being the graph theory modeling approach of Fekete and Schepers [3].**

In our models some of these symmetries are avoided by the assumptions discussed at the beginning of Section 2: we do not explicitly distinguish the position of an item in a shelf (neither the **position of a shelf in the rectangular stock**), and the fact that n potential shelves are considered (each of them with a prefixed height) **allows to further restrict the search space**.

In addition, some other symmetries can be avoided in both models by using two sets of linear inequalities which are presented in the following for M1, and then adapted to M2.

The first set of inequalities is aimed at removing from the search space the equivalent solutions in which a shelf k is used instead of a shelf ℓ, both corresponding to items with the same shape's type, and such that $k > \ell$. We call these inequalities *Ordering Inequalities* (OIs), and, **by using the α-notation introduced for M2 (see Section 2.2), we prove their correctness through the following theorem.**

Theorem 2 *For any shape's type i ($i = 1, \ldots, m$), linear inequalities*

$$x_{tt} \geq x_{t+1,t+1} \quad (t \in [\alpha_{i-1} + 1, \alpha_i - 1]) \tag{25}$$

do not change the optimal solution **value** *of M1.*

Proof. Suppose that in an optimal solution of 2TDK one of the inequalities (25), say $x_{kk} \geq x_{k+1,k+1}$, is violated. This implies that item k has been cut off from one of the shelves $1, \ldots, k-1$, say s, whereas item $k+1$ initializes its shelf. However, since k and $k+1$ are identical, it is always possible to **"swap" shelves k and $k+1$ by: (i) cutting item $k+1$ from s, (ii) initializing shelf k with its corresponding item, and (iii) obtaining from shelf k the items which were cut from shelf $k+1$.** Hence, we obtain an equivalent optimal solution such that inequality $x_{kk} \geq x_{k+1,k+1}$ is satisfied. □

The second set of inequalities concerns the number of additional items of a given shape's type, say i, which can be cut from shelves, say ℓ and k, such that (by using the β-notation introduced for M2) $\beta_\ell = \beta_k = i$. In particular, we want to remove from the search space those solutions in which the number of these items cut from shelf k is greater than the number of those cut from shelf ℓ if $k > \ell$. We refer to these inequalities as *Extended* Ordering Inequalities (EOIs), and their correctness is proved by the following theorem.

Theorem 3 *For any shape's type i ($i = 1, \ldots, m$), linear inequalities*

$$\sum_{s=t+1}^{t+ub_i-1} x_{st} \geq \sum_{s=t+2}^{t+ub_i-1} x_{s,t+1} \quad (t \in [\alpha_{i-1}+1, \alpha_i - 1]) \tag{26}$$

do not change the optimal solution **value** *of M1.*

Proof. The swapping argument in the proof of Theorem 2 can be easily adapted. Suppose that in an optimal solution of 2TDK one of the inequalities (26), say the one corresponding to shelves k and $k+1$, is violated. Since the corresponding items k and $k+1$ have the same shape's type, there is an equivalent solution in which the items cut from shelf k are obtained from shelf $k+1$ and vice-versa, so as to satisfy the violated inequality. □

Corresponding OIs and EOIs can be derived for M2: we obtain respectively

$$q_t \geq q_{t+1} \qquad (i = 1, \ldots, m;\ t \in [\alpha_{i-1}+1, \alpha_i - 1]) \tag{27}$$

and

$$x_{it} \geq x_{i,t+1} \qquad (i = 1, \ldots, m;\ t \in [\alpha_{i-1}+1, \alpha_i - 1]) \tag{28}$$

whose correctness is immediate from Theorems 2 and 3 above.

Both Ordering Inequalities and Extended Ordering Inequalities can be added to the models in the construction phase ($O(n) + O(n)$ inequalities in both cases), and their effectiveness is computationally tested on the same set of instances introduced in Section 3.4. **Table 2 reports computational results comparing M1 and M2 in their basic version and with the addition of OIs and EOIs.** The results are given separately for weighted and unweighted instances, and, for each model both in basic and strengthened versions, the average computing time (in seconds) and the number of unsolved instances within the time limit of 1,800 seconds are reported. Recalling that each of the instances

Table 2: FC-2TDK: Effectiveness of OIs and EOIs for M1 and M2.

Instances	M1				M2			
	without OIs,EOIs		with OIs,EOIs		without OIs,EOIs		with OIs,EOIs	
type	time	unsolved	time	unsolved	time	unsolved	time	unsolved
weighted	38.97	-	8.66	-	121.27	1	67.75	-
unweighted	307.83	6	38.55	-	319.91	6	70.10	-

is solved twice (first cut either horizontal or vertical), the entries of the table consider 28 and 48 different runs for the weighted and unweighted cases, respectively.

The results in Table 2 show that the addition of linear inequalities in order to avoid symmetries in the models is very effective by allowing the solution of the entire set of instances. Moreover, since we considered as solution time for the unsolved instances the time limit, the average computing times are an optimistic estimate, thus the already relevant speed up shown by the table could be even higher. From now on, we consider the version of M1 and M2 with the addition of OIs and EOIs.

We conclude the section by briefly mentioning the standard branching strategy implemented by Cplex 6.5.3 and by proposing a simple problem-oriented improvement. In order to show that the proposed models are effective for a general-purpose ILP solver, we did not tune Cplex, so that, for example, we leave Cplex automatically choose up/down branch direction and backtracking tolerance. A simple and quite effective improvement can be, however, achieved by defining a priority order for the variables to branch on. Specifically, we assign a higher priority to the variables corresponding to the initialization of a shelf (x_{kk} in M1 and q_k in M2). For each of these variables, the priority value is set to the value of the profit of the corresponding item, thus giving larger priority to items with larger profit.

Table 3 reports the results of this simple branching rule ("improved") compared with the automatic variable selection ("default") implemented by Cplex 6.5.3. The table is organized as Table 2 where columns indicating the number of unsolved instances are replaced by columns indicating the average number of branch-and-bound nodes.

Table 3: FC-2TDK: A simple problem-oriented branching rule.

Instances	M1				M2			
	default		improved		default		improved	
type	time	nodes	time	nodes	time	nodes	time	nodes
weighted	8.66	329.6	9.00	361.9	67.75	6538.2	51.80	5567.0
unweighted	38.55	4877.3	23.61	2333.9	70.10	7047.7	42.29	5023.5

The results in Table 3 suggest that the proposed ordering is quite effective

in the selection of the branching variable, by effecting in an overall reduction of both computing times and number of branch-and-bound nodes.

4.2 Literature comparison

In Table 4 we report the computational comparison of our models (improved as shown in the previous section) with respect to the exact dynamic programming approach proposed by Hifi and Roucairol [11]. **We consider again separately the weighted case (the first 14 instances) and unweighted one (the last 24 instances). For each instance of the discussed sets and for both horizontal and vertical patterns, the computing times of M1 (t.M1), M2 (t.M2), and the one of the algorithm [11] (t.HR) are reported.** The computing times of algorithm HR are expressed in CPU seconds on a UltraSparc10 250 Mhz.

The results in Table 4 show that our models are competitive with the exact approach of Hifi and Roucairol [11]. In fact, both models are able to solve in reasonable computing times all the instances proposed in [11] where the optimality of the solution of instance Hchl2 was not proved by Hifi and Roucairol. In particular, M2 is very effective when the ratio n/m is high, i.e., if the average number of items of each shape's type is relevant, see e.g., instances Hchl3s and Hchl4s. Conversely, when only two or three items on average have the same shape's type, then M1 is faster, see e.g., instances Hchl2 and CHL6. This behavior could be exploited in order to select the more suitable model according to the structure of a given instance.

5 Models' extensions

In the previous sections we discussed the Fixed Constrained version of 2TDK, and in particular the non-exact case. An immediate extension to the exact case is possible for both models: in M1 (resp. M2) it is enough to set $x_{jk} = 0$ if $h_k \neq h_j$ (resp. $\bar{h}_k \neq \bar{h}_{\beta_j}$), or more efficiently not to define those variables.

In the following sections we consider the extensions needed by the models in order to address the most common variants of 2TDK discussed in Section 1. **Both M1 and M2 are considered with the addition of the symmetry-breaking inequalities and with the simple branching rule discussed in Section 4.1.**

Note that, without loss of generality, we address in the following sections only the case in which the first cut is horizontal.

5.1 The *unconstrained* case

Both models can be easily extended to the case in which $ub_i = +\infty$ $(i = 1, \ldots, m)$, i.e., no upper bound on the number of items of each shape's type is defined. This case, called *unconstrained* in Section 1, has been considered in the literature, e.g., [9], and can be addressed by simply observing that such an upper bound always exists due to "shape"

Table 4: FC-2TDK: M1 and M2 vs. Hifi and Roucairol [11].

Instance	First Cut is Horizontal			First Cut is Vertical		
ID	t.M1	t.M2	t.HR	t.M1	t.M2	t.HR
HH	0.28	0.05	0.20	0.25	0.13	0.30
2	0.35	0.32	2.90	1.33	0.42	3.80
3	0.35	0.23	0.20	1.35	0.72	0.20
A1	0.88	0.40	0.20	1.30	0.77	0.30
A2	1.33	0.58	0.80	1.37	0.75	0.70
STS2	16.82	15.27	5.60	1.25	0.65	0.50
STS4	11.42	9.98	9.20	4.88	6.00	1.90
CHL1	8.30	4.00	610.10	6.03	9.80	1496.30
CHL2	0.12	0.13	0.10	0.22	0.25	0.10
CW1	2.32	0.82	1.00	1.00	0.57	1.00
CW2	0.87	0.78	2.10	1.82	1.42	2.20
CW3	2.55	1.72	6.40	7.48	1.37	6.30
Hchl2	61.77	300.02	7200.00	93.80	1674.82	7200.00
Hchl9	3.62	1.90	858.00	6.13	5.12	1017.40
2s	0.48	0.43	4.60	0.65	0.35	4.80
3s	0.33	0.25	0.10	0.75	0.45	0.10
A1s	0.27	0.47	0.10	1.08	0.48	0.20
A2s	2.57	0.77	0.20	3.08	0.73	0.20
STS2s	10.12	11.85	1.10	4.42	1.90	0.80
STS4s	13.10	15.25	8.90	5.42	13.83	1.90
OF1	0.07	0.05	0.10	0.07	0.07	0.10
OF2	0.28	0.22	0.10	0.20	0.15	0.10
W	0.75	0.52	0.20	0.35	0.18	0.10
CHL1s	4.30	5.15	6.10	11.42	80.22	21.50
CHL2s	0.18	0.17	0.10	0.18	0.13	0.10
A3	1.78	1.87	0.30	1.68	3.25	0.40
A4	1.58	1.85	1.60	1.87	2.12	2.10
A5	3.97	1.53	2.40	8.20	3.18	2.50
CHL5	0.03	0.03	0.10	0.05	0.05	0.10
CHL6	21.50	38.52	38.20	21.43	720.00	33.40
CHL7	54.23	181.73	44.60	43.35	475.50	50.60
CU1	11.78	1.70	1.00	448.50	1.93	1.00
CU2	3.67	1.80	2.70	11.02	4.40	2.80
Hchl3s	312.93	13.97	15.30	119.42	2.22	20.40
Hchl4s	402.13	5.62	507.40	3.20	1.10	268.40
Hchl6s	19.60	45.25	14.80	37.13	139.38	7.20
Hchl7s	168.20	751.40	96.20	58.25	907.13	39.10
Hchl8s	0.72	0.42	26.40	0.42	0.28	28.70

reasoning. The simplest way of computing such a bound is to consider for each shape's type i ($i = 1, \ldots, m$), $ub_i = \lfloor (HW)/(\bar{h}_i \bar{w}_i) \rfloor$, i.e., the geometric upper bound computed as area of the rectangular stock divided by the area of the shape's type. A slight improvement can be obtained by considering $ub_i = \lfloor H/\bar{h}_i \rfloor \cdot \lfloor W/\bar{w}_i \rfloor$. This leads to a mapping of an unconstrained instance to a constrained one, thus immediately allowing our models to be used without any change.

The drawback of this mapping for our models is obvious: since the bound we compute is not tight, the size of the models grows up very quickly both in terms of number of variables and constraints. This fact dramatically decreases the effectiveness of M1 while M2 is still usable in practice since the items of the same type are considered together at least as x variables. We computationally verified this behavior by considering the classes of instances[4] discussed by Hifi [10], and some results are reported in Table 5.

Table 5: FU-2TDK: Performance of M1 and M2 on the Unconstrained variant.

Instance					M1				M2			
ID	m	n	Opt	$\frac{U_M}{Opt}$	nv	nc	t.M1		nv	nc	t.M2	
BW	32	1214	2307817	1.088	737505	4792	1800.00	(+)	11019	4825	679.48	
HZ2	5	73	7934	1.178	2701	282	10.70		247	288	1.73	
MW1	10	73	3882	1.085	2701	272	3.08		415	283	0.35	
UW1	25	112	6036	1.273	6328	398	14.05		1197	424	1.02	
W1	20	610	161424	1.077	186355	2400	1800.00	(16.07)	5232	2421	51.50	
B	32	1214	8997780*	1.000	737505	4792	1800.00	(+)	11019	4825	1800.00	(3.37)
H	5	87	12132	1.026	3828	338	1800.00	(0.75)	278	344	2.30	
HZ1	6	257	5226	1.000	33153	1016	154.48		785	1023	8.03	
U1	10	476	22167051	1.015	113526	1884	1800.00	(7.04)	3030	1895	1031.35	
U2	10	189	19967604	1.029	17955	736	1800.00	(1.46)	1076	747	27.18	
UU1	25	93	240346	1.040	4371	322	25.22		1054	348	3.95	

(+) Upper bound undetermined within the given time limit.

* **8997780 is the optimal solution value as computed in [10].**

We considered as an example a single instance for each class reported in [10], and we tested both M1 and M2 as discussed in Section 4 for a given time limit of 1,800 seconds on a Digital Alpha 533 MHz[5].

In Table 5 we report, for each instance, its identifier (ID), the number of shape's type (m), and the number of items after the mapping (n). In addition, we report the optimal solution value (Opt), and the ratio between the upper bound of the continuous relaxation (U_M) and Opt. Then, for each model, the number of variables and constraints (nv and nc respectively), and the computing time (t.M1 and t.M2 respectively) are reported. Finally, in round brackets, we report the percentage gap between lower and upper bound where the instance is not solved to optimality within the time limit (directly reported by Cplex).

[4]Available at `ftp://panoramix.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/`.

[5]In Table 5 and in the other ones of Section 5 the first set of instances refers to the weighted case, and the second set to the unweighted one.

Table 5 does not report results for the instances of the class LW and LU which are unsolvable for our model: the smallest weighted instance of this class, LW1, involves after the mapping 102748 items (so as the corresponding unweighted one LU1), thus the size of both models becomes intractable.

These results show that both models **remain able to solve in a reasonable amount of computing time** a large set of unconstrained instances of 2TDK, **thus testifying the flexibility of the proposed framework.** In particular, M2 greatly benefits from the fact of considering items of the same shape's type partially together.

However, it is important to point out that for pure *unconstrained* **2TDK, i.e., FU-2TDK, ILP models are far from being the state of the art technique. Indeed, FU-2TDK is solvable by computing the optimal solution of a set of $m + 1$ disjoint unconstrained knapsack problems: one for each shape's type, m, and one for cutting the shelves from the stock unit. Moreover, as already proposed by Gilmore and Gomory [7], these knapsack problems can be solved in a single step by dynamic programming. Effective implementations (also for FU-3TDK) of this framework have been proposed by Hifi [10] obtaining results which are around 2 orders of magnitude better than those in Table 5.**

5.2 The *rotation* case

The second variant of 2TDK we consider is the one in which the items can be rotated by 90°, called *rotation* in Section 1. The basic idea for extending both models to this case is to introduce for each possible item, say j, a "companion" item, say j', for which the height and weight values are swapped, and then add to the formulation suitable constraints avoiding that j and j' are both inserted.

More formally, since in M1 each item is considered separately, for each item j ($j = 1, \ldots, n$) the companion item $j + n$ is considered such that $h_{j+n} = w_j$ and $w_{j+n} = h_j$, and the $2n$ items are sorted by non-increasing height values. A straightforward way of imposing the constraints on the packing of the companion items is to replace constraints (3) by

$$\sum_{k=1}^{j} x_{jk} + \sum_{k=1}^{\theta_j} x_{\theta_j,k} \leq 1 \quad (j = 1, \ldots, 2n; j < \theta_j) \tag{29}$$

where θ_j ($j = 1, \ldots, 2n$) indicates the companion of item j after the sorting and renumbering phase.

However, it is not possible to use these constraints in combination with the OIs. Consider, for example, two identical items j and $j + 1$, and their companions (after sorting) θ_j and θ_{j+1}, and suppose that in the optimal solution two shelves must be initialized with height h_j and w_j. The associated OIs force the opening of shelves j (j **being the first item of height** h_j) and θ_j (θ_j **being the first item of rotated height** w_j), whereas this is forbidden by a corresponding constraint (29) **since j and θ_j are the same item**.

A simple way of overcoming this problem is to **relax in a surrogate fashion** constraints (29) in the following way. Let $T = \{1, \ldots, m\}$ be the set of the shape's types, and

let $\gamma_j \in T$ $(j = 1, \ldots, 2n)$ indicate the shape's type to which items j and θ_j belong. We obtain the following m **surrogate** constraints

$$\sum_{j \in J \mid \gamma_j = t} \sum_{k=1}^{j} x_{jk} \leq ub_t \quad \forall\, t \in T \tag{30}$$

where $J = \{1, \ldots, 2n\}$ is the set of all items.

Finally, in the other constraints of M1 we need to replace n with $2n$.

A similar reasoning is applied for M2. In this case the constraints appear naturally in a "**surrogate**" form, and constraints (30) can be written as

$$\sum_{i \in I \mid \delta_i = t} (\sum_{k=1}^{\alpha_i} x_{ik} + \sum_{k=\alpha_{i-1}+1}^{\alpha_i} q_k) \leq ub_t \quad \forall\, t \in T \tag{31}$$

where $I = \{1, \ldots, 2m\}$ is the set of the shape's types either in the original orientation or rotated, and $\delta_i \in T$ $(i \in I)$ indicates the original shape's type corresponding to the new shape type i. Constraints (31) replace constraints (10).

Obviously, the mapping discussed in Section 2.2 and denoted by arrays α and β has to be updated, and again we need to replace n with $2n$.

Both M1 and M2 were tested on a small set of the instances used in sections 3.4 and 4 in which the rotation version of the problem is solved. The results are reported in Table 6 in which for each instance we indicate the identifier (ID), the number of shape's types (m), and the number of items to be cut off (n). Note that in order to solve R-2TDK the instance is almost doubled in size. In addition, for each instance we report the optimal solution value (Opt), the ratio between the upper bound produced by the continuous relaxations of the models (U_M) and the optimal solution value, and the computing times of M1 (t.M1) and M2 (t.M2). In round brackets, we report the percentage gap between lower and upper bound where the instance is not solved to optimality within the time limit.

Table 6 proves that the extension of M1 and M2 are still suitable for the solution of the rotation version of 2TDK. However, the computing times required for instances solved by allowing the items to be rotated are in general much higher than those given in Table 4. **Concerning the behavior of the models, as discussed in Section 4.2, M1 is more effective than M2 when the ratio n/m is small, for example instance Hchl2 ($n/m < 2.2$). Moreover, M1 turns out to be very effective for the overall set of instances in Table 6, and greatly benefits of the new simple branching rule discussed in Section 4.1[6].**

5.3 The *double-Constrained* case

The last variant of 2TDK we consider is the one in which also a lower bound on the number of items of each type is specified, i.e., one is required to cut off at least lb_i items of type

[6]See, Monaci [18] for a comparison of the performance of M1 and M2 with and without the branching rule.

Table 6: RC-2TDK: Performance of M1 and M2 on the Rotation variant.

ID	m	n	Opt	$\frac{U_M}{Opt}$	t.M1	t.M2	
HH	5	18	11301	1.101	0.92	0.32	
2	10	23	2791	1.042	1.10	1.25	
A1	20	62	1980	1.093	3.45	6.22	
STS2	30	78	4610	1.044	24.27	25.62	
CHL1	30	63	8720	1.050	42.50	181.03	
CW1	25	67	6746	1.066	14.88	6.23	
Hchl2	35	75	9921	1.035	311.92	1800.00	(0.45)
2s	10	23	2718	1.030	0.70	0.88	
A1s	20	62	2985	1.005	2.12	2.78	
STS2s	30	78	4659	1.003	16.07	9.13	
OF1	10	23	2713	1.032	0.68	0.37	
W	20	62	2754	1.017	5.13	2.62	
CHL1s	30	63	13164	1.003	15.75	73.97	
A3	20	46	5529	1.013	5.77	17.58	
CHL5	10	18	399	1.002	0.13	0.07	
CU1	25	82	12500	1.000	17.13	4.57	
Hchl3s	10	51	12230	1.018	121.40	245.98	

i ($i = 1, \ldots, m$). This variant, which is denoted by *double-Constrained* (dC-2TDK), has been recently addressed by Cung and Hifi [1] for the more general guillotine TDK problem, and has several practical applications.

Note that finding a feasible solution of dC-2TDK is already NP-complete, since it corresponds to the recognition version of the Two-Dimensional Shelf Bin Packing, which is in turn a generalization of the classical One-Dimensional Bin Packing [15].

Both M1 and M2 can be easily extended to consider this variant. In particular, for M1 we can simply write, for each $i = 1, \ldots, m$, the first lb_i constraints of type (3) as equalities. For M2, instead, we need to add, for each $i = 1, \ldots, m$, a new constraint of the following form:

$$\sum_{k=1}^{\alpha_i} x_{ik} + \sum_{k=\alpha_{i-1}+1}^{\alpha_i} q_k \geq lb_i \tag{32}$$

which assures that at least lb_i items of type i are cut off.

We tested our models on the instances used by Cung and Hifi [1] (kindly provided by the authors), and the results are shown in Table 7. This table contains the same information as the previous ones with an additional column indicating the overall number of items which must be cut off (L), i.e. $L = \sum_{i=1}^{m} lb_i$.

Again, Table 7 shows that the addition of the lower bounding constraint is suitably taken into account by the extensions of M1 and M2, and in fact, the computing times to solve this more constrained variant are considerably smaller than the corresponding ones in Table 4.

Table 7: FdC-2TDK: Performance of M1 and M2 on the double-Constrained variant.

ID	m	n	L	Opt	$\frac{U_M}{Opt}$	t.M1	t.M2
2_borne	10	23	7	2336	1.168	0.15	0.25
A1_borne	20	62	5	1660	1.157	0.30	0.50
CHL1_borne	30	63	10	infeasible		0.37	0.18
CHL3_borne	15	35	10	5283	1.000	0.13	0.10
Hchl1_borne	30	65	12	9744	1.104	0.75	1.02
sts2_borne	30	78	8	4110	1.117	16.05	13.27
2s_borne	10	23	7	2225	1.175	0.13	0.22
A1s_borne	20	62	5	2583	1.138	0.52	0.37
CHL1s_borne	30	63	10	infeasible		0.40	0.53
CHL3s_borne	15	35	10	7402	1.000	0.13	0.20
Hchl1s_borne	30	65	12	14538	1.113	0.88	1.23
Hchl3s_borne	10	51	13	11784	1.053	119.75	9.30
chl7s_borne	35	75	11	15711	1.074	53.25	194.38
sts2s_borne	30	78	8	4199	1.102	9.42	11.67

6 Conclusions *** new ***

The *2-staged* Two-Dimensional Knapsack problem (2TDK) is a classical variant of the Two-Dimensional Knapsack with a long history in the cutting and packing literature and some important real-world applications.

We introduced two new ILP models for the most classical variant of 2TDK, we studied the characteristics of these models and we analyzed their performance in terms of quality of the bound with respect to the classical column generation approach. Moreover, we analyzed the effectiveness of these models within a branch-and-bound framework by comparing them with the most effective combinatorial algorithm in the literature [11]. To this aim we used the standard branch-and-bound algorithm implemented in Cplex 6.5.3 (enhanced by a simple problem-oriented branching rule), and an aggressive policy of addition of linear inequalities in order to avoid symmetries.

Finally, we extended these models to the other variants of 2TDK in the literature by showing that they remain suitable in all of them (with some concerns with respect to the size of the instances and their structure), i.e., that they represent an effective and flexible algorithmic approach for this wide cutting and packing area.

Acknowledgments

We are grateful to Silvano Martello and Daniele Vigo for introducing us to the wide domain of Cutting & Packing. Thanks are also due to Adam N. Letchford and Alberto Caprara who read a preliminary version of the paper, and to the anonymous referees whose tight remarks consistently improved, in our view, the paper. We acknowledge the support given to this project by the Consiglio Nazionale delle Ricerche (CNR), Italy.

References

[1] V.-D. Cung and M. Hifi. Handling lower bound constraints in 2D guillotine cutting. Talk presented at ISMP 2000, Atlanta, 2000.

[2] H. Dyckhoff, G. Scheithauer, and J. Terno. Cutting and Packing (C&P). In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons, Chichester, 1997.

[3] S.P. Fekete and J. Schepers. On more-dimensional packing I: Modeling. Technical Report ZPR97-288, Mathematisches Institut, Universität zu Köln, 1997.

[4] S.P. Fekete and J. Schepers. On more-dimensional packing II: Bounds. Technical Report ZPR97-289, Mathematisches Institut, Universität zu Köln, 1997.

[5] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[6] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849–859, 1961.

[7] P. C. Gilmore and R. E. Gomory. Multistage cutting problems of two and more dimensions. *Operations Research*, 13:94–119, 1965.

[8] E. Hadjiconstantinou and N. Christofides. An exact algorithm for the orthogonal, 2-D cutting problems using guillotine cuts. *European Journal of Operational Research*, 83:21–38, 1995.

[9] M. Hifi. Contribution à la résolution de quelques problèmes difficiles de l'optimization combinatoire. Habilitation thesis, PRiSM, Universitè de Versailles St-Quentin en Yvelines, 1999.

[10] M. Hifi. Exact algorithms for large-scale unconstrained two and three staged cutting problems. *Combinatorial Optimization and Applications*, 1999 (to appear).

[11] M. Hifi and C. Roucairol. Approximate and exact algorithm for constrained (un)weighted two-dimensional two-staged cutting stock problems. *Journal of Combinatorial Optimization*, 2001 (to appear).

[12] M. Hifi and V. Zissimopoulos. A recursive exact algorithm for weighted two-dimensional cutting problems. *European Journal of Operational Research*, 91:553–564, 1996.

[13] A. Lodi. *Algorithms for Two-Dimensional Bin Packing and Assignment Problems*. PhD thesis, University of Bologna, Italy, 2000.

[14] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: a survey. *European Journal of Operational Research*, 2001 (to appear).

[15] A. Lodi, S. Martello, and D. Vigo. Models and bounds for two-dimensional shelf packing problems. Technical Report OR/00/2, DEIS - Università di Bologna, 2000.

[16] G. S. Lueker. Bin packing with items uniformly distributed over intervals [a,b]. In *Proc. 24th Annual Symp. Found. Comp. Sci.*, pages 289–297, 1983.

[17] S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44:388–399, 1998.

[18] M. Monaci. *Algorithms for Packing and Scheduling Problems*. PhD thesis, University of Bologna, Italy, 2001.

[19] R. Morabito and V. Garcia. The cutting stock problem in hardboard industry: a case study. *Computer and Operations Research*, 25:469–485, 1998.

[20] F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86:565–594, 1999.

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