Journal of the Operational Research Society (2012), 1–9

www.palgrave-journals.com/jors/

A branch-and-price algorithm for the two-stage guillotine cutting stock problem

M Mrad[1*], I Meftahi[2] and M Haouari[3]

[1]*King Saud University, Riyadh, Saudi Arabia;* [2]*University of Carthage, Tunis, Tunisia; and* [3]*Department of Mechanical and Industrial Engineering, College of Engineering, Qatar University, Doha, Qatar*

We investigate the two-stage guillotine two-dimensional cutting stock problem. This problem commonly arises in the industry when small rectangular items need to be cut out of large stock sheets. We propose an integer programming formulation that extends the well-known Gilmore and Gomory model by explicitly considering solutions that are obtained by both slitting some stock sheets down their widths and others down their heights. To solve this model, we propose an exact branch-and-price algorithm. To the best of our knowledge, this is the first contribution with regard to obtaining integer optimal solutions to Gilmore and Gomory model. Extensive results, on a set of real-world problems, indicate that the proposed algorithm delivers optimal solutions for instances with up to 809 items and that the hybrid cutting strategy often yields improved solutions. Furthermore, our computational study reveals that the proposed modelling and algorithmic strategy outperforms a recently proposed arc-flow model-based solution strategy.

Journal of the Operational Research Society advance online publication, 11 July 2012
doi:10.1057/jors.2012.70

Keywords: two-stage cutting stock; guillotine cutting; branch-and-price

1. Introduction

In this paper, we consider the following two-dimensional cutting stock problem. We are given a supply of identical stock rectangles of length H and width W, and a set I of m small rectangular items that have to be cut out of the stock rectangles. Each item $s \in I$ is characterized by a height h_s, a width w_s, and a demand d_s, that corresponds to the number of units of item s that should be cut out. An important feature of the problem under consideration is that only *two-stage guillotine cuts* are permitted. This cutting process is achieved in at most two stages: First, a stock rectangle is slit down its width into strips. Next, these strips are chopped in turn across their heights. Alternatively, it is possible to reverse the ordering of the two cutting directions: that is, start by cutting the stock rectangle across the height direction and then proceed with the width direction. Possibly, a final trimming stage might be required to remove waste and produce an item of a requested size. The problem is to produce the requested items using a minimum number of stock rectangles. This challenging combinatorial optimization problem has a wealth of pertinence to many practical real applications, especially in wood and glass industries, where the goal is to minimize the trim loss.

*Correspondence: M Mrad, Industrial Engineering Department, College of Engineering, King Saud University, PO Box 800, Riyadh 11421, Saudi Arabia.

In the sequel, we shall refer to this problem as the *Two-Stage Guillotine Cutting Stock Problem* (2G-CSP for short).

Problem 2G-CSP has a long history that goes back to nearly half a century ago. Indeed, Gilmore and Gomory (1965) introduced this problem in their seminal paper on multistage cutting stock problem. They presented a formulation of 2G-CSP with a huge (exponential) number of variables and proposed to solve its LP relaxation by a column generation algorithm, where new cutting patterns are iteratively added to a restricted model by solving an integer knapsack subproblem. However, to the best of our knowledge, the computational performance of this column generation algorithm has never been assessed in the literature. Following this pioneering work, an extensive literature has been published on guillotine cutting stock problems. Puchinger and Raidl (2007) focused in their work on the three-stage two-dimensional cutting stock problem, where in the first stage they applied horizontal cuts, in the second stage vertical cuts, and in the third stage horizontal cuts again. They presented integer linear programming models for both the restricted and unrestricted versions of the problem. The unrestricted version is formulated as a set covering problem which is solved using a branch-and-price algorithm. Vanderbeck (2001) adapted a nested decomposition and a column generation method used in a recursive form to deal with the two-dimensional three-staged cutting problem. Furthermore, the column generation method was also adapted for the

two-dimensional cutting stock problem with a guillotine constraint by Cintra *et al* (2008), who considered the cases where the stock rectangles have different sizes and where orthogonal rotations are permitted. Belov and Scheithauer (2006) proposed an algorithm in which they combined the branch-and-price approach with the Chvatal-Gomory cutting planes approach to deal with the one-dimensional and two-dimensional cutting stock problem. For the two-dimensional case the problem is solved in two stages. The set of cutting planes includes two types of cuts: Gomory fractional and mixed-integer cuts and branching constraints cuts. Hifi and Roucairol (2001) considered the case of weighted and non-weighted variants. They proposed approximate algorithms for the exact cutting, where trimming is not allowed, and also for the nonexact cutting where trimming is allowed. The algorithms both generate horizontal and vertical strips that are combined to obtain horizontal or vertical cutting patterns. Furthermore, they proposed exact branch-and-bound algorithms for the nonexact and the exact cutting variants, respectively. Lodi and Monaci (2003) reformulate the cutting stock problem as a two-dimensional knapsack problem where each item is characterized by a height, a width and a profit, the problem requires finding a plan of cut of a unique rectangular stock while maximizing the total profit. They proposed two integer linear models that can be adapted for different variants of the problem. Macedo *et al* (2010) proposed an arc-flow model for the two-dimensional guillotine cutting stock problem. They extended the work of Valério de Carvalho (1999) on the one-dimensional cutting stock problem. In Macedo *et al* (2010), a cutting pattern is viewed as a path in an acyclic graph. They considered two types of graphs: one for each stage cut. The length of the arcs in the first-stage graph represents the distinct item heights. In the second stage, they use a graph for each strip derived from the first stage. Each graph in the second stage is constructed based on the items that can be cut from the corresponding strip. The objective is to minimize the total flow through the graph corresponding to the first stage. Furthermore, they extended their formulation to accommodate both cutting stage ordering. Recently, the unconstrained two-dimensional guillotine-cutting problem (ie, the number of cutting stages is free) has been investigated by Clautiaux *et al* (2011) who proposed a clever graph-theoretical approach.

In addition to formal approaches, several heuristics have been proposed so far to deal with these problems. In particular, Beasley (1985) proposed various optimization-based heuristics for both staged and non-staged two-dimensional guillotine cutting problems. Also, Alvarez-Valdes *et al* (2007) presented heuristics for the weighted constrained two-dimensional two-stage cutting-stock problem. They proposed algorithms that are based on the greedy randomized adaptive search procedure (GRASP). In the constructive phase, they used two iterative procedures: one is based on adding pieces to the partial solution, while the second is based on adding strips to the partial solution. In a second phase, a path relinking approach is invoked to generate high-quality solutions. Hifi and M'Hallah (2006) proposed approximate algorithms for the constrained two-staged two-dimensional cutting problem basing on a strip generation procedure (SGP). This approach consists in constructing a set of strips and looking then for a good combination of some of those generated strips. Each item is characterized by a profit, thus they proposed an integer linear program that seeks to find cutting patterns that maximize the total profit while satisfying the demand for each item. This integer program has been solved using a greedy algorithm. Finally, they improved the obtained solution using a hill climbing approach. Riehme *et al* (1996) investigated approximate algorithms for the two-dimensional guillotine cutting stock problem where the stocks are of different types and where for each type a fixed supply is considered. The solutions are generated in two phases. In the first phase, they consider the aggregated demand of the stock pieces having the same width into one long piece and solve a one-dimensional cutting stock problem to generate strips so as to cover the order demands. Next, in the second step, the selected strips are cut into the initial stock pieces. This second phase requires solving a second one-dimensional cutting stock problem.

In this paper, we propose an exact branch-and-price (B&P) algorithm for 2G-CSP. More precisely, we make the following contributions:

(1) To obtain improved solutions, we propose a formulation that considers *both* stage cutting ordering simultaneously. That is, we seek for a solution that both includes patterns that were obtained by first cutting across the height direction and then proceeding with the width direction, and also those obtained by reversing the cutting process.
(2) We solve the proposed formulation using a branch-and-price algorithm that builds on the pioneering work of Gilmore and Gomory. As far as we know, this is the first contribution with regard to obtaining *integer* optimal solutions to the Gilmore and Gomory model.
(3) We present the results of extensive computational results that provide evidence that the proposed algorithm delivers integer optimal solutions within short CPU times. Also, we show that considering the hybrid cutting strategy offers a significant advantage over the traditional approach that considers only one single cutting process.

The remainder of this paper is organized as follows. Section 2 is devoted to the presentation of the proposed formulation as well as the column generation algorithm that is used to solve the LP relaxation. In Section 3, we provide a description of the branching procedure.

In Section 4, the performance of our branch-and-price algorithm is analysed through an extensive computational study. Finally, we conclude by providing some concluding remarks and directions for future research.

In the sequel, we shall refer to a pattern that is obtained by slitting in the first stage the stock rectangle into strips with heights corresponding to the heights of demanded rectangles as an *h-pattern*. Similarly, we refer to the pattern that is obtained through the reverse cutting process as a *w-pattern*.

2. An integer programming formulation

In this section, we present a formulation for 2G-CSP that is a generalization of the Gilmore and Gomorys model who considered one single cutting ordering. Prior to providing the formulation, we present the following notation:

Parameters:

m^h number of distinct values of heights
m^w number of distinct values of widths
$h_{(k)}$ kth smallest height ($k=1,\ldots,m^h$)
$w_{(k)}$ kth smallest width ($k=1,\ldots,m^w$)
π^h number of h-patterns
π^w number of w-patterns
p_i^h number of one-dimensional patterns that are obtained by chopping in the second stage a strip with height $h_{(i)}$ ($i=1,\ldots,m^h$) and width W into rectangles corresponding to demanded items
p_i^w number of one-dimensional patterns that are obtained by chopping in the second stage a strip with width $w_{(i)}$ ($i=1,\ldots,m^w$) and height H into rectangles corresponding to demanded items
a_{ij}^h number of strips of height $h_{(i)}$ that are included in the jth h-pattern, $i=1,\ldots,m^h$, $j=1,\ldots,\pi^h$
a_{ij}^w number of strips of width $w_{(i)}$ that are included in the jth w-pattern, $i=1,\ldots,m^w$, $j=1,\ldots,\pi^w$
b_{sik}^h number of rectangles of type s that are included in the kth pattern of the strip with height $h_{(i)}$, $s\in I$: $h_s\leqslant h_{(i)}$, $k=1,\ldots,p_i^h$, $i=1,\ldots,m^h$
b_{sik}^w number of rectangles of type s that are included in the kth pattern of the strip with width $w_{(i)}$, $s\in I$: $w_s\leqslant w_{(i)}$, $k=1,\ldots,p_i^w$, $i=1,\ldots,m^w$

Decision variables:

x_j^h number of stock rectangles that are cut in the first stage according to the jth h-pattern, $j=1,\ldots,\pi^h$
x_j^w number of stock rectangles that are cut in the first stage according to the jth w-pattern, $j=1,\ldots,\pi^h$
y_{ik}^h number of strips with height $h_{(i)}$ and width W that are chopped in the second stage according to pattern k, $k=1,\ldots,p_i^h$, $i=1,\ldots,m^h$
y_{ik}^w number of strips with width $w_{(i)}$ and height H that are chopped in the second stage according to pattern k, $k=1,\ldots p_i^w$, $i=1,\ldots,m^w$

The formulation can be stated as follows:

$$\text{Minimize} \sum_{t\in\{h,w\}}\sum_{j=1}^{\pi^t} x_j^t \tag{1}$$

subject to

$$\sum_{j=1}^{\pi^t} a_{ij}^t x_j^t \geqslant \sum_{k=1}^{p_i^t} y_{ik}^t, \quad i=1,\ldots,m^t, t\in\{h,w\}, \tag{2}$$

$$\sum_{t\in\{h,w\}}\sum_{i=1}^{m^t}\sum_{k=1}^{p_i^t} b_{sik}^t y_{ik}^t \geqslant d_s, \quad s=1,\ldots,m, \tag{3}$$

$$x_j^t \geqslant 0, \quad j=1,\ldots,\pi^t, t\in\{h,w\} \tag{4}$$

$$y_{ik}^t \geqslant 0, \quad k=1,\ldots,p_i^t,\ i=1,\ldots,m^t, t\in\{h,w\} \tag{5}$$

$$(x,y) \text{ integer.} \tag{6}$$

The objective function (1) is to minimize the total number of h-patterns and w-patterns. Constraint (2) require that the total number of strips obtained in the first stage cut are larger than1 or equal to those used in the second stage cut. Constraint (3) ensures that the strips that are cut in the second stage produce the required demands. Finally, constraints (4)–(6) impose that the variables are nonnegative integers.

2.1. Solution of the LP relaxation by column generation

Since Model (l)–(6) includes a huge number of variables, we solve its LP relaxation by column generation. At this point, it is worth mentioning that the historical development of this fundamental mathematical programming approach is intimately related to cutting stock applications. Indeed, in their pioneering work on cutting stock problems, Gilmore and Gomory (1961, 1963, 1965) laid the foundations of the column generation technique and paved the way for its rich applications in integer programming.

2.1.1. Generation of an x-variable with minimum reduced cost. Assume that $(u_1^h,\ldots,u_{m^h}^h,u_1^w,\ldots,u_{m^w}^w,v_1,\ldots,v_m)^t$ is a vector of nonnegative dual variables. Thus, the reduced cost of a first-stage variable x_j^h is $c_j^h = 1-\sum_{i=1}^{m^h} a_{ij}^h u_i^h$. Hence, a minimum reduced cost variable is obtained by solving the following integer knapsack problem:

$$\text{Maxmize} \sum_{i=1}^{m^h} u_i^h a_{ij}^h \tag{7}$$

subject to

$$\sum_{i=1}^{m^h} h_{(i)} a_{ij}^h \leqslant H, \tag{8}$$

$$a_{ij}^h \geqslant 0, \text{ integer}, \quad i = 1, \ldots, m^h \tag{9}$$

This problem is solved by dynamic programming as a longest path problem in an acyclic graph. For the sake of clarity, and even though this transformation belongs to folklore, we briefly describe the reduction to a longest path problem. The underlying acyclic digraph $G=(V, A)$ is constructed as follows. A node (i, θ) is associated to each state that corresponds to fillings items with heights from $\{h_{(1)}, \ldots, h_{(i)}\}$ into a knapsack with capacity θ $(i=1, \ldots, m^h,\ \theta=0, \ldots, H)$. There is an arc that links node (i, θ) to node (i', θ') if and only if: (i) $i' = i+1$, and (ii) $\theta' = \theta + kh_{i'}$ (where k is a nonnegative integer). This arc corresponds to the decision of filling k units of item i'' into the knapsack. Hence, its cost is $ku_{i'}^h$. Moreover, we include two additional nodes: a 'start' node $(0, 0)$ and an 'end' node (m^h+1, H). There is an arc that links node $(0, 0)$ to node $(1, \theta)$ having a cost $\lfloor \theta/h_1 \rfloor u_1^h$ $(\theta=0, h_1, \ldots, \lfloor H/h_1 \rfloor h_1)$. Also, each node (m^h, θ) $(\theta=0, \ldots, H)$ is linked by a zero-cost arc to node $(m^h+1,\ H)$.

It is easily realized that there is a one-to-one correspondence between each path in G between nodes $(0, 0)$ and (m^h+1, H) and each feasible knapsack solution, with both solutions having the same total profit.

Similarly, a minimum reduced-cost variable associated with a w-pattern is obtained through solving the following integer knapsack problem:

$$\text{Maximize} \sum_{i=1}^{m^w} u_i^w a_{ij}^w \tag{10}$$

subject to

$$\sum_{i=1}^{m^w} w_{(i)} a_{ij}^w \leqslant W, \tag{11}$$

$$a_{ij}^w \geqslant 0, \text{ integer}, \quad \text{i} = 1, \ldots, m^w \tag{12}$$

2.1.2. Generation of a y-variable with minimum reduced cost. The reduced cost of a second-stage variable y_{ik}^h is $\gamma_{ik}^h = u_i^h - \sum_{s \in I: h_s \leqslant h_{(i)}} b_{sik}^h \upsilon_s$. Thus, a variable y_{ik}^h with minimum reduced cost is obtained by solving the following integer knapsack problem:

$$\xi_i^h = \text{Max} \sum_{s \in I: h_s \leqslant h_{(i)}} \upsilon_s b_{sik}^h \tag{13}$$

subject to

$$\sum_{s \in I: h_s \leqslant h_{(i)}} w_i b_{sik}^h \leqslant W, \tag{14}$$

$$b_{sik}^h \geqslant 0, \text{ integer}, \quad s \in I : h_s \leqslant h_{(i)} \tag{15}$$

Clearly, if $\xi_i^h > u_i^h$ then the pattern variable corresponding to the derived optimal solution should enter the basis.

Assume that the items are indexed in nondecreasing heights (that is, $h_1 \leqslant \cdots \leqslant h_m$). Also, define $F_s(x)$ as the optimal value of the integer knapsack problem that is defined on the item set $\{1, 2, \ldots, s\}$ and capacity x. Hence, we have $\xi_i^h = F_s(W)$ where s is the largest index such that $h_s = h_{(i)}$. We have:

$F_0(0) = 0$

$F_s(x) = \max_{0 \leqslant z \leqslant \lfloor x/w_s \rfloor} \{F_{s-1}(x - zw_s) + z\upsilon_s\}$ for $x \in [0, W]$, $s = 1, \ldots, m$.

It is easy to check that if an acyclic graph, that is similar to the one described in Section 2.1.1, is used to solve the integer knapsack problem defined by (13)–(15), then the value of $F_s(x)$ $(x \in [0, W], s = 1, \ldots, m)$ is just equal to the value of a longest path between nodes $(0, 0)$ and (s, x). Consequently, the values of ξ_i^h can be computed through finding a longest path problem (which amounts to solving a single integer knapsack problem).

Remark 1 To improve the efficacy of the approach, we append to Model (13)–(15) the following valid constraints:

$$\sum_{s \in I: h_s = h_{(i)}} b_{sik}^h \geqslant 1, \tag{16}$$

$$b_{sik}^h \leqslant d_s \quad s \in I : h_s \leqslant h_{(i)}. \tag{17}$$

Constraint (16) requires that an h-pattern of height $h_{(i)}$ should necessarily include at least one item having this height. Also, constraint (17) enforces that the number of units of an item that are included in a pattern should not exceed the demand for this item.

Similarly, the reduced cost of a variable y_{ik}^w is $\gamma_{ik}^w = u_i^w - \sum_{s=1}^{m} b_{sik}^w \upsilon_s$. A variable with minimum reduced cost is obtained by solving the following integer knapsack problem:

$$\xi_i^w = \text{Max} \sum_{s \in I: w_s \leqslant w_{(i)}} \upsilon_s b_{sik}^w \tag{18}$$

subject to

$$\sum_{s \in I: w_s \leqslant w_{(i)}} h_i b_{sik}^w \leqslant H, \tag{19}$$

$$b_{sik}^w \geqslant 0, \text{ integer}, \quad s \in I : w_s \leqslant w_{(i)} \tag{20}$$

If $\xi_i^w > u_i^w$ then the pattern variable corresponding to the derived optimal solution should enter the basis.

In our implementation, at each iteration of the column generation algorithm, we solve four different pricing problems: two of them correspond to x-variables, while

the others correspond to y-variables. In addition, to accelerate the convergence of the column generation algorithm, each time the pricing problem is solved by dynamic programming, several columns with negative reduced cost columns are generated.

3. Branching strategy

Instead of branching on the original (x, y)-variables, we branch on arc variables that appear in the equivalent arc-flow model. This branching strategy is similar to the one that has been implemented by Valério de Carvalho (1999) for solving the one-dimensional bin packing problem (or equivalently, the one-dimensional cutting stock problem). For the sake of clarity, we briefly describe the arc-flow model.

3.1. The arc-flow model

To begin with, we introduce the basic idea that has been proposed by Valério de Carvalho (1999) for the one-dimensional cutting stock problem. Given that the stock height is H and the height of item k is h_k ($k = 1, \ldots, m$), we construct an associated acyclic directed graph G in the following way. The set of vertices is $V = \{0, 1, \ldots, H\}$ and the set of arcs is $A = \{(i,j)$: $0 \leqslant i < j \leqslant H$ and $j - i = h_k$, $k = 1, \ldots, m\} \cup \{(i,j)$: $0 \leqslant i < j \leqslant H$ and $j - i = 1\}$. A key observation is that a valid path in G between nodes 0 and H is equivalent to a cutting pattern. Also, a flow f_{ij} in arc (i,j) corresponds to the number of items k of size $(j-i) = h_k$ placed at position i from the beginning of a stock sheet and corresponds to the unoccupied portions of the bin if $(j-i) = 1$. The objective is to minimize the total flow between nodes 0 and H. Given that, many paths in the graph described above may include the same set of items, Valério de Carvalho (1999) provides reduction criteria that reduce the number of arcs in the graph as well as the symmetry of the solution space.

Example 1 *Let $H = 8$ and consider three items 1, 2, and 3 of sizes 4, 3 and 2 respectively. The corresponding graph after using the reduction criteria with 8 nodes and 12 arcs is represented in Figure 1. A cutting pattern $x = (1,0,2)$ is represented in Figure 2.*

Recently, Macedo *et al* (2010) have extended this flow model to the two-dimensional guillotine cutting stock problem. They considered two distinct graphs: one for each cutting stage. Let G_h^0 and G_w^0 denote the graphs that correspond to the first-stage cutting by considering cuts along the height and width directions, respectively. Also, let $G_h^{s_h}$ (with $S_h \in \{1, \ldots, m_h\}$) and $G_w^{s_w}$ (with $s_w \in \{1, \ldots, m_w\}$) denote the graphs that correspond to the second-stage cutting by considering the height and width directions, respectively.

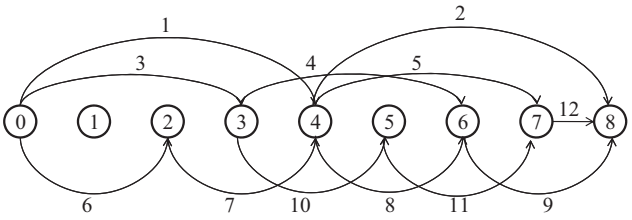


Figure 1 Graph corresponding to Example 1.

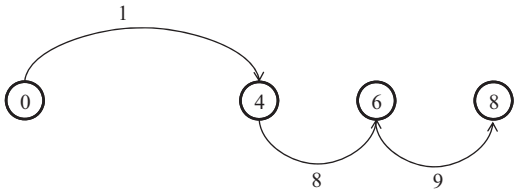


Figure 2 Path corresponding to the pattern $(1, 0, 2)$.

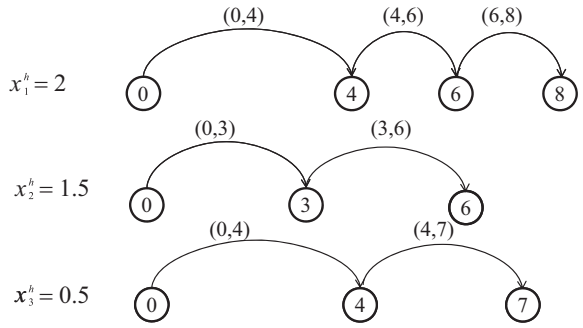
The arc weights correspond to those items that belong to the corresponding strips. Interestingly, Macedo *et al* (2010) described several graph reduction procedures that aim at discarding unnecessary arcs and break symmetry.

3.2. Branching scheme

Assume that at a given node of the search tree, the LP relaxation is solved using the column generation algorithm. Then, the optimal (continuous) solution is converted into the corresponding arc-flow models variables. The value of each arc is set equal to the sum of the decision variables corresponding to the patterns that include this arc. If fractional arc-flows are detected, then we branch on the largest arc having a fractional flow that is the nearest to the source node 0. Let (i,j) be the selected arc and f_{ij} the flow on this arc. Then, we create two nodes by appending the constraints $f_{ij} \leqslant \lfloor f_{ij} \rfloor$ and $f_{ij} \geqslant \lceil f_{ij} \rceil$ respectively.

Example 2 *Let $H = 8$ and the distinct height values are 4, 3 and 2. Assume that the three following h-patterns have been generated: Pattern 1: $(1,0,2)$, Pattern 2: $(0,2,0)$, and Pattern 3: $(1,1,0)$. The optimal solution of the LP relaxation is $x_1^h = 2$, $x_2^h = 1.5$, and $x_3^h = 0.5$. These patterns as well as the corresponding arc-flows are depicted in Figure 3. We see from this figure that the largest arc having a fractional flow is arc $(0,4)$. We have $f_{04} = x_1^h + x_3^h = 2.5$. Thus, two nodes are created. In the first descendant node, we append the cut $x_1^h + x_3^h \leqslant 2$, and the second one we append the cut $x_1^h + x_3^h \geqslant 3$.*

The dual information relative to each branching constraint is easily taken in consideration in the pricing problem which is the longest path problem on the same



$f_{04} = x_1^h + x_3^h = 2.5$
$f_{46} = f_{68} = x_1^h = 2$
$f_{03} = f_{36} = x_2^h = 1.5$
$f_{47} = x_3^h = 0.5$

Figure 3 Arc-flows corresponding to Example 2.

graph of the arc-flow formulation. The branching nodes are processed using a depth-first search strategy.

3.3. Upper bound computation

To derive a tight upper bound that might be useful to accelerate the convergence of the B&P, we implemented the following approach. After solving at the root node of the LP relaxation, we consider the resulting reduced master program and solve it as an integer LP using a commercial solver. In so doing, we generate a feasible heuristic solution.

4. Computational results

The proposed branch-and-price algorithm was coded in C++ and implemented on a Pentium IV 2.2 GHz Personal Computer with 4 GB RAM. All LPs were solved using the commercial solver CPLEX 11.0. The CPU time limit was set equal to 2 h.

The test-bed utilized for the computational study includes 43 real 2G-CSP instances from the wood industry (Macedo *et al*, 2010). These instances are of unequal sizes: while the smallest one includes 16 identical items (that is, $m = 1$), the largest one exhibits 809 items and 31 different sizes.

In a first set of experiments, we assess the performance of our branch-and-price algorithm as well as three two-stage cutting strategies: (i) *W*: Only *w*-patterns are considered in the first cutting stage, (ii) *H*: Only *h*-patterns are considered in the first cutting stage, and (iii) *H&W*: Both *w*-patterns and *h*-patterns are considered in the first stage. Basically, the two former cutting strategies are precisely those described in the original work of Gilmore and Gomory (1965). However, the solution of these integer models has never been reported in the literature. The results are displayed in Table 1. In this table, we display for each instance the number of different items (n), and the total number of items (n_t). Also, we report, for each cutting strategy, the value of the LP relaxation (LP), the value of the integer solution (Exact), the CPU time (Time), and the total number of explored nodes (NN). Note that the values with (*) represent the best solutions founded within the time limit of 7200 s.

Looking at Table 1, we see that the B&P algorithm efficiently delivered proven optimal solutions for the cutting strategies *W*, *H*, and *W&H*, for 38, 43, and 39 instances, respectively. Furthermore, for all non-solved instances, the B&P output approximate solutions that often exhibit unitary absolute gaps. Interestingly, we see that the cutting strategy *W&H* strictly outperformed both *W* and *H* for six instances. Furthermore, we observe that the problems are often quickly solved at the root node. Hence, we found that the average CPU time of the solved instances for the cutting strategy *W&H* is only 14.14 s.

In addition, we compared the performance of our B&P when the cutting strategy *W&H* is considered with the arc-flow model of Macedo *et al* (2010). This latter model was solved using CPLEX 11.0. The results are reported in Table 2. In this table, we indicate for each instance and for each solution strategy whether an optimal solution has been obtained within the 2-h time limit and we report the CPU time required to solve it (Time 1 and Time 2, respectively).

We see from Table 2 that the arc-flow model failed to deliver optimal solutions for eight instances (instead of 5 for the B&P algorithm). Furthermore, it requires significantly longer CPU times. In particular, we observe that for the problem instance A-19, the ratio of the CPU time of the compact model to the average CPU time of the B&P is 311 to 1.

Pushing our analysis a step further, we also considered an additional set of 12 benchmark instances of Cintra *et al* (2008). These instances are labelled gcut-1, . . . , gcut-12, respectively. The results are displayed in Table 3. We see from this table that these 12 instances are easily solved both by the proposed B&P and also by the arc flow model. Moreover, we considered five additional hard instances initially proposed by Cintra for the rectangular knapsack problem. These instances are labelled gcut-13, . . . , gcut-17, respectively. In order to convert these instances into instances for the 2S-CSP, we randomly generated for each item a demand from $U[1, 100]$. The results are displayed in Table 4. We observe from this table, that none of these five hard instances was solved by the arc flow model. Furthermore, we see that for three instances, even the LP relaxation of the compact model remained unsolved after reaching the 2-h CPU time limit. However, the B&P successfully solved two instances and delivered solutions with unitary gaps for two unsolved instances.

Table 1 Performance of the branch-and-price algorithm

Instance	*n*	n_t	*W*				*H*				*W&H*			
			LP	*Exact*	*Time*	*NN*	*LP*	*Exact*	*Time*	*NN*	*LP*	*Exact*	*Time*	*NN*
A-1	2	24	3	3	0.11	0	3.3	4	0.07	0	3	3	0.04	0
A-2	4	38	36	36	0.03	0	36	36	0.05	0	36	36	0.09	0
A-3	2	17	8	8	0.02	0	8	8	0.02	0	8	8	0.04	0
A-4	2	16	2.67	3	0.02	0	2.67	3	0.02	0	2.67	3	0.06	0
A-5	8	138	12.36	13	0.18	0	12.53	13	0.18	0	12.21	13	0.40	0
A-6	2	7	1.89	2	0.03	0	2	2	0.03	0	1.89	2	0.02	0
A-7	5	58	13.06	14	0.04	0	13.13	14	0.05	0	13	13	0.08	0
A-8	1	16	1.07	2	0.04	0	1.07	2	0.01	0	1.07	2	0.02	0
A-9	30	770	58.04	59	5.75	0	60.67	61	4.24	0	58.03	59	11.90	0
A-10	3	44	2.3	3	0.05	0	1.97	3	0.12	3	1.97	3	0.35	6
A-11	20	724	47.19	48	2.17	0	45.76	46	22.44	2	45.61	46	3.16	0
A-12	3	44	14	14	0.02	0	14	14	0.032	0	14	14	0.05	0
A-13	8	304	13.35	14	0.24	0	13.53	14	0.14	0	13.2	14	0.47	0
A-14	31	809	67.13	68	7.34	0	66.82	67	13.23	0	66.21	67	10.89	0
A-15	12	339	39.06	40	0.47	0	38.94	39	0.69	1	38.51	39	1.31	0
A-16	27	744	84.03	85	7.36	0	82.26	83	2.25	0	81.92	82	11.90	0
A-17	3	135	4.82	5	0.11	0	4.70	5	0.05	0	4.70	5	0.06	0
A-18	20	559	66.90	68	25.17	58	64.96	65	1.10	0	64.05	65	2.79	0
A-19	27	507	54.93	56	47.62	46	57.23	58	4.40	0	54.72	55	8.67	0
A-20	10	515	25.08	26	0.44	0	26.10	27	0.31	0	25.03	26	0.60	0
A-21	21	450	26.67	27	1.85	0	27.32	28	2.82	0	26.39	27	3.24	0
A-22	1	24	2.4	3	0.04	0	2.4	3	0.04	0	2.4	3	0.02	0
A-23	8	248	12.67	13	0.25	0	12.92	14	3883.36	98 700	12.52	13	0.62	0
A-24	107	217	34.13	36*	7200	907	34.36	35	152.45	0	33.93	35*	7200	1258
A-25	75	156	17.03	18	114.11	3	17.33	18	184.48	10	16.80	18*	7200	2052
A-26	34	61	7.15	8	5.83	0	7.08	8	6.21	0	6.94	7	109.08	190
A-27	79	180	18.85	20*	7200	1631	19.21	20	101.62	0	18.69	20*	7200	1820
A-28	54	106	10.56	12*	7200	7078	11.17	12	37.53	0	10.43	11	181.26	19
A-29	82	218	27.15	31*	7200	588	27.29	28	101.11	0	26.89	28*	7200	2010
A-30	24	39	3.83	5	4.24	3	3.68	4	2.30	0	3.58	4	7.24	0
A-31	36	64	7.73	8	7.99	0	7.49	8	8.74	0	7.47	8	19.79	0
A-32	99	184	26.13	27	163.44	0	26.90	27	42.89	0	25.92	26	108.99	0
A-33	134	309	34.01	41*	7200	129	34.67	35	125.6	0	33.77	36*	7200	406
A-34	26	46	5.40	6	1.82	0	5.21	6	1.18	0	5.18	6	2.61	0
A-35	68	144	16.30	18*	7200	4000	16.40	17	12.03	0	16.06	17	43.57	0
A-36	16	52	8.88	9	0.36	1	8.88	9	0.54	1	8.88	9	0.46	0
A-37	8	78	4.58	5	0.13	0	4.63	5	0.36	0	4.53	5	0.34	0
A-38	42	160	22.29	23	5.57	0	22.12	23	3.11	0	21.81	22	9.92	0
A-39	11	22	3.28	4	0.25	0	3.06	4	0.74	0	3.05	4	0.58	0
A-40	40	163	15.87	17	5.85	11	15.90	17*	7200	32 250	15.43	16	7.10	0
A-41	32	71	17.33	18	0.92	0	19	19	2.53	2	17.2	18	2.54	0
A-42	8	13	7.25	8	0.06	0	7.17	8	0.06	0	7.13	8	0.10	0
A-43	11	22	6.75	7	0.12	0	6.38	7	0.40	0	6.38	7	0.97	0

Table 2 Comparison of the B&P and the arc-flow model-based approach

Instance	*n*	n_t	*B&P*		*Arc-flow model*		Time2/ Time1
			Solved	*Time*1	*Solved*	*Time* 2	
A-1	2	24	Yes	0.04	Yes	0.12	3.00
A-2	4	38	Yes	0.09	Yes	0.16	1.78
A-3	2	17	Yes	0.04	Yes	0.12	3.00
A-4	2	16	Yes	0.06	Yes	0.15	2.50
A-5	8	138	Yes	0.4	Yes	1.07	2.68
A-6	2	7	Yes	0.02	Yes	0.11	5.50
A-7	5	58	Yes	0.08	Yes	0.21	2.63
A-8	1	16	Yes	0.02	Yes	0.07	3.50
A-9	30	770	Yes	11.9	Yes	567.31	47.67
A-10	3	44	Yes	0.35	Yes	0.43	1.23
A-11	20	724	Yes	3.16	Yes	28.19	8.92
A-12	3	44	Yes	0.05	Yes	0.19	3.80
A-13	8	304	Yes	0.47	Yes	2.68	5.70
A-14	31	809	Yes	10.89	Yes	293.71	26.97
A-15	12	339	Yes	1.31	Yes	4.68	3.57
A-16	27	744	Yes	11.9	Yes	292.73	24.60
A-17	3	135	Yes	0.06	Yes	0.28	4.67
A-18	20	559	Yes	2.79	Yes	18.25	6.54
A-19	27	507	Yes	8.67	Yes	2694.6	310.79
A-20	10	515	Yes	0.6	Yes	3.38	5.63
A-21	21	450	Yes	3.24	Yes	162.93	50.29
A-22	1	24	Yes	0.02	Yes	0.07	3.50
A-23	8	248	Yes	0.62	Yes	1.5	2.42
A-24	107	217	No	7200	No	7200	1.00
A-25	75	156	No	7200	No	7200	1.00
A-26	34	61	Yes	109.08	Yes	101.71	0.93
A-27	79	180	No	7200	No	7200	1.00
A-28	54	106	Yes	181.26	No	7200	39.72
A-29	82	218	No	7200	No	7200	1.00
A-30	24	39	Yes	7.24	Yes	64.72	8.94
A-31	36	64	Yes	19.79	Yes	137.64	6.96
A-32	99	184	Yes	108.99	No	7200	66.06
A-33	134	309	No	7200	No	7200	1.00
A-34	26	46	Yes	2.61	Yes	10.7	4.10
A-35	68	144	Yes	43.57	No	7200	165.25
A-36	16	52	Yes	0.46	Yes	2.42	5.26
A-37	8	78	Yes	0.34	Yes	0.71	2.09
A-38	42	160	Yes	9.92	Yes	1151	116.03
A-39	11	22	Yes	0.58	Yes	1.13	1.95
A-40	40	163	Yes	7.1	Yes	22.3	3.14
A-41	32	71	Yes	2.54	Yes	5.84	2.30
A-42	8	13	Yes	0.1	Yes	0.26	2.60
A-43	11	22	Yes	0.97	Yes	0.63	0.65

5. Conclusion

In this paper, we have addressed the two-stage guillotine two-dimensional cutting stock problem. This problem commonly arises in the wood, steel, or aluminium industry when small rectangular items need to be cut out of large stock rectangles. We have proposed a model that generalizes the well-known Gilmore and Gomory model by explicitly considering a hybrid cutting strategy that yields a mix of both *h*-patterns and *w*-patterns. To solve this model, we proposed an exact branch-and-price algorithm that is based on a branching scheme initially introduced by Valério de Carvalho (1999) in the context of the one-dimensional bin packing. To the best of our knowledge, this is the first contribution with regard to obtaining integer optimal solutions to Gilmore and Gomory model. We have provided the results of computational results to demonstrate the efficacy of the modelling and algorithmic strategy as well as the benefit of the hybrid cutting strategy. Furthermore, we provided evidence that the proposed solution strategy outperforms a recently proposed arc-flow model-based solution strategy.

Table 3 Performance of the branch-and-price algorithm and arc flow model on instances gcut-1..gcut-12

Instance	*n*	n_t	*B&P*				*Arc-flow model*	
			W&H				*W&H*	
			LP	*Exact*	*Time(s)*	*NN*	*Exact*	*Time(s)*
gcut-1	10	669	293.25	294	0.08	0	294	0.10
gcut-2	20	982	344.25	345	0.18	0	345	0.24
gcut-3	30	1489	332.13	333	0.59	0	333	0.38
gcut-4	50	2751	835.83	836	4.49	25	836	0.86
gcut-5	10	645	196.83	197	0.69	28	197	0.17
gcut-6	20	1064	342.67	343	0.40	0	343	0.31
gcut-7	30	1626	591	591	1.97	13	591	0.46
gcut-8	50	2363	690	690	4.87	0	690	1.75
gcut-9	10	590	130.67	131	0.26	3	131	0.27
gcut-10	20	830	293	293	0.63	0	293	0.49
gcut-11	30	1298	329.38	330	1.96	0	330	1.25
gcut-12	50	2081	671.5	672	7.32	0	672	2.48

Table 4 Performance of the branch and price approach and arc-flow model on instances gcut-13..gcut-17

Instance	*n*	n_t	*B&P*				*Arc-flow model*		
			W&H				*W&H*		
			LP	*Exact*	*Time (s)*	*NN*	LP	*Exact*	*Time(S)*
gcut-13	32	1694	91.69	92	27.42	0	91.68	Unsolved	7200
gcut-14	34	1652	74.34	75	37.10	0	74.34	Unsolved	7200
gcut-15	52	2360	70.80	72	7200	262	Unsolved	Unsolved	7200
gcut-16	62	3170	86.59	88	7200	114	Unsolved	Unsolved	7200
gcut-17	82	4019	94.76	99	7200	35	Unsolved	Unsolved	7200

Future work needs to be focused on designing effective exact approaches to a variant of 2G-CSP with multiple stock sizes. This variant assumes that the stock rectangles have unequal sizes and costs. Despite the practical industrial relevance of this latter model, it has received scant attention in the literature. We believe that an issue worthy of future investigation is to extend the ideas discussed in this paper for optimally solving this challenging cutting stock problem.

References

Alvarez-Valdes R, Marti R, Tamarit JM and Parajon A (2007). GRASP and path relinking for the two-dimensional two-stage cutting-stock problem. *INFORMS Journal on Computing* **19**(2): 261–272.

Beasley JE (1985). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society* **36**(4): 297–306.

Belov G and Scheithauer G (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *European Journal of Operational Research* **171**(1): 85–406.

Cintra GF, Miyazawa FK, Wakabayashi Y and Xavier AC (2008). Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. *European Journal of Operational Research* **191**(1): 61–85.

Clautiaux F, Jouglet A and Moukrim A (2011). A new graph-theoretical model for the guillotine-cutting problem. *INFORMS Journal on Computing*, advance online publication, 17 October, doi: 10.1287/ijoc.1110.0478.

Gilmore P and Gomory RE (1961). A linear programming approach to the cutting stock problem. *Operations Research* **9**(6): 849–859.

Gilmore P and Gomory RE (1963). A linear programming approach to the cutting stock problem—Part II. *Operations Research* **11**(6): 863–888.

Gilmore P and Gomory RE (1965). Multistage cutting stock problems of two and more dimensions. *Operations Research* **13**(1): 94–120.

Hifi M and M'Hallah R (2006). Strip generation algorithms for constrained two-dimensional two-staged cutting problems. *European Journal of Operational Research* **172**(2): 515–527.

Hifi M and Roucairol C (2001). Approximate and exact algorithms for constrained (un)weighted two-dimensional two-staged cutting stock problems. *Journal of Combinatorial Optimization* **5**(4): 465–494.

Lodi A and Monaci M (2003). Integer linear programming models for two-staged two-dimensional knapsack problems. *Mathematical Programming Series B* **94**(2–3): 257–278.

Macedo R, Alves C and Valério de Carvalho JM (2010). Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research* **37**(6): 991–1001.

Puchinger J and Raidl GR (2007). Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research* **127**(3): 1304–1327.

Riehme J, Scheithauer G and Terno J (1996). The solution of two-stage guillotine cutting stock problems having extremely varying order demands. *European Journal of Operational Research* **91**(3): 543–552.

Valério de Carvalho JM (1999). Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research* **86**: 629–659.

Vanderbeck F (2001). A nested decomposition approach to a three-stage, two-dimensional cutting stock problem. *Management Science* **47**(6): 864–879.

Received July 2011;
accepted April 2012 after one revision