A Column Generation Based Heuristic for Two-Dimensional Cutting Stock Problems with Variable Dimensions

A Thesis
Presented To
Departamento de Ingeniería Industrial

By

Mauricio A. Ramírez

Advisor: Andrés L. Medaglia, Ph.D.

In Partial Fulfillment
Of the Requirements for the Degree
Magíster en Ingeniería Industrial

Ingeniería Industrial
Centro para la Optimización y la Probabilidad Aplicada (COPA)
Universidad de los Andes
July 2010

A Column Generation Based Heuristic for Two-Dimensional Cutting Stock Problems with Variable Dimensions

Approved by:

Andrés L. Medaglia, Advisor

Jorge E. Mendoza

Eric Pinson

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In the Two-Dimensional Multiple Stock Size Cutting Stock Problem with Variable Dimensions (2D MSSCSP-VD) one wishes to fulfill the demand for a set of two-dimensional items out of a set of stock sheets, each of which has dimensions that may vary in some given range, at minimum cost. Research is scarce on Cutting Stock problems involving variable dimensions so we propose a heuristic to tackle this problem based on existing column generation and dynamic programming strategies. To evaluate its flexibility and the quality of its solutions we test it on a wide range of instances both from the literature and designed by us. The computational experiments show that not only does the approach produce solutions of excellent quality, but also that it is flexible and outperforms other heuristics for related problems that can be reduced to the 2D MSSCSP-VD, such as the 2D MSSCSP and 2D SSSCSP.

1. Introduction

Cutting Stock Problems (CSPs) consist of deciding how to cut a set of small objects (the *items*) from a set of large objects (the *stock sheets*), minimizing the waste (or another measure of cost) while meeting the demand for each item. Because of their practical applications and theoretical relevance, CSPs have received a great deal of attention during the past decades. However, most of the existing literature focuses on problem types in which the dimensions of the stock sheets are problem parameters rather than decision variables. In this work a CSP in which the stock sheet dimensions are allowed to vary within a given range is studied. A column generation based heuristic to tackle the problem is proposed and its performance is evaluated on a large set of different instances.

This paper is organized as follows. Section 2 describes the problem and two of its applications found in the cardboard and steel industries; Section 3 reviews the relevant literature; Section 4 describes the proposed approach; Section 5 summarizes the computational experiments carried out on a large set of instances both from the literature and generated based on the two real-world applications; finally, Section 6 concludes the paper and outlines research perspectives.

2. Problem Description and Motivation

The cutting stock problem with variable dimensions at hand can be described as follows. Let $I = \{1,\ldots,m\}$ be the set of items, each item $i \in I$ having width w_i, height h_i, and demand d_i. The demand for these items must be fulfilled out of the set $J = \{1,\ldots,n\}$ of stock sheets. Due to business conditions and technological constraints, both the width and the height of each stock sheet are allowed to vary within a given range. Therefore, each stock sheet $j \in J$ has minimum and maximum widths $\underline{W_j}$ and $\overline{W_j}$, and minimum and maximum heights $\underline{H_j}$ and $\overline{H_j}$(see Figure 1). There is a cost per area unit u_j and an unlimited quantity available for each stock sheet. Finally, the objective is to meet the demand for items out of the supply of stock sheets at minimum cost.

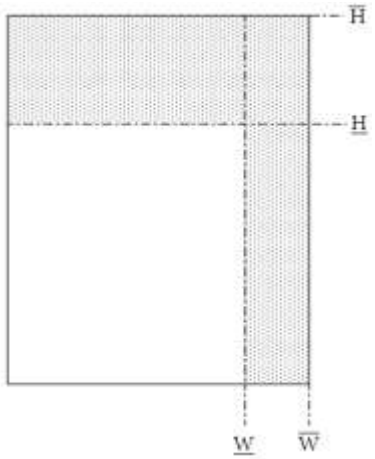

Figure 1. Stock sheet with variable dimensions (valid cutting sizes are shaded)

We consider two additional constraints often found in practical settings. The first one, known as *orientation* constraint, states that the items cannot be rotated on the stock sheets (i.e., an item of width *w* and height *h* and an item of width *h* and height *w* are considered to be different). The second constraint, namely, the *guillotine* constraint, refers to the type of cuts made on the stock sheets. This constraint requires that all cuts made to obtain the items are guillotine cuts, that is, cuts that are

parallel to one of the sides of a rectangle and go all the way through from one of its edges to another one (see Figure 2).

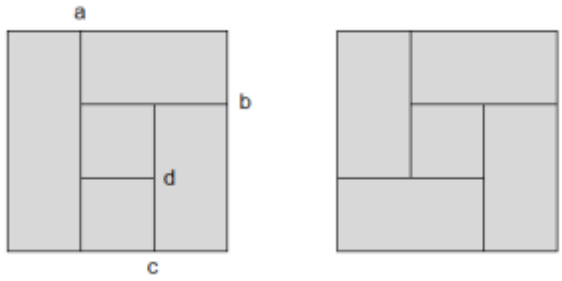

Figure 2. Guillotine (left) and non-guillotine (right) patterns.

The guillotine constraint requires that a sequence of guillotine cuts to obtain the items exists. For instance, to cut the five items in the example shown in Figure 2, the cuts must be performed in the order a-b-c-d. It is worth noting that if the orientation of the cutting tool is fixed (say vertical) as it is often the case in practical settings, performing the cutting sequence a-b-c-d implies three changes in the orientation of the stock sheet. Such changes may be expensive in terms of cost or time, or relative to the benefit obtained by doing so. In these cases it may be desirable to have only a few changes in the orientation of the cuts. Whenever this number is limited to some value k, the problem is said to be *k-stage*. In case this constraint is not enforced, the problem is considered to be *non-stage*.

The two-dimensional cutting stock problem with variable dimensions described above naturally arises in several practical scenarios. For instance, our work was motivated by two applications in the cardboard and steel industries. The first one is the case of a cardboard company that produces boxes that are used by its customers to pack their products. Each of these boxes is put together from one or more cardboard rectangles. Since many types of boxes can be produced, many different types of rectangles are required, each of them typically having a large demand. The rectangles are cut from stock sheets using guillotine cuts and the stock sheets are obtained from a set of rolls available from an external supplier. Each roll has a fixed width and from it stock sheets may be cut with height in a range defined by the supplier's technological constraints. The orientation constraint naturally arises, since the rolls have a fiber direction that needs to be satisfied to preserve the structural properties of the boxes. Finally, because rolls may be of different materials and calibers their costs may differ. The company must then decide i) which stock sheets to cut from the supplier's rolls and their height; and ii) how to cut the required rectangles from these stock sheets. In this case each roll may be seen as a stock sheet with fixed width and variable height.

The second application comes from a company that produces and sells steel cylinders that are used to distribute gas. Each of these cylinders is produced by putting together a number of components, mostly rectangular shapes that are further processed (e.g., by bending them). Since most types of cylinders are similar, only a few different types of rectangles are required, each of them typically having a very large demand. The company cuts these rectangles from stock sheets using guillotine cuts. The stock sheets are cut from steel rolls that are purchased from an external supplier, who offers rolls with width in a given range, defined by its technological constraints, and virtually infinite height. However, to properly handle the heavy steel, the gas cylinder company requires stock sheets cut from the purchased rolls to have heights within a given range. Contrary to the cardboard box case, the orientation of the steel sheets does not affect the structural properties of the cylinders, thus orientation constraints do not need to be enforced. The company must then decide i) which steel rolls to order from the external supplier and their width; ii) which stock sheets to cut from the purchased steel rolls and their height; and iii) how to cut the required steel rectangles from these stock sheets. In this case, the supplier's width constraints combined with the company's height

preferences may be seen as defining one stock sheet with variable width and variable height from which all demand is to be met.

3. Literature Review

3.1. Typology of Cutting and Packing Problems

Dyckhoff (1990) proposed a first typology that attempted to classify Cutting and Packing (C&P) problems. Despite being a pioneering effort, some problems could be classified into different categories and the taxonomy proved to be insufficient with time as the field matured and new problems arose. More recently, Wäscher et al. (2007) proposed a new typology with standard categories that allow consistent problem classification and are flexible enough to consider problems that may appear in the future.

In our problem both dimensions of the stock sheets may vary in some continuous range, thus it belongs to the type of Open Dimension Problems proposed in the new typology. However, since in our case i) the objective is to minimize a function of the employed stock sheets, ii) although there may be many different small items, the demand for each of them is very large so in essence they are weakly heterogeneous in comparison to the total demand, iii) there is a moderate number of different stock sheets (a limited number in the case of the cardboard company and one in the case of the steel company), and iv) all objects are 2-dimensional, our problem can be labeled as a Two-Dimensional Multiple Stock Size Cutting Stock Problem with Variable Dimensions (2D MSSCSP-VD).

The 2D MSSCSP-VD generalizes several Cutting Stock and Bin Packing problems, including the Two-Dimensional Multiple Stock Size Cutting Stock Problem (2D MSSCSP) and the Two-Dimensional Single Stock Size Cutting Stock Problem (2D SSSCSP). In fact, we can show that the 2D MSSCSP-VD is NP-hard as it also generalizes the well-known one-dimensional version of Bin Packing (which is itself NP-hard). Given an instance of Bin Packing, we can transform it into an instance of 2D MSSCSP-VD in polynomial time by assuming that all items, as well as the bin, have height 1. In this case, we get an instance of 2D MSSCSP-VD in which i) there is only one stock sheet j, ii) the stock sheet's minimum and maximum width $\underline{W_j}$ and $\overline{W_j}$ are the same and equal to the bin's width, iii) the stock sheet's minimum and maximum height $\underline{H_j}$ and $\overline{H_j}$ are the same and equal to 1, and iv) all items have height 1 and their original width. A solution to such an instance of the 2D MSSCSP-VD can transformed back into a solution to Bin Packing in polynomial time.

3.2. Related Work

Extensive work has been carried out on C&P problems in one and more dimensions, with Cutting Stock and Bin Packing problems in two dimensions having been studied since the sixties. The works by Gilmore and Gomory (1961, 1963, 1965) are often cited as some of the pioneering approaches to one and two-dimensional cutting stock problems. For recent surveys on C&P and recent advances in the area, the reader is referred to Lodi et al. (2002), Wäscher et al. (2007), and Oliveira and Wäscher (2007). For a survey of recent advances on the most commonly discussed Open Dimension Problem, Strip Packing (SP), the reader is referred to Riff et al. (2009). Many techniques have been proposed to tackle these problems, including linear programming, approximation algorithms, column generation, heuristics, metaheuristics, and, more recently, constraint programming.

Pisinger and Sigurd (2007) propose a branch-and-price method to solve the Two-Dimensional Single Bin Size Bin Packing Problem (2D SBSBPP). They model the problem as a set covering problem and then use column generation to repeatedly solve its linear relaxation. To generate attractive patterns a combination of integer and constraint programming models is used. Many difficult constraints, such as guillotine patterns, k-stage constraints, and bin irregularities, can be guaranteed through the constraint programming model. Once the relaxation is solved, branching is done and the process is repeated until an optimal integer solution is found. An extension of this approach to the

Two-Dimensional Multiple Bin Size Bin Packing Problem (2D MBSBPP) can be found in Pisinger and Sigurd (2005).

Cintra et al. (2008) propose a heuristic to solve the 2D SSSCSP and 2D MSSCSP based on a similar approach. While solving the linear relaxation of the set covering model, attractive patterns are generated with a dynamic programming formulation proposed by Beasley (1985) to solve a two-dimensional knapsack (referred to as Rectangular Knapsack, RK). This recurrence, at the same time, is based on the concept of discretization points presented by Hertz (1972). The solution is guaranteed to meet the guillotine constraint and k-stage and orientation constraints can be met if required by the problem at hand. Once the relaxation is solved, the solution is rounded down and remaining unfulfilled demands are solved by repeating the same procedure.

Macedo et al. (2010) solve the 2D SSSCSP by extending an integer linear programming formulation originally proposed by Valério de Carvalho (1999) for the one-dimensional version of the problem. An arc-flow model is proposed in which the flow through a directed graph represents the number of stock sheets that are to be used; each arc in the graph maps to one or more items, and it must be guaranteed that enough flow goes through the mapped arcs to meet the demand for items. In the extension, the problem is split into two simultaneous stages. The first stage decides a set of strips to horizontally cut from the stock sheet (i.e., the strip heights), while the second stage decides how to vertically cut each of these strips to obtain the final items. While it is possible to consider rotations and define the direction of the first cut, it is not obvious how to incorporate into the model more than one stock sheet or a higher number of stages.

Silva et al. (2010) address the 2D SSSCSP with guillotine constraints by using an approach that produces 2-stage and 3-stage patterns. This approach is based on the notion of cuts and residual plates, i.e., the ways a given rectangle can be cut to generate items from it. An integer programming model decides which cuts are to be made (producing some set of residual plates) such that the demand is fulfilled. They also propose extensions to the basic model for the cases in which rotations are allowed and when more than one stock sheet is available (the 2D MSSCSP).

Among Open Dimension Problems, Strip Packing (SP) is probably the one that has been discussed the most in the literature. In its general version, a set of items and a strip of fixed width and infinite height are given. The objective is to determine the minimum height required to pack all the items in the strip. Different versions may include other considerations such as rotations or guillotine cuts. Some recent approaches were proposed by Martello et al. (2003), who derived tight bounds used in a branch-and-bound approach; Bettinelli et al. (2008), who proposed a branch-and-price approach to solve a particular version of SP; and Kenmochi et al. (2009), who also proposed a solution based on branch-and-bound. However, research is scarce on the 2D MBSBPP and 2D MSSCSP when stock sheet dimensions are variable. Very recently, Wäscher et al. (2007) stated that open dimension problems such as these, involving more than one large object, have not been frequently discussed in the area of C&P, a notable exception being the work by Benati (1997).

Benati (1997) studied a problem that is very similar to, and can be categorized as, a 2D MSSCSP-VD. In his case, a set of *n* items needs to be cut from a set of *m* rolls. Each roll has a fixed width and a virtually infinite height; however, due to technological constraints cuts cannot be made at a height greater than a given value $\overline{H}$, thus limiting the size of the stock sheets that may be obtained from the rolls. Rotations are allowed, so the problem is non-oriented. No roll costs are considered and the sole objective is to minimize waste. The author proposes a greedy heuristic that generates a pattern having minimum waste until demand for all items has been met. Experimental results show that his approach is not only fast but also effective, achieving near-optimal waste percentages. All solutions are guaranteed to meet the guillotine constraint and consider at most 2 stages.

Cui and Lu (2009) recently studied a problem in steel bridge construction involving large objects with variable width and height. A set of rectangular steel items (which they refer to as *blanks*) is to be cut from a set of stock sheets. The stock sheets are purchased from a supplier and may have any width between some given values $\underline{W}$ and $\overline{W}$ and any height between some given values $\underline{H}$ and $\overline{H}$. These width and height constraints may be seen as defining one unique stock sheet with variable width and size from which all items are to be cut, very much like in the case of the steel company

described in Section 2. The objective is to minimize waste. To approach the problem, they propose a heuristic that attempts to pack as many blanks as possible into low-waste patterns. Any remaining demand is then fulfilled through layered patterns that are generated with a recursive algorithm. The reported results are of very good quality, reaching utilization percentages of nearly 100%. However, one crucial difference between this problem and ours lies in the characteristics of the assortment of small items. While in our scenario we assume that this assortment is weakly heterogeneous, yielding a cutting stock problem, Cui and Lu (2009) state that the demand of a small item is often low and the number of different blank types is high, thus implicitly assuming a strongly heterogeneous assortment, which yields a bin packing problem. This difference is crucial, as methods designed for one problem may not perform well when solving instances of the other and vice versa. The last recursive step may consume a long time when the number of blanks is very large. This means that while instances with small demands may run in reasonable times, others such as ours with large demands may take significantly longer times to solve, making the approach impractical.

Research on Cutting Stock and Bin Packing problems with variable dimensions is therefore scarce and only a few greedy heuristic approaches to tackle them have been proposed. The present work introduces a new heuristic to solve the 2D MSSCSP-VD via column generation that is, to the best of our knowledge, the first one to make use of this technique in the case of variable dimensions. We argue that the concept of discretization points developed by Herz (1972) can be used to generate attractive patterns in the 2D MSSCSP-VD. Furthermore, our approach is the only one out of the existing ones that allows considering different stock sheet costs, since other existing approaches are limited to minimizing waste. On a different line, given the possibility of reducing other CSPs such as the 2D MSSCSP and 2D SSSCSP (and their one-dimensional counterparts) to the 2D MSSCSP-VD, it is important to note that the proposed approach can be used to solve these problems as well, which means it is very flexible and could be applied in a wide range of industrial settings. As will be shown in Section 5, the approach produces near-optimal solutions in very competitive times, and consistently outperforms other heuristics for both variable and fixed dimension problems commonly found in the literature.

4. An Approach via Column Generation and Dynamic Programming

To solve the 2D MSSCSP-VD we propose the following four-step strategy: 1) modeling the 2D MSSCSP-VD as a set covering problem; 2) solving the linear relaxation via column generation; 3) determining new patterns in the column generation procedure via dynamic programming; and 4) finding the optimal integer solution to the 2D MSSCSP-VD set covering model restricting the set of columns to those considered during the column generation step.

Steps 1 and 2 have been extensively used in the literature to tackle CSPs. In the set covering formulations for CSPs, each variable corresponds to a feasible cutting pattern; therefore, because of the large number of variables, these models fit well the column generation scheme. Step 3 was originally proposed by Cintra et al. (2008) to generate feasible cutting patterns for the 2D SSSCSP, 2D MSSCSP, and SP; we extended it to the 2D MSSCSP-VD. Finally, in Step 4 we optimally solve the (integer) set covering model considering only the generated columns. In the remainder of this section we will discuss in detail each of the four steps of our solution strategy.

4.1. Set Covering Model

The 2D MSSCSP-VD can be modeled as a set covering problem following the ideas originally proposed by Gilmore and Gomory (1961). Let P_j be the set of feasible patterns that can be cut from stock sheet j, p a given pattern in P_j, u_j the cost per unit area of stock sheet j, w_p the width of pattern p (that must be between $\underline{W_j}$ and $\overline{W_j}$,) h_p the height of pattern p (that must be between $\underline{H_j}$ and $\overline{H_j}$), x_p the number of times pattern p is cut from stock sheet j, a_{ip} the number of times item i appears on pattern p, and d_i the demand for item i. The cost of a pattern p will be given by the cost per area unit

of the stock sheet j it is cut from multiplied by the pattern's area, i.e., it will be equal to $(u_j w_p h_p)$. The 2D MSSCSP-VD integer programming formulation follows:

$$min \quad \sum_{j \in J} \sum_{p \epsilon P_j} (u_j w_p h_p) x_p \tag{1}$$

$$s.t. \quad \sum_{j \in J} \sum_{p \epsilon P_j} a_{ip}\, x_p \geq d_i, \quad \forall i \in I \tag{2}$$

$$x_p \in \mathbb{Z}^+ \cup \{0\}, \quad \forall j \in J, p \in P_j \tag{3}$$

In the above formulation (1) is the minimization of the sum of costs of patterns that are cut from the different stock sheets. Constraints (2) guarantee that the demand for items is met with the cut patterns. Finally, constraints (3) define the integer nature of the decision variables, i.e., of the number of times each pattern is cut. Note that the only constraints on the width and the height of pattern p cut from stock sheet j are that these lie within that stock sheet's feasible dimensions, that is, $w_p \in [\underline{W_j}, \overline{W_j}]$ and $h_p \in [\underline{H_j}, \overline{H_j}]$.

By dropping integrality constraints (3) we obtain the relaxed set covering model. The relaxed model has a huge number of variables. In fact, one could argue that since both width and height are variables in a continuous range then infinite pattern sizes (and therefore infinite patterns) need to be considered. However, in the next subsection we argue that it is sufficient to consider only a finite subset of these.

4.2. Generating columns via dynamic programming

The relaxed version of the set covering formulation (1)-(3) is a perfect candidate for column generation. The number of feasible cutting patterns is very large and so is the number of variables in the model since there is a one to one correspondence between both sets. At each iteration of the Simplex algorithm a negative reduced cost pattern is generated and added to the model. The pattern's cost as defined in (1) is added to the objective function and the items contained in it are added to their corresponding constraints in (2).

One advantage of this approach is that all problem conditions and constraints (2D, multiple stock sheets, variable dimensions, guillotine cuts, orientation, stages) need to be guaranteed only at this point. This allows enormous flexibility in terms of the methods used to generate a pattern; dynamic, linear, and constraint programming, specialized algorithms, and heuristics are some of the most common approaches for this. The problem of generating a new variable (i.e., a new pattern) that meets the required conditions is commonly referred to as the *pricing* problem, or simply as the *subproblem*.

Let $\bar{c}_p$ be the reduced cost of non-basic pattern p cut from stock sheet j, and π be the vector of dual variables at the current iteration of the Simplex algorithm. Detailing the cost of pattern p as done in (1), the reduced cost of non-basic patterns can then be calculated as:

$$\bar{c}_p = u_j w_p h_p - \pi^T a_p \tag{4}$$

Rewriting the last term, equation (4) becomes:

$$\bar{c}_p = u_j w_p h_p - \sum_{i \in I} \pi_i a_{ip} \quad (5)$$

The first term in the right-hand side is the cost of the candidate pattern and is always positive since the cost per area unit and the dimensions are positive by definition. The second term in the right-hand side may be interpreted as a measure of how valuable the items packed into the pattern are at the current Simplex iteration. Note that to each item $i \in I$ corresponds a constraint in (2) and therefore a dual variable π_i. This can be seen as how valuable each item is when deciding which ones to include in the pattern: for each time item i is included, the reduced cost of the pattern will decrease by π_i units. The issue lies then on generating a minimum-reduced cost, guillotine pattern that meets both the orientation and k-stage constraints whenever these are required.

This subproblem is similar to that faced by Cintra et al. (2008) when dealing with the 2D SSSCSP and 2D MSSCSP, which required solving a RK. In the RK, a single stock sheet of width W and height H is available. A set I of items is to be cut from the stock sheet, each item $i \in I$ having width w_i, height h_i, and value v_i, so that the value of cut items is maximized. Guillotine cuts are required and it is assumed that the quantity of each item is unlimited. Beasley (1985) proposed a recurrence that solves the RK based on the concept of discretization points by Herz (1972). In the case of the 2D SSSCSP, if the recurrence is applied on the single stock sheet considering the dual variables of the constraints as item values, a minimum-cost guillotine pattern will be generated. In the case of the 2D MSSCSP the same strategy can be followed, with a slight modification: for each of the stock sheets one solves a RK and then returns the minimum-reduced pattern among all of them.

Our proposed approach, based on dynamic programming and column generation, is inspired by Cintra's et al., makes use of Beasley's dynamic programming algorithm for the RK, and of Herz' concept of discretization points. The idea behind the approach is that even though a dimension (width or height) is continuous, when making cuts to generate patterns it is not necessary to consider all (infinite) possible cut points, but only a finite subset of these. The set of relevant patterns, which was initially thought to be infinite, immediately becomes finite as many of the patterns can be safely discarded. This can be done thanks to Herz' result, which is briefly discussed next, and its applicability to the case of variable dimensions.

Given a dimension of size D (e.g., a width of size W or a height of size H), we define the set of discretization points for that dimension as all points d with $d \leq D$ that are reachable by an integer conical combination of the items (i.e., a combination of $w_1,\ldots,w_m$ for the width and $h_1,\ldots,h_m$ for the height). Whenever all cuts in a pattern are made at discretization points the pattern is said to be *canonical*. The result by Herz states that it is only necessary to consider canonical patterns, because for each non-canonical pattern there exists at least one equivalent canonical pattern. Following the notation in the literature, we refer to the set of discretization points for the width as P and to the set of discretization points for the height as Q.

Beasley used this result to propose a recurrence formula over the set of rectangles defined by PxQ that optimally solves the RK with guillotine cuts. Given a stock sheet of size (w,h), the idea is to determine which guillotine cuts to make at discretization points such that the value $V(w,h)$ of items fitting into the resulting rectangles is maximal. As it suffices to consider patterns with cuts such as these (made at discretization points) the solution will be optimal. Note that when dimensions are fixed, set P is built by determining all discretization points up to the maximum (fixed) width W and set Q is built by determining all discretization points up to the maximum (fixed) height H. Since dimensions are fixed, only one out of the many rectangles defined by PxQ can be considered as a solution, namely that of size (W,H) and having value $V(W,H)$.

In the variable-dimension case, however, continuous cutting ranges for stock sheet j are defined by $\underline{W}_j$ and $\overline{W}_j$, and $\underline{H}_j$ and $\overline{H}_j$. But the result by Herz states that even though these ranges contain infinite possible cutting points, only a finite subset of them is relevant. Therefore, one can first determine the set P of discretization points (for the width) up to the maximum width $\overline{W}_j$ and the set Q of discretization points (for the height) up to the maximum height $\overline{H}_j$. Each rectangle of size (w,h),

$w \in P$ and $h \in Q$, will have a profit $S_j(w,h)$ given by the best value $V(w,h)$ that can be cut from it (which can be determined through Beasley's recurrence) minus the cost c_{p^*} of cutting the pattern p^* that leads to the optimal value $V(w,h)$:

$$S_j(w,h) = V(w,h) - c_{p^*} \tag{6}$$

The cost c_p of pattern p was defined in (1) and so we rewrite the last term:

$$S_j(w,h) = V(w,h) - u_j w_{p^*} h_{p^*} \tag{7}$$

Since in our 2D MSSCSP-VD formulation the cost of a pattern is not dependent on its characteristics but only on the size of the stock sheet from which it is cut, the above equation can be simplified as follows:

$$S_j(w,h) = V(w,h) - u_j wh \tag{8}$$

Note, however, that additional pattern costs could be incorporated into (8) if problem conditions were to change. In particular, costs that are not proportional to a pattern's area could be considered. An example of such a cost is the setup cost of rotating a guillotine's orientation (i.e., moving from one stage to the next): if two patterns have the same value and same area, but a different number of stages, the one with fewer stages may be preferred in practice. In general, costs that are not proportional to a pattern's area could be incorporated.

To determine the minimal reduced-cost, feasible, non-basic pattern on stock sheet *j*, an RK is solved on a stock sheet of size $(WMAX_j, HMAX_j)$ using Beasley's recurrence. Afterwards, profit (8) is calculated for each rectangle of size *(w,h)*, $(w \in P, WMIN_j \leq w \leq WMAX_j)$ and $(h \in Q, HMIN_j \leq h \leq HMAX_j)$. The most profitable of these rectangles will define the minimum-reduced cost pattern for stock sheet *j*. The procedure can then be repeated across all stock sheets to obtain the global minimum-reduced cost pattern that will enter the basis at the current iteration.

4.3. Generating integer solutions

After a finite number of iterations, no patterns with negative reduced cost will exist and the relaxed set covering model will have been solved. The optimal solution over this set of patterns may not be integral, since integrality constraints (3) were relaxed. In this case, it is necessary to somehow convert the solution into an integer solution.

Cintra et al. propose rounding down all fractional solutions from the linear programming relaxation. Any unfulfilled demands are considered to be a *residual instance* and the linear relaxation is again solved considering only residual demands. Whenever the rounded-down solution is all zeroes, two approaches are taken. Either a specialized algorithm is used to completely fulfill the remaining demand, or a specialized algorithm is used to fulfill some of the demand and whatever remains is solved as a residual instance of the linear relaxation.

Alternatively, we propose a more straightforward procedure to obtain an integer solution. Once the linear relaxation is solved, we solve the integer program (1)-(3) considering only the set of patterns (columns) that were generated using the built-in branch-and-bound routine of a commercial solver.

5. Computational Experiments

To test our column generation based heuristic, we coded the dynamic programming algorithms to solve the subproblem, as presented in Cintra et al. (2008), in C++ and compiled them as a Mosel[1] module using Microsoft Visual C++ 2008. Then we implemented the master problem in Mosel and set it to run on the Xpress Optimizer, version 20.00.05.. All computational experiments reported in this section were run on a machine with an Intel Core 2 Duo 1.8GHz with 4GB RAM and running Windows XP Professional SP3.

5.1. 2D MSSCSP-VD (Variable Height) instances

We first tested our approach on 2 sets of 2D MSSCSP-VD instances with fixed width and variable height. The first set is comprised of 12 instances and was generated by us based on information from the cardboard company. These instances are all labeled "vh" (for variable height) followed by the number of the instance. All instances have 10 stock sheets, with the width of each stock sheet being fixed and randomly selected from the range [1000, 1200], and the minimum and maximum height of each stock sheet being 0 and 1200, respectively. Area unit costs are different across stock sheets. The item height and width are randomly selected from the interval [100, 1000] and the demand from [1000, 5000]. We tried 6 parameter combinations (non/2-stage, rotations), for a total of 72 instances. We assume that the set of initial cuts can be done in any direction. Tables 1-6 summarize the results under the light of 7 performance metrics: the lower bound obtained by the linear relaxation ("LB"), the best integer solution found ("Sol"), the gap with the relaxation ("Gap"), the time spent on the column generation step ("CG Time"), the time spent to solve the integer program over the set of generated columns ("BB Time"), the total execution time (Time), and the number of generated columns ("Cols"),

Results for the first set are very promising. Gaps with the relaxation are almost negligible, averaging a mere 0.01% across all 72 instances. This clearly shows that the integer solutions obtained by the heuristic are of excellent quality. Furthermore, the number of columns generated is essentially constant across different instances and parameter combinations, averaging 41 generated columns per instance. Running times are competitive and well-suited for time constraints generally found in industry, with the oriented instances taking 59.95 s on average to solve and the non-oriented instances taking 411.79 s on average to solve. The difference in times between both subsets can be explained by the fact that solving a non-oriented instance is equivalent to solving an instance with more items, and this may significantly increase the size of the recurrence that needs to be solved.

[1] Mosel is a modeling and programming language for rapid prototyping of optimization models that is closely tied to the commercial Xpress-MP optimizer by FICO.

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vh1	12,153,622.89	12,154,728.21	0.01%	23.35	0.02	23.36	42
vh2	11,826,667.24	11,827,887.79	0.01%	22.77	0.08	22.85	49
vh3	12,026,476.72	12,027,377.29	0.01%	19.49	0.03	19.52	34
vh4	14,284,628.03	14,285,628.10	0.01%	10.55	0.02	10.56	20
vh5	11,388,088.63	11,389,249.62	0.01%	2.42	0.14	2.56	32
vh6	14,527,635.00	14,529,386.25	0.01%	4.49	0.02	4.50	35
vh7	20,899,576.26	20,901,627.12	0.01%	6.99	0.02	7.00	18
vh8	10,697,033.68	10,697,550.99	0.00%	29.00	0.01	29.01	55
vh9	15,387,727.74	15,389,649.64	0.01%	10.81	0.02	10.83	46
vh10	13,986,145.36	13,987,572.44	0.01%	4.95	0.01	4.96	24
vh11	12,256,724.93	12,257,290.30	0.00%	8.59	0.01	8.60	33
vh12	12,283,011.28	12,284,224.05	0.01%	15.19	0.03	15.22	44

Table 1. Results for the 2D MSSCSP-VD (variable height), non-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vh1	12,157,219.79	12,157,788.23	0.00%	113.97	0.02	113.99	40
vh2	11,827,017.93	11,827,515.24	0.00%	110.52	0.02	110.53	45
vh3	12,026,476.72	12,028,479.11	0.02%	101.02	0.01	101.03	31
vh4	14,284,628.03	14,285,745.03	0.01%	71.88	0.02	71.89	25
vh5	11,456,961.77	11,458,621.03	0.01%	13.50	0.02	13.52	28
vh6	14,544,121.88	14,546,569.44	0.02%	24.31	0.02	24.33	29
vh7	20,899,576.26	20,902,302.80	0.01%	35.84	0.02	35.86	16
vh8	10,755,403.64	10,756,529.89	0.01%	91.75	0.05	91.80	33
vh9	15,388,243.62	15,389,860.33	0.01%	56.53	0.03	56.56	41
vh10	13,991,746.79	13,994,139.70	0.02%	27.33	0.06	27.39	20
vh11	12,272,284.63	12,274,030.20	0.01%	42.47	0.03	42.50	29
vh12	12,324,944.84	12,325,939.99	0.01%	63.14	0.06	63.20	34

Table 2. Results for the 2D MSSCSP-VD (variable height), 2-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vh1	12,153,622.89	12,154,728.21	0.01%	175.27	0.01	175.28	42
vh2	11,826,667.24	11,827,887.79	0.01%	183.81	0.02	183.83	49
vh3	12,026,476.72	12,027,984.48	0.01%	151.47	0.02	151.49	36
vh4	14,284,628.03	14,285,628.10	0.01%	94.02	0.01	94.03	20
vh5	11,388,088.63	11,389,249.62	0.01%	18.55	0.11	18.66	32
vh6	14,527,635.00	14,529,386.25	0.01%	36.28	0.02	36.30	35
vh7	20,899,576.26	20,901,627.12	0.01%	61.55	0.02	61.56	20
vh8	10,697,033.68	10,698,152.12	0.01%	232.97	0.02	232.99	54
vh9	15,387,727.74	15,389,649.64	0.01%	86.60	0.02	86.61	46
vh10	13,986,145.36	13,987,572.44	0.01%	39.59	0.02	39.61	24
vh11	12,256,724.93	12,257,290.30	0.00%	63.61	0.02	63.63	33
vh12	12,283,011.28	12,284,713.58	0.01%	102.45	0.05	102.50	41

Table 3. Results for the 2D MSSCSP-VD (variable height), 4-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vh1	11,840,813.28	11,841,414.11	0.01%	144.49	0.03	144.52	55
vh2	10,866,853.37	10,867,086.55	0.00%	147.35	0.11	147.46	54
vh3	11,310,303.58	11,311,413.76	0.01%	194.30	0.02	194.32	57
vh4	13,327,059.76	13,327,750.34	0.01%	146.32	0.05	146.36	50
vh5	11,205,813.89	11,206,926.98	0.01%	28.49	0.02	28.50	44
vh6	13,914,050.51	13,915,121.66	0.01%	33.90	0.01	33.91	44
vh7	19,694,467.61	19,695,256.95	0.00%	95.81	0.02	95.83	39
vh8	10,147,861.17	10,147,907.06	0.00%	178.08	0.33	178.41	61
vh9	14,949,597.20	14,951,082.11	0.01%	56.67	0.07	56.74	41
vh10	13,336,303.23	13,338,284.57	0.01%	57.33	0.02	57.34	37
vh11	11,759,720.18	11,760,550.51	0.01%	59.58	0.03	59.61	43
vh12	11,666,538.06	11,667,399.05	0.01%	95.41	0.03	95.44	53

Table 4. Results for the 2D MSSCSP-VD (variable height), non-stage, non-oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vh1	11,840,813.28	11,841,744.59	0.01%	512.29	0.02	512.31	46
vh2	10,883,655.76	10,884,263.78	0.01%	543.64	0.06	543.70	45
vh3	11,337,978.70	11,338,337.43	0.00%	766.84	0.13	766.96	55
vh4	13,359,722.92	13,361,007.58	0.01%	667.92	0.02	667.93	51
vh5	11,227,350.52	11,228,442.84	0.01%	168.44	0.01	168.45	38
vh6	13,917,006.17	13,917,773.34	0.01%	229.13	0.01	229.14	42
vh7	19,715,521.97	19,716,941.13	0.01%	362.88	0.02	362.89	33
vh8	10,164,102.63	10,164,695.40	0.01%	644.70	0.06	644.76	48
vh9	14,982,988.49	14,983,383.99	0.00%	304.45	0.09	304.55	43
vh10	13,345,483.10	13,346,862.91	0.01%	299.37	0.02	299.39	36
vh11	11,763,845.04	11,763,968.13	0.00%	311.55	0.01	311.56	44
vh12	11,691,237.88	11,693,135.60	0.02%	396.49	0.03	396.52	44

Table 5. Results for the 2D MSSCSP-VD (variable height), 2-stage, non-oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vh1	11,840,813.28	11,841,749.06	0.01%	1004.37	0.05	1004.42	61
vh2	10,867,139.80	10,867,272.15	0.00%	986.69	0.23	986.92	55
vh3	11,310,303.58	11,311,413.76	0.01%	1178.11	0.02	1178.13	57
vh4	13,327,059.76	13,327,750.34	0.01%	911.61	0.05	911.65	50
vh5	11,205,813.89	11,206,926.98	0.01%	248.10	0.02	248.12	44
vh6	13,914,050.51	13,915,121.66	0.01%	298.02	0.02	298.04	44
vh7	19,694,467.61	19,695,256.95	0.00%	752.83	0.02	752.84	39
vh8	10,147,861.17	10,147,907.06	0.00%	1087.57	0.41	1087.98	61
vh9	14,949,597.20	14,950,008.58	0.00%	429.52	0.03	429.55	45
vh10	13,336,303.23	13,338,284.57	0.01%	324.85	0.01	324.86	32
vh11	11,759,720.18	11,760,550.51	0.01%	444.13	0.03	444.16	43
vh12	11,666,538.06	11,667,592.06	0.01%	711.15	0.17	711.32	49

Table 6. Results for the 2D MSSCSP-VD (variable height), 4-stage, non-oriented

The second set was proposed by Benati (1997) so we compare our results with those of his greedy heuristic. The testbed comprises 33 instances: 18 problems that are publicly available and 15 that are

no longer available but whose conditions can be reproduced following the characteristics provided by the author. The instances by Benati map to the case of the cardboard company, where only the height of the stock sheets is considered to be variable. All benchmarks assume that the problem is non-oriented. Stock sheet costs are all the same, so the objective is to minimize waste. In all instances, both directions are allowed for the initial cuts and rotations are allowed. Since Benati's heuristic produces only 2-stage patterns, we restricted the maximum number of stages to 2 in order to allow a head-to-head comparison with his approach. Instances labeled "P" are those reported in Benati's original article that are still available, while instances labeled "R" are those that we reproduced.

The "P" instances have between 1 and 3 stock sheets, with the width of each stock sheet being between 1050 and 1330 and fixed, and its minimum and maximum height being 0 and 1500, respectively. The item height and width range from about 200 to 1300, and the demand ranges from [100, 5000]. Table 7 summarizes this set, including a new metric ("Gap BKS"), which is the gap between the previous solution (in this case coming from Benati, calculated given the waste percentage he presents) and the current best solution. Once more, results show that our approach is very effective at producing excellent solutions: out of 18 instances we improved all 18 integer solutions. Furthermore, the improvement was significant as we reduced Benati's average gap of 1.04% (calculated between his solution and our relaxation) to only 0.01%. The average improvement over previous solutions is given by the average of column "Gap BKS" and is equal to 1.04%. The average number of generated columns once more is low and equal to 29, with an average total running time (linear relaxation plus integer program) of only 4.8 s.

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Sol Benati	Gap BKS
P1-A1	4,528,317,500.00	4,528,317,500	0.00%	7.99	0.11	8.09	25	4,780,947,734	5.28%
P1-A2	4,465,807,416.67	4,466,065,750	0.01%	10.94	0.02	10.95	27	4,550,755,844	1.86%
P1-A3	4,384,048,333.33	4,384,177,500	0.00%	10.08	0.02	10.09	24	4,416,383,133	0.73%
P1-B1	4,563,353,333.33	4,563,731,300	0.01%	6.13	0.08	6.20	35	4,567,334,190	0.08%
P1-B2	4,339,697,500.00	4,339,886,000	0.00%	8.03	0.02	8.05	36	4,352,351,439	0.29%
P1-B3	4,337,551,666.67	4,337,958,000	0.01%	9.31	0.02	9.33	35	4,352,351,439	0.33%
P2-A1	6,467,116,061.83	6,467,524,000	0.01%	3.92	0.50	4.42	30	6,528,688,019	0.94%
P2-A2	6,443,398,579.55	6,443,867,500	0.01%	3.16	0.14	3.30	19	6,453,820,783	0.15%
P2-A3	6,443,398,579.55	6,443,625,900	0.00%	3.47	0.13	3.59	20	6,460,376,879	0.26%
P2-B1	6,518,946,861.11	6,519,290,200	0.01%	1.80	0.13	1.92	23	6,605,312,629	1.30%
P2-B2	6,451,794,444.44	6,453,537,600	0.03%	1.14	0.06	1.20	11	6,492,032,462	0.59%
P2-B3	6,421,579,166.67	6,423,014,420	0.02%	1.45	0.05	1.50	12	6,464,974,077	0.65%
P3-B1	18,568,411,111.11	18,568,521,000	0.00%	1.44	0.02	1.45	45	18,657,924,131	0.48%
P3-B2	17,623,587,500.00	17,624,805,000	0.01%	2.00	0.02	2.02	46	17,728,557,166	0.59%
P3-B3	17,393,125,000.00	17,393,125,000	0.00%	1.97	0.02	1.98	37	17,533,251,922	0.80%
P3-C1	18,672,313,333.33	18,672,428,600	0.00%	2.19	0.02	2.20	42	19,201,564,989	2.76%
P3-C2	17,731,420,000.00	17,731,834,400	0.00%	2.06	0.02	2.08	30	17,914,935,754	1.02%
P3-C3	17,304,945,000.00	17,305,722,000	0.00%	2.22	0.02	2.24	28	17,403,027,645	0.56%

Table 7. Results for the 2D MSSCSP-VD (variable height, Benati 'P' instances), 2-stage, non-oriented

The "R" instances consider 1 stock sheet, its width being 100 and fixed, and its minimum and maximum height being 0 and 150, respectively. The number of items ranges from 25 to 1000, with item height and width randomly selected from the interval [1, 100], and the demand randomly selected from the interval [100, 20000]. Table 8 summarizes the results for this set, which is no longer available but whose conditions we reproduced to generate a new comparable one for which we could present results. The average gap continues to be a mere 0.01%, although the average number of columns and average running time increase significantly to 1,397 and 1,434 s, respectively. These

increases are explained by the last five instances which consider 1000 items and require a significantly larger number of columns to be generated when solving the linear relaxation.

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
R25A	542.643.693,04	542.662.000	0,00%	3,44	0,02	3,45	63
R25B	553.380.736,98	553.434.000	0,01%	2,75	0,02	2,77	52
R25C	801.253.328,68	801.281.200	0,00%	3,61	0,02	3,63	73
R25D	656.570.147,14	656.620.100	0,01%	3,81	0,02	3,83	65
R25E	764.784.580,00	764.798.500	0,00%	3,83	0,02	3,84	72
R100A	2.722.298.823,33	2.722.430.600	0,00%	43,38	0,03	43,41	379
R100B	3.267.443.923,00	3.267.569.000	0,00%	44,30	0,03	44,33	375
R100C	2.802.726.368,00	2.802.856.600	0,00%	41,37	0,02	41,39	360
R100D	2.564.569.395,93	2.564.731.200	0,01%	47,97	0,03	48,00	411
R100E	2.761.497.170,00	2.761.612.900	0,00%	35,74	0,02	35,75	342
R1000A	24.359.232.494,00	24.361.326.100	0,01%	4423,65	2,64	4426,29	3788
R1000B	26.133.482.089,00	26.135.545.100	0,01%	4057,85	2,44	4060,29	3640
R1000C	26.763.099.454,00	26.765.043.000	0,01%	4632,58	2,25	4634,83	4047
R1000D	24.166.402.169,00	24.168.673.600	0,01%	3948,51	1,73	3950,25	3523
R1000E	25.454.412.240,00	25.456.467.000	0,01%	4203,83	2,20	4206,03	3759

Table 8. Results for the 2D MSSCSP-VD (variable height, Benati 'R' instances), 2-stage, non-oriented

5.2. 2D MSSCSP-VD (Variable Width and Height) instances

To the best of our knowledge this is the first time that a 2D MSSCSP-VD with both variable width and height is tackled in the literature, so no benchmarks exist for this it. Nonetheless, we generated a set of 9 instances based on information gathered from the steel company. These instances are all labeled "vd" (for variable dimensions) followed by the number of items under consideration and the size of the stock sheets ("s" for small, "m" for medium, "l" for large). All instances consider 3 stock sheets. Stock sheet ranges for small instances are randomly selected from the interval [1000, 1100], for medium instances from the interval [1500, 1650], and for large instances from the interval [2000,2200]. Stock sheet costs are all the same and equal to 1, so the objective is to minimize waste. The item height and width are randomly selected from the interval [100, 1000] and the demand from [1000, 20000]. We tried 4 parameter combinations (non/2-stage, rotations), for a total of 36 instances. We assume that the set of initial cuts can be done in any direction. Tables 9-12 summarize the results.

Results are consistent with those for the variable height variant and prove that our heuristic has an excellent performance in the case of variable width and height as well. Gaps continue to be very small, showing that solutions are of very good quality. Across all parameter combinations, the average gap is less than 0.01% for "s" instances and 0.01% for both "m" and "l" instances. Not only does this show that the gap is in fact almost negligible, but also that it is not dependent on the size of the problem at hand. The number of generated columns averages 39 for small instances, 54 for medium instances, and 62 for large instances. Running time has a larger variability: while "s" instances are easily solved and run in 72 s on average, this value increases to 935 s in the case of "m" instances, and to 3880 s in the case of "l" instances. The most extreme case is instance vd30l with 2 stages and no orientation, which takes more than 12,500 s (roughly 3.5 h) to solve. The reason behind this is that the real world instances faced by the steel company are much more difficult than ones previously published in the literature, such as those by Benati. Assuming that dimensions are measured in mm, Benati's instances only deal with stock sheets whose sizes are at most around 1,000 mm x 1,000 mm (and generally much less); this is approximately the size of the "s" instances we designed, and as can be seen these are solved quite easily. However, we increase this size up to more than 2,000 mm x

2,000 mm ("l" instances), and this makes the subproblem more difficult to solve via dynamic programming as the space of valid rectangles grows significantly.

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vd10s	39,738,168,722.83	39,738,589,650	0.00%	0.05	0.02	0.06	10
vd10m	29,016,197,692.77	29,019,113,427	0.01%	1.02	0.01	1.03	14
vd10l	27,776,733,128.11	27,779,605,557	0.01%	19.95	0.39	20.34	25
vd20s	46,471,009,063.14	46,473,059,881	0.00%	12.84	0.02	12.86	49
vd20m	42,857,762,538.06	42,861,905,714	0.01%	328.97	17.76	346.73	69
vd20l	42,772,364,054.09	42,777,686,318	0.01%	1601.43	599.92	2201.34	63
vd30s	111,591,923,926.00	111,592,955,160	0.00%	25.47	0.01	25.48	63
vd30m	98,137,757,680.39	98,148,665,859	0.01%	564.27	0.05	564.32	87
vd30l	97,836,953,528.87	97,844,365,678	0.01%	3348.15	251.89	3600.04	112

Table 9. Results for the 2D MSSCSP-VD (variable width and height), non-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vd10s	39,738,168,722.83	39,738,589,650	0.00%	0.11	0.02	0.13	9
vd10m	29,030,498,879.80	29,031,910,687	0.00%	3.39	0.03	3.42	15
vd10l	28,422,038,022.72	28,425,291,432	0.01%	48.25	0.16	48.41	21
vd20s	47,234,713,067.19	47,236,246,160	0.00%	29.97	0.01	29.98	35
vd20m	43,609,713,390.45	43,613,252,530	0.01%	696.45	0.45	696.90	53
vd20l	43,342,219,715.48	43,349,776,388	0.02%	2556.00	599.38	3155.38	43
vd30s	111,591,923,926.00	111,592,955,160	0.00%	91.44	0.02	91.45	63
vd30m	98,867,852,709.24	98,877,687,495	0.01%	1272.19	0.17	1272.36	75
vd30l	98,756,077,067.64	98,764,450,710	0.01%	6009.07	177.15	6186.22	85

Table 10. Results for the 2D MSSCSP-VD (variable width and height), 2-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vd10s	36,836,258,740.00	36,836,258,740	0.00%	0.45	0.02	0.47	9
vd10m	27,637,864,284.50	27,639,751,550	0.01%	21.66	0.02	21.67	20
vd10l	27,517,988,834.64	27,520,074,676	0.01%	286.43	0.05	286.47	34
vd20s	44,388,202,018.75	44,391,287,835	0.01%	71.93	0.02	71.95	54
vd20m	42,628,862,141.18	42,633,280,123	0.01%	862.89	599.74	1462.63	58
vd20l	42,617,915,250.73	42,625,478,177	0.02%	3148.21	599.70	3747.91	63
vd30s	110,624,754,486.00	110,627,848,188	0.00%	111.07	0.01	111.08	58
vd30m	97,612,647,319.25	97,619,567,501	0.01%	1613.20	101.74	1714.95	99
vd30l	97,437,189,492.27	97,450,856,517	0.01%	6818.38	599.49	7417.86	123

Table 11. Results for the 2D MSSCSP-VD (variable width and height), non-stage, non-oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols
vd10s	36,836,258,740.00	36,836,258,740	0.00%	2.11	0.02	2.13	12
vd10m	27,923,782,185.43	27,927,035,496	0.01%	74.45	0.02	74.46	21
vd10l	27,766,575,613.70	27,768,510,304	0.01%	600.15	0.06	600.22	27
vd20s	45,012,134,352.48	45,014,496,245	0.01%	195.41	0.02	195.43	46
vd20m	43,104,063,937.11	43,109,541,409	0.01%	1947.39	1.52	1948.90	55
vd20l	42,862,486,776.94	42,867,399,866	0.01%	6011.66	0.41	6012.06	49
vd30s	110,624,754,486.00	110,626,845,318	0.00%	328.26	0.20	328.47	54
vd30m	97,909,836,682.71	97,919,878,244	0.01%	3109.59	0.20	3109.79	78
vd30l	97,785,223,824.00	97,798,372,774	0.01%	12685.92	599.19	13285.10	94

Table 12. Results for the 2D MSSCSP-VD (variable width and height), 2-stage, non-oriented

As discussed in Section 3, the 2D MSSCSP-VD generalizes several cutting stock and bin packing problems. Therefore, we took advantage of the flexibility of our heuristic and assessed its performance by testing it on a large set of instances for the related 2D SSSCSP and 2D MSSCSP. In the following subsections, we present results for our approach on each of these two problems.

5.3. 2D SSSCSP instances

We tested our heuristic on the set of 2D SSSCSP instances proposed by Cintra et al. (2008) and compared our results with those obtained by their related column generation based heuristic. The benchmarks comprise 12 instances and 6 parameter combinations (non/2/4-stage, oriented/non-oriented), for a total of 72 instances. The instances assume the initial set of cuts is made on the horizontal direction. It is worth recalling that although both approaches are conceptually very similar, ours differs in that to obtain an integer solution we solve optimally over the set of generated patterns instead of using the rounding-down, residual-instance strategy. The results are summarized in Tables 13-18. Since in the 2D SSSCSP there is only one stock sheet, in these tables columns labeled "Sheets" refer to the results in terms of sheets used instead of area. Also, instances for which the solution produced by our heuristic is equal to the previous one are marked with a dash in column "Gap BKS".

The results show that our approach works well on this problem as well. Out of 72 instances, we improve the integer solutions for 12 of them and obtain the same one on another 50, reducing the average gap with the relaxation across all instances from 0.19% to 0.18%. The average improvement over previous solutions is given by the average of column "Gap BKS" and equal to 0.03%. However, it is worth noting that our implementation requires significantly less effort, generating only 39 columns (on average) versus 468 in their case, with an average running time (linear relaxation plus integer program) of just 5.4 s. The drastic reduction is explained by the fact that while our approach applies column generation only once, theirs uses it once for each residual instance that is solved.

Instance	LB (Area)	Sol (Area)	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	LB (Sheets)	Sol (Sheets)	Cols Cintra	Sol Cintra (Sheets)	Gap BKS
gcut1d	18,328,125.00	18,375,000	0.26%	0.03	0.03	0.06	10	294.0	294	24	294	-
gcut2d	21,515,625.00	21,562,500	0.22%	0.19	0.03	0.22	14	345.0	345	91	345	-
gcut3d	20,718,750.00	20,750,000	0.15%	0.72	0.38	1.09	50	332.0	332	314	333	0.30%
gcut4d	52,239,583.33	52,312,500	0.14%	1.19	0.02	1.20	50	836.0	837	1031	837	-
gcut5d	49,208,333.33	49,500,000	0.59%	0.05	0.02	0.06	10	197.0	198	29	198	-
gcut6d	85,666,666.67	86,000,000	0.39%	0.20	0.02	0.22	25	343.0	344	115	344	-
gcut7d	147,750,000.00	148,000,000	0.17%	0.42	0.02	0.44	25	591.0	592	158	592	-
gcut8d	172,500,000.00	172,750,000	0.14%	5.81	0.59	6.41	93	690.0	691	670	692	0.14%
gcut9d	130,666,666.67	132,000,000	1.01%	0.06	0.02	0.08	14	131.0	132	44	132	-
gcut10d	293,000,000.00	294,000,000	0.34%	0.13	0.02	0.14	12	293.0	294	47	293	-0.34%
gcut11d	329,375,000.00	330,000,000	0.19%	2.28	0.09	2.37	42	330.0	330	358	331	0.30%
gcut12d	671,500,000.00	672,000,000	0.07%	8.97	0.11	9.08	66	672.0	672	674	672	-

Table 13. Results for the 2D SSSCSP, non-stage, oriented

Instance	LB (Area)	Sol (Area)	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	LB (Sheets)	Sol (Sheets)	Cols Cintra	Sol Cintra (Sheets)	Gap BKS
gcut1d	18,390,625.00	18,437,500	0.25%	0.03	0.03	0.06	6	295.0	295	21	295	-
gcut2d	21,515,625.00	21,562,500	0.22%	0.25	0.02	0.27	17	345.0	345	173	345	-
gcut3d	21,365,056.82	21,500,000	0.63%	1.02	0.06	1.08	62	342.0	344	534	343	-0.29%
gcut4d	52,781,250.00	52,875,000	0.18%	1.41	0.02	1.42	53	845.0	846	1506	845	-0.12%
gcut5d	51,666,666.67	52,000,000	0.64%	0.05	0.01	0.06	8	207.0	208	21	207	-0.48%
gcut6d	93,666,666.67	93,750,000	0.09%	0.16	0.06	0.22	16	375.0	375	86	375	-
gcut7d	149,875,000.00	150,250,000	0.25%	0.55	0.08	0.63	28	600.0	601	357	600	-0.17%
gcut8d	179,812,500.00	180,250,000	0.24%	5.45	0.23	5.69	77	720.0	721	693	720	-0.14%
gcut9d	134,250,000.00	136,000,000	1.29%	0.06	0.02	0.08	12	135.0	136	57	135	-0.74%
gcut10d	314,833,333.33	315,000,000	0.05%	0.30	0.01	0.31	24	315.0	315	122	315	-
gcut11d	348,125,000.00	350,000,000	0.54%	2.22	0.03	2.25	33	349.0	350	289	349	-0.29%
gcut12d	674,562,500.00	676,000,000	0.21%	11.27	0.02	11.28	71	675.0	676	1167	676	-

Table 14. Results for the 2D SSSCSP, 2-stage, oriented

Instance	LB (Area)	Sol (Area)	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	LB (Sheets)	Sol (Sheets)	Cols Cintra	Sol Cintra (Sheets)	Gap BKS
gcut1d	18,328,125.00	18,375,000	0.26%	0.03	0.01	0.04	8	294.0	294	25	294	-
gcut2d	21,515,625.00	21,562,500	0.22%	0.03	0.02	0.05	15	345.0	345	157	345	-
gcut3d	20,718,750.00	20,750,000	0.15%	0.98	0.06	1.05	45	332.0	332	621	333	0.30%
gcut4d	52,239,583.33	52,312,500	0.14%	1.66	0.02	1.67	46	836.0	837	1606	837	-
gcut5d	49,208,333.33	49,500,000	0.59%	0.05	0.01	0.06	9	197.0	198	40	198	-
gcut6d	85,666,666.67	86,000,000	0.39%	0.03	0.02	0.05	24	343.0	344	136	344	-
gcut7d	147,750,000.00	148,000,000	0.17%	0.64	0.02	0.66	24	591.0	592	295	592	-
gcut8d	172,500,000.00	172,750,000	0.14%	8.45	0.11	8.56	83	690.0	691	1539	691	-
gcut9d	130,666,666.67	132,000,000	1.01%	0.06	0.02	0.08	10	131.0	132	55	131	-0.76%
gcut10d	293,000,000.00	294,000,000	0.34%	0.19	0.03	0.22	12	293.0	294	93	294	-
gcut11d	329,375,000.00	330,000,000	0.19%	4.75	0.02	4.77	47	330.0	330	535	330	-
gcut12d	671,500,000.00	672,000,000	0.07%	13.86	0.03	13.89	59	672.0	672	927	673	0.15%

Table 15. Results for the 2D SSSCSP, 4-stage, oriented

Instance	LB (Area)	Sol (Area)	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	LB (Sheets)	Sol (Sheets)	Cols Cintra	Sol Cintra (Sheets)	Gap BKS
gcut1d	18,140,625.00	18,187,500	0.26%	0.03	0.02	0.05	4	291.0	291	30	291	-
gcut2d	17,617,187.50	17,687,500	0.40%	0.56	0.03	0.59	20	282.0	283	214	283	-
gcut3d	19,535,361.84	19,562,500	0.14%	2.69	0.03	2.72	92	313.0	313	918	314	0.32%
gcut4d	52,218,750.00	52,250,000	0.06%	1.72	0.02	1.74	36	836.0	836	1172	836	-
gcut5d	43,489,583.33	43,750,000	0.60%	0.20	0.02	0.22	13	174.0	175	61	175	-
gcut6d	75,125,000.00	75,250,000	0.17%	1.08	0.03	1.11	32	301.0	301	117	302	0.33%
gcut7d	135,500,000.00	135,500,000	0.00%	2.23	0.03	2.27	40	542.0	542	357	543	0.18%
gcut8d	162,306,547.62	162,750,000	0.27%	14.97	0.13	15.09	103	650.0	651	811	651	-
gcut9d	121,923,076.92	123,000,000	0.88%	0.28	0.02	0.30	14	122.0	123	53	123	-
gcut10d	269,500,000.00	270,000,000	0.19%	1.72	0.02	1.73	23	270.0	270	93	270	-
gcut11d	297,388,888.89	299,000,000	0.54%	17.81	0.17	17.99	71	298.0	299	674	299	-
gcut12d	601,000,000.00	602,000,000	0.17%	41.28	0.03	41.31	95	601.0	602	965	601	-0.17%

Table 16. Results for the 2D SSSCSP, non-stage, non-oriented

Instance	LB (Area)	Sol (Area)	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	LB (Sheets)	Sol (Sheets)	Cols Cintra	Sol Cintra (Sheets)	Gap BKS
gcut1d	18,140,625.00	18,187,500	0.26%	0.03	0.02	0.05	4	291.0	291	26	291	-
gcut2d	17,617,187.50	17,625,000	0.04%	0.75	0.02	0.77	25	282.0	282	359	283	0.35%
gcut3d	19,746,459.36	19,812,500	0.33%	2.81	0.03	2.84	87	316.0	317	1023	317	-
gcut4d	52,218,750.00	52,312,500	0.18%	2.17	0.02	2.19	42	836.0	837	1523	837	-
gcut5d	43,598,484.85	43,750,000	0.35%	0.27	0.02	0.28	13	175.0	175	45	175	-
gcut6d	75,285,714.29	75,500,000	0.28%	1.31	0.03	1.34	33	302.0	302	166	302	-
gcut7d	135,500,000.00	135,500,000	0.00%	2.48	0.05	2.53	39	542.0	542	193	543	0.18%
gcut8d	162,306,547.62	162,500,000	0.12%	16.16	0.08	16.24	101	650.0	650	630	650	-
gcut9d	125,000,000.00	126,000,000	0.79%	0.41	0.02	0.42	16	125.0	126	61	126	-
gcut10d	269,500,000.00	270,000,000	0.19%	1.91	0.02	1.92	22	270.0	270	177	271	0.37%
gcut11d	298,215,909.09	299,000,000	0.26%	22.30	0.05	22.34	78	299.0	299	677	300	0.33%
gcut12d	601,000,000.00	602,000,000	0.17%	41.94	0.03	41.97	88	601.0	602	1207	602	-

Table 17. Results for the 2D SSSCSP, 2-stage, non-oriented

Instance	LB (Area)	Sol (Area)	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	LB (Sheets)	Sol (Sheets)	Cols Cintra	Sol Cintra (Sheets)	Gap BKS
gcut1d	18,140,625.00	18,187,500	0.26%	0.05	0.02	0.06	4	291.0	291	26	291	-
gcut2d	17,617,187.50	17,687,500	0.40%	0.92	0.01	0.93	20	282.0	283	274	283	-
gcut3d	19,535,361.84	19,562,500	0.14%	3.99	0.23	4.22	90	313.0	313	1103	314	0.32%
gcut4d	52,218,750.00	52,312,500	0.18%	2.64	0.02	2.66	38	836.0	837	1446	836	-0.12%
gcut5d	43,489,583.33	43,750,000	0.60%	0.36	0.02	0.38	12	174.0	175	65	175	-
gcut6d	75,125,000.00	75,250,000	0.17%	1.91	0.05	1.95	34	301.0	301	230	302	0.33%
gcut7d	135,500,000.00	135,500,000	0.00%	3.56	0.03	3.59	40	542.0	542	568	542	-
gcut8d	162,306,547.62	162,750,000	0.27%	26.47	0.13	26.60	117	650.0	651	1159	651	-
gcut9d	121,923,076.92	123,000,000	0.88%	0.50	0.02	0.52	14	122.0	123	58	123	-
gcut10d	269,500,000.00	270,000,000	0.19%	2.52	0.02	2.53	20	270.0	270	109	270	-
gcut11d	297,388,888.89	298,000,000	0.21%	28.83	0.08	28.91	64	298.0	298	996	299	0.33%
gcut12d	601,000,000.00	601,000,000	0.00%	60.25	0.02	60.27	85	601.0	601	1535	602	0.17%

Table 18. Results for the 2D SSSCSP, 4-stage, non-oriented

5.4. 2D MSSCSP instances

In addition to the tests conducted on the two variants of 2D MSSCSP-VD and the 2D SSSCSP, we also tried our approach on the instances used by Cintra et al. (2008) for the 2D MSSCSP and compared our results against those reported in their work. The testbed is comprised of 12 instances and considers 6 parameter combinations (non/2/4-stage, oriented/non-oriented), for a total of 72 instances. Again, the assumption is made that the initial set of cuts is made on the horizontal direction.

The results in tables 19-24 show that our approach is also competitive for tackling the 2D MSSSCP. Out of the 72 instances, our approach improved 71 of the best known solutions. This demonstrates that not only is the approach flexible (as shown by tests on the 2D MSSCSP-VD, 2D SSSCSP, and 2D MSSCSP) but also that it is able to produce solutions of excellent quality for each of these, a feature that not many related approaches can claim. The average gap with the relaxation was reduced from the 0.52% reported by Cintra et al. to 0.15%, with an average improvement of 0.37% on each instance. In addition, the smaller effort required by our approach compared to that of Cintra et al. becomes more evident in these benchmarks: 61 columns in our case (on average) versus 9,060 in theirs (on average). On the other hand, the running time (column generation plus integer program) increases slightly when compared to the results for the 2D SSSCSP, reporting an average of 84.5 s.

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Cols Cintra	Sol Cintra	Gap BKS
gcut1d	14,822,812.50	14,875,000	0.35%	0.22	0.27	0.49	12	309	14,880,000	0.03%
gcut2d	15,673,933.19	15,704,375	0.19%	0.53	7.30	7.83	33	2771	15,733,125	0.18%
gcut3d	19,769,831.29	19,786,875	0.09%	1.33	13.97	15.30	71	15148	19,930,000	0.72%
gcut4d	46,257,549.22	46,274,375	0.04%	3.74	0.27	4.00	114	25871	46,346,250	0.16%
gcut5d	41,517,000.00	41,525,000	0.02%	0.05	0.02	0.06	12	499	41,737,500	0.51%
gcut6d	73,967,812.50	74,050,000	0.11%	0.42	0.69	1.11	36	1505	74,187,500	0.19%
gcut7d	122,295,271.74	122,392,500	0.08%	1.30	599.41	600.71	54	3273	122,735,000	0.28%
gcut8d	155,221,710.77	155,337,500	0.07%	11.31	5.86	17.17	115	11933	155,602,500	0.17%
gcut9d	128,389,230.77	129,080,000	0.54%	0.11	1.30	1.41	15	1038	129,360,000	0.22%
gcut10d	252,565,036.23	252,960,000	0.16%	0.61	1.50	2.11	34	1641	254,130,000	0.46%
gcut11d	292,879,166.67	293,840,000	0.33%	5.33	599.94	605.27	51	7147	295,250,000	0.48%
gcut12d	599,851,250.00	599,970,000	0.02%	33.67	6.97	40.64	123	7392	602,240,000	0.38%

Table 19. Results for the 2D MSSCSP, non-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Cols Cintra	Sol Cintra	Gap BKS
gcut1d	14,822,812.50	14,875,000	0.35%	0.08	0.13	0.20	12	397	14,880,000	0.03%
gcut2d	16,740,781.25	16,805,625	0.39%	0.78	0.98	1.77	28	492	16,820,625	0.09%
gcut3d	20,149,803.57	20,185,625	0.18%	2.22	0.53	2.75	59	7877	20,267,500	0.40%
gcut4d	46,523,511.25	46,535,625	0.03%	6.58	0.17	6.75	118	11569	46,591,875	0.12%
gcut5d	41,667,500.00	41,727,500	0.14%	0.06	0.02	0.08	10	110	42,022,500	0.70%
gcut6d	77,621,562.50	77,877,500	0.33%	0.36	0.14	0.50	20	539	78,167,500	0.37%
gcut7d	123,946,562.50	124,217,500	0.22%	1.75	599.54	601.29	47	1316	124,257,500	0.03%
gcut8d	161,074,884.11	161,117,500	0.03%	19.06	9.80	28.86	105	3958	161,575,000	0.28%
gcut9d	130,802,500.00	131,860,000	0.80%	0.07	0.07	0.14	11	86	131,830,000	-0.02%
gcut10d	260,444,166.67	261,330,000	0.34%	0.73	1.91	2.64	28	434	262,470,000	0.43%
gcut11d	303,137,516.56	303,920,000	0.26%	8.42	15.11	23.53	45	6926	304,440,000	0.17%
gcut12d	609,519,416.67	610,210,000	0.11%	50.28	2.55	52.83	114	5452	611,230,000	0.17%

Table 20. Results for the 2D MSSCSP, 2-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Cols Cintra	Sol Cintra	Gap BKS
gcut1d	14,822,812.50	14,875,000	0.35%	0.09	0.13	0.22	12	307	14,880,000	0.03%
gcut2d	15,673,933.19	15,688,125	0.09%	1.33	0.50	1.83	33	3163	15,730,625	0.27%
gcut3d	19,769,831.29	19,786,250	0.08%	3.59	10.30	13.89	72	12327	19,864,375	0.39%
gcut4d	46,257,549.22	46,274,375	0.04%	7.34	0.24	7.58	103	18410	46,343,750	0.15%
gcut5d	41,517,500.00	41,525,000	0.02%	0.08	0.02	0.09	12	489	41,737,500	0.51%
gcut6d	73,967,812.50	74,050,000	0.11%	0.92	0.66	1.58	38	1005	74,187,500	0.19%
gcut7d	122,295,271.74	122,380,000	0.07%	2.78	599.97	602.75	52	3715	122,745,000	0.30%
gcut8d	155,221,710.77	155,332,500	0.07%	27.97	6.50	34.47	129	17864	155,832,500	0.32%
gcut9d	128,389,230.77	129,080,000	0.54%	0.16	1.36	1.52	15	727	129,360,000	0.22%
gcut10d	252,565,036.23	252,890,000	0.13%	1.25	0.63	1.88	34	1649	254,130,000	0.49%
gcut11d	292,879,166.67	293,720,000	0.29%	14.92	599.42	614.34	56	8413	294,200,000	0.16%
gcut12d	599,851,250.00	600,070,000	0.04%	85.14	2.42	87.56	123	7886	602,360,000	0.38%

Table 21. Results for the 2D MSSCSP, 4-stage, oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Cols Cintra	Sol Cintra	Gap BKS
gcut1d	13,790,625.00	13,814,375	0.17%	0.14	0.13	0.27	13	402	13,823,750	0.07%
gcut2d	15,083,409.09	15,086,875	0.02%	1.99	0.17	2.16	47	2714	15,160,625	0.49%
gcut3d	19,118,423.47	19,121,250	0.01%	4.11	6.72	10.83	94	11627	19,241,875	0.63%
gcut4d	44,575,105.30	44,581,250	0.01%	9.58	6.60	16.18	135	26535	44,663,125	0.18%
gcut5d	38,454,765.63	38,582,500	0.33%	0.36	0.09	0.45	14	2175	38,890,000	0.79%
gcut6d	69,599,732.14	69,615,000	0.02%	1.91	0.14	2.05	40	2646	70,192,500	0.82%
gcut7d	114,503,487.94	114,527,500	0.02%	5.94	2.14	8.08	72	12951	114,867,500	0.30%
gcut8d	151,462,312.94	151,480,000	0.01%	27.94	13.22	41.16	137	47421	152,262,500	0.51%
gcut9d	118,806,666.67	119,030,000	0.19%	0.52	0.11	0.63	17	1093	119,730,000	0.58%
gcut10d	246,552,500.00	246,700,000	0.06%	5.16	0.22	5.38	45	3379	248,620,000	0.77%
gcut11d	281,713,516.65	281,780,000	0.02%	30.19	4.17	34.36	82	19249	283,780,000	0.70%
gcut12d	559,820,015.84	560,280,000	0.08%	82.93	0.72	83.64	132	32690	561,680,000	0.25%

Table 22. Results for the 2D MSSCSP, non-stage, non-oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Cols Cintra	Sol Cintra	Gap BKS
gcut1d	13,828,125.00	13,849,375	0.15%	0.16	0.05	0.20	11	416	13,908,750	0.43%
gcut2d	15,432,371.32	15,458,125	0.17%	2.64	0.61	3.25	43	4616	15,474,375	0.11%
gcut3d	19,310,805.27	19,340,625	0.15%	5.16	599.70	604.86	79	12159	19,436,875	0.50%
gcut4d	44,767,392.35	44,779,375	0.03%	18.72	1.94	20.66	134	21902	44,905,000	0.28%
gcut5d	40,087,187.50	40,170,000	0.21%	0.56	0.63	1.19	13	341	40,382,500	0.53%
gcut6d	70,839,625.00	70,955,000	0.16%	3.19	2.88	6.06	37	2411	71,162,500	0.29%
gcut7d	114,817,716.28	114,870,000	0.05%	10.55	52.72	63.27	72	13326	115,312,500	0.38%
gcut8d	152,634,892.31	152,650,000	0.01%	58.28	10.19	68.47	129	28128	153,410,000	0.50%
gcut9d	119,568,000.00	120,670,000	0.91%	0.70	3.14	3.84	14	756	121,040,000	0.31%
gcut10d	247,872,857.14	248,060,000	0.08%	10.03	0.52	10.55	48	1545	249,260,000	0.48%
gcut11d	286,973,906.44	287,080,000	0.04%	48.08	20.19	68.27	74	23447	289,430,000	0.81%
gcut12d	562,898,801.25	563,250,000	0.06%	180.50	599.71	780.21	129	28565	564,650,000	0.25%

Table 23. Results for the 2D MSSCSP, 2-stage, non-oriented

Instance	LB	Sol	Gap	CG Time (s)	BB Time (s)	Time (s)	Cols	Cols Cintra	Sol Cintra	Gap BKS
gcut1d	13,790,625.00	13,814,375	0.17%	0.25	0.11	0.36	13	415	13,823,750	0.07%
gcut2d	15,083,409.09	15,086,875	0.02%	4.16	0.13	4.28	47	3010	15,161,875	0.49%
gcut3d	19,118,423.47	19,120,625	0.01%	7.58	2.81	10.39	85	16255	19,181,875	0.32%
gcut4d	44,575,105.27	44,581,250	0.01%	20.00	1.34	21.34	133	27104	44,723,750	0.32%
gcut5d	38,454,765.63	38,582,500	0.33%	0.80	0.16	0.95	14	1662	38,890,000	0.79%
gcut6d	69,599,732.14	69,627,500	0.04%	4.69	0.48	5.17	41	2639	70,192,500	0.80%
gcut7d	114,503,487.94	114,527,500	0.02%	14.66	3.91	18.56	72	12339	114,867,500	0.30%
gcut8d	151,462,312.94	151,470,000	0.01%	62.17	4.22	66.39	133	30285	151,745,000	0.18%
gcut9d	118,806,666.67	119,030,000	0.19%	1.11	0.08	1.19	17	1198	119,730,000	0.58%
gcut10d	246,552,500.00	246,800,000	0.10%	10.59	5.61	16.20	42	3409	248,620,000	0.73%
gcut11d	281,851,974.22	281,950,000	0.03%	63.16	414.94	478.10	74	27379	283,560,000	0.57%
gcut12d	559,820,015.84	560,230,000	0.07%	240.08	1.45	241.53	145	32554	561,640,000	0.25%

Table 24. Results for the 2D MSSCSP, 4-stage, non-oriented

6. Conclusions and Future Work

In the present work we studied a Two-Dimensional Multiple Stock Size Cutting Stock Problem with Variable Dimensions (2D MSSCSP-VD). Despite its wide practical applicability, evidenced by two applications in the cardboard and steel industries, academic research on the 2D MSSCSP-VD is rather scarce. To tackle the problem we proposed an approach based on traditional column generation strategies, formulating it as a set covering problem, solving its linear relaxation and then optimally solving the corresponding integer program considering only the set of generated columns. At each iteration of the column generation step, attractive patterns were created by extending the applicability of discretization points to the case of variable dimensions and using an existing dynamic programming algorithm to solve the subproblem.

Because of the industrial motivation of the problem, the approach needed to be both flexible (to tackle slight differences in problem conditions such as those between the cardboard and steel companies) and effective (to be competitive against other proposed solutions). Therefore, we tested our implementation on instances of the 2D MSSCSP-VD with variable height, 2D MSSCSP-VD with variable width and height, 2D SSSCSP, and 2D MSSCSP. Computational experiments carried out on a large set of instances from the literature show that the proposed method is competitive when solving several cutting stock problems.

For the 2D MSSCSP-VD with variable height we compared our approach against two different set of instances. The first set was designed by us based on data from the cardboard company and the results of the approach were excellent, with the average gaps being almost negligible. The second set is to the best of our knowledge the only one that previously existed in the literature. We presented results for all 33 instances of this set, obtaining near-optimal solutions in all cases. Furthermore, in very competitive times we significantly improved the solutions for the 18 instances which are publicly available.

Although there are no benchmarks for the 2D MSSCSP-VD with variable width and height, we once more designed a new set of instances for this problem based on the data from the steel company. Given the size of these instances, they proved to be much harder: some of them took as much as almost four hours to solve. However, running times for instances similar in size to those in the literature were still small. In all cases, gaps were proven to be almost negligible, repeatedly averaging approximately 0.01%.

Due to the theoretical flexibility of the approach, we decided to take it further and test it on related problems that can be reduced to the 2D MSSCSP-VD and assess its performance on them. Therefore, we chose two closely related problems for this purpose, namely, the 2D SSSCSP and the 2D MSSCSP. For the 2D SSSCSP, out of 72 instances found in the literature we improved the integer

solutions for 12 of them and obtained the same results in another 50 when compared to a recent heuristic that shares many of the components embedded in our approach. For the 2D MSSCSP we clearly and consistently outperformed that same heuristic: out of 72 instances, we improved the integer solutions for 71 of them. We again obtained small integrality gaps regardless of the problem's difficulty: an average of 0.30% for the 2D SSSCSP instances and 0.15% for the 2D MSSCSP instances. In conclusion, the proposed approach is flexible, in terms of the different types of problems it can handle, accurate, and fast.

Much work can be envisioned to improve the current approach. For instance, to accelerate the column generation step, one could rely on heuristics to generate feasible patterns instead of running the full dynamic programming algorithm most of the time. This would result, however, in a tradeoff between the time employed to generate the patterns and the time taken to converge to the linear optimum (and eventually the integer one as well) that should be evaluated.

Second, the current approach may not behave well with bin packing problems for the following reason. If the stock sheet size is large relative to the size of the items (which is generally the case), the dynamic algorithm will presumably fit many of the attractive items into the pattern. However, since in bin packing problems only one or very few of each item is demanded, these patterns will translate into poor utilization ratios when solving the resulting integer program. Hence, the number of items that are available could be restricted to the demand, or some other defined value, when solving the subproblem.

Third, we are currently assuming that a pattern's cost is given only by its height, width, and the area unit cost of the stock sheet out of which it is cut. However, our formulation is flexible and allows assigning any cost to a pattern when calculating its reduced cost. This could be exploited in situations where additional considerations need to be taken into account, i.e., when a pattern's cost is not simply proportional to its area.

Finally, we hope for an optimal method to be built on top of the current approach. One possibility would be to incorporate it into a branch-and-price approach, solving the linear relaxation at each node of the branch-and-bound tree via column generation. This would, however, be prohibitive in practical terms for the more difficult instances, as most likely it would take an excessive amount of time to solve the tree. On the other hand, small instances such as those by Cintra, Benati, and the 's' set could possibly be tackled this way.

References

Beasley, J.E. (1985). Algorithms for unconstrained two-dimensional guillotine cutting. Journal of the Operational Research Society, Volume 36, No. 4, pp. 297-306.

Benati, S. (1997). An Algorithm for a Cutting Stock Problem on a Strip. Journal of the Operational Research Society, Volume 48, No. 3, pp. 288-294.

Bettinelli, A., Ceselli, A., and Righini, G. (2008). A branch-and-price algorithm for the two-dimensional level strip packing problem. 4OR, Volume 6, No. 4, pp. 361-374.

Cintra, G.F., Miyazawa, F.K., Wakabayashi, Y., and Xavier E.C. (2008). Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. European Journal of Operational Research, Volume 191, Issue 1, pp. 61-85.

Cui, Y. and Lu, Y. (2009). Heuristic algorithm for a cutting stock problem in the steel bridge construction. Computers & Operations Research, Volume 36, Issue 2, pp. 612-622.

Dyckoff, H. (1990). A typology of cutting and packing problems. European Journal of Operational Research, Volume 44, Issue 2, pp. 145-159.

Gilmore P. and Gomory, R. (1961). A linear programming approach to the cutting stock problem. Operations Research Volume 9, No. 6, pp. 849–859.

Gilmore P. and Gomory, R. (1963). A linear programming approach to the cutting stock problem – Part II. Operations Research Volume 11, No. 6, pp. 863–888.

Gilmore P. and Gomory, R. (1965). Multistage cutting stock problems of two and more dimensions. Operations Research Volume 13, No. 1, pp. 94–120.

Herz, J.C. (1972). A recursive computational procedure for two-dimensional stock cutting. IBM Journal of Research and Development pp. 462-469.

Kenmochi, M., Imamichi, T., Nonobe, K., Yagiura, M., and Nagamochi, H. (2009). Exact algorithms for the two-dimensional strip packing problem with and without rotations. European Journal of Operational Research, Volume 198, Issue 1, pp. 73-83.

Lodi, A., Martello, S., and Vigo, D. (2002). Recent advances on two-dimensional bin packing problems. Discrete Applied Mathematics, Volume 123, Issues 1-3, pp. 379-396.

Macedo, R., Alves, C., Valério de Carvalho, J.M. (2010) (Forthcoming). Arc-flow model for the two-dimensional guillotine cutting stock problem. Computers & Operations Research, Volume 37, Issue 6, pp. 991-1001.

Martello, S., Monaci, M., and Vigo, D. (2003). An Exact Approach to the Strip Packing Problem. INFORMS Journal on Computing, Volume 15, No. 3, pp. 310-319.

Oliveira, J.F. and Wäscher, G. (2007). Editorial: Cutting and Packing. European Journal of Operational Research, Volume 183, Issue 3, pp. 1106-1108.

Pisinger, D. and Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. Discrete Optimization, Volume 2, Issue 2, pp. 154-167.

Pisinger, D. and Sigurd, M. (2007). Using Decomposition Techniques and Constraint Programming for Solving the Two-Dimensional Bin-Packing Problem. INFORMS Journal on Computing, Volume 19, No. 1, pp. 36-51.

Riff, M.C., Bonnaire, X., and Neveu, B. (2009). A revision of recent approaches for two-dimensional strip packing problems. Engineering Applications of Artificial Intelligence, Volume 22, Issues 4-5, pp. 823-827.

Silva, E., Alvelos, F., and Valério de Carvalho, J.M. (2010) (Forthcoming). An integer programming model for two- and three-stage two-dimensional cutting stock problems. European Journal of Operational Research, Volume 205, Issue 3, pp. 699-708.

Valério de Carvalho, J.M. (1999). Exact solution of bin packing problems using column generation and branch and bound. Annals of Operations Research, Volume 86, pp. 629-659.

Wäscher, G., Hauβner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. European Journal of Operational Research, Volume 183, Issue 3, pp. 1109-1130.