European Journal of Operational Research 205 (2010) 699–708

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

An integer programming model for two- and three-stage two-dimensional cutting stock problems

Elsa Silva [a], Filipe Alvelos [a,b,*], J.M. Valério de Carvalho [a,b]

[a] Centro de Investigação Algoritmi, Universidade do Minho, Braga, Portugal
[b] Departamento de Produção e Sistemas, Universidade do Minho, Braga, Portugal

ARTICLE INFO

Article history:
Received 31 December 2008
Accepted 21 January 2010
Available online 29 January 2010

Keywords:
Cutting
2D rectangular SSSCSP with guillotine constraints
Integer programming
Length of the cutting operations

ABSTRACT

In this paper, an integer programming model for two-dimensional cutting stock problems is proposed. In the problems addressed, it is intended to cut a set of small rectangular items of given sizes from a set of larger rectangular plates in such a way that the total number of used plates is minimized.

The two-stage and three-stage, exact and non-exact, problems are considered. Other issues are also addressed, as the rotation of items, the length of the cuts and the value of the remaining plates.

The new integer programming model can be seen as an extension of the "one-cut model" proposed by Dyckhoff for the one-dimensional cutting stock problem. In the proposed model, each decision variable is associated with cutting one item from a plate or from a part of a plate resulting from previous cuts (residual plates).

Comparative computational results of the proposed model and of models from the literature are presented and discussed.

1. Introduction

In this paper, an exact approach to two-dimensional cutting stock (2DCS) problems is proposed. The approach is based on the definition of an integer programming model, with a pseudo-polynomial number of variables and constraints, to be optimized directly by a general integer programming solver.

In a 2DCS problem it is intended to cut a set of rectangular items from a set of rectangular plates in such a way that the number of used plates is minimized. The plates are available in (a virtually) infinite number and all have the same dimensions, i.e., the same width and height. A set of items to be cut, grouped by types according to their dimensions (width and height), is given. Each item type is defined by a width, a height and a demand (corresponding to the number of items of the type to be cut).

According to the typology of Wäscher et al. (2007) the problems addressed in this paper are two-dimensional rectangular Single Stock Size Cutting Stock Problems (SSSCSP) with additional constraints related with the types of cuts allowed. In Section 3, the extension to deal with multiple stock sizes (MSSCSP) is also described.

The motivation behind this work lies in the woodcut industry where plates of wood must be cut in pieces (items) to satisfy customer orders. In this industry, due to technological constraints, usually only cuts parallel to the sides of the plate (orthogonal cuts) and from one border to the opposite one (guillotine cuts) are allowed. Furthermore, the number of stages (set of cuts with the same orientation – horizontal or vertical) in which a plate can be cut is also frequently limited to two or three.

In a two-stage problem, the items are obtained by a set of horizontal cuts, dividing the plate in stripes, followed by a set of vertical cuts, separating the different items. If an additional set of horizontal cuts is allowed in order to separate the waste from the items, then the problem is a non-exact two-stage problem (in opposition to the exact case where all the items in a given stripe have the same height). Fig. 1 shows a non-exact two-stage cutting pattern, where the white rectangles correspond to waste and the shaded rectangles to items.

The three-stage problem is also considered in this paper. In this problem, a third set of cuts is allowed: after the plate is cut in stripes (in the first stage), each stripe is then cut in stacks (second stage) and finally the items in each stack are separated by a third set of (horizontal) cuts (third stage). As in the two-stage problem, the exact and non-exact cases are considered. In the latter, an additional set of vertical cuts is allowed for separating the items from the waste. In the former, all the items in a stack must have the same width. In both cases, the restricted version of the three-stage problem is considered, in which, in each stripe, there is always a

* Corresponding author. Address: Departamento de Produção e Sistemas, Universidade do Minho, Braga, Portugal. Tel.: +351 253604751; fax: +351 253604741.
E-mail addresses: elsa@dps.uminho.pt (E. Silva), falvelos@dps.uminho.pt (F. Alvelos), vc@dps.uminho.pt (J.M. Valério de Carvalho).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.01.039

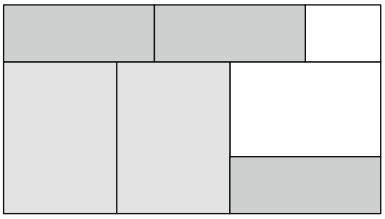

Fig. 1. A non-exact two-stage cutting pattern.

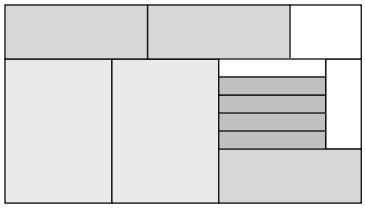

Fig. 2. A non-exact three-stage cutting pattern.

stack with only one item defining its height. See Fig. 2 for an example of a non-exact three-stage cutting pattern.

Previous exact approaches have been proposed for 2DCS problems.

Following their seminal work on the one-dimensional cutting stock problem, Gilmore and Gomory (1965) proposed a column generation algorithm for the two-stage 2DCS problem. As in the one-dimensional case, in the restricted master problem, variables are associated with cutting patterns. The subproblem generates attractive patterns, which, for the two-stage problem, are obtained by solving a sequence of two integer knapsack problems. In Cintra et al. (2008), the subproblem of the column generation algorithm is solved by dynamic programming.

Vanderbeck (2001) also used a column generation based approach for a three-stage 2DCS problem with additional constraints related to the cutting process. The subproblem of generating attractive patterns is also solved by column generation.

Specifically for the two-stage problem, Gilmore and Gomory (1965) also proposed an integer programming model with two sets of decision variables: a set of decision variables related with the definition of the height of the stripes and the other set related with the items included in each stripe. As suggested by Gilmore and Gomory, this model can be solved directly or by applying column generation.

Cutting stock problems are closely related to bin-packing problems. In bin-packing problems there is a strongly heterogeneous assortment of items, i.e., there are many item types with small demands (in the limit, all items are different, being the demands of the item types equal to one). On the other hand, in cutting stock problems, there is a weakly heterogeneous assortment of items, i.e., there is a small number of item types and their demands are large.

Column generation has also been used for bin-packing problems. Two recent approaches were proposed by Pisinger and Sigurd (2007) and Puchinger and Raidl (2007). In the former, the subproblem is solved by constraint programming, allowing to take into account more general patterns, such as non-guillotine cutting. In the latter, the three-stage problem is addressed. The subproblem is solved by a hierarchy of four methods (a greedy heuristic is first applied followed by a genetic algorithm, which is followed by an exact method for a simplified version of the (sub)problem, and, only if no attractive pattern is obtained by the previous methods, is the (sub)problem solved exactly).

Compact integer programming models have also been proposed for bin-packing problems. Since the number of variables and constraints is polynomial with respect to the size of the instance, the implementation step is reduced to inputting the model in an integer programming solver. That is the case of the models proposed in Lodi et al. (2004) for the two-stage problem and extended by Puchinger and Raidl (2007) to the three-stage problem. In both cases, decision variables are related to the assignment of the items to bins, stripes or stacks. Typically the linear relaxation of these "assignment" models provide lower bounds that, in general, are worse than the ones given by the linear relaxation of the models mentioned so far (Gilmore and Gomory, 1965; Cintra et al., 2008; Vanderbeck, 2001; Pisinger and Sigurd, 2007).

A cutting problem related to the one addressed in this paper is the constrained two-dimensional cutting stock problem. In this problem, it is intended to maximize the profit of the items (available in finite number) cut from a single plate. In Belov and Scheithauer (2006), a branch-and-cut-and-price algorithm is proposed for the two-stage non-exact constrained two-dimensional cutting stock problem. In each branch-and-price node, Chvátal-Gomory cuts and Gomory mixed-integer cuts are used to improve the quality of the linear programming relaxation solved. For the same problem, Hifi and Roucairol (2001) proposed exact and heuristic algorithms and Lodi and Monaci (2003) proposed two integer linear programming models. More recently, Cui (2008) presented a branch-and-bound procedure combined with dynamic programming for the homogenous three-stage constrained two-dimensional cutting stock problem.

For a survey on two-dimensional packing problems, the interested reader is referred to Lodi et al. (2002). For more general surveys on cutting and packing problems see Wäscher et al. (2007) and Dyckhoff et al. (1997). There is also a categorized database of publications on cutting and packing available at the "EURO Special Interest Group on Cutting and Packing" site (http://paginas.fe.up.pt/~esicup/).

In this paper, an approach to obtain optimal solutions to two-stage and three-stage 2DCS problems, which is not based on column generation, is proposed. The model is an extension of the one-cut model for the one-dimensional cutting stock problem proposed by Dyckhoff (1981). The approach is based on an integer programming model that can be solved by a general integer programming solver, taking advantage of the efficiency and robustness that (mixed-) integer programming solvers have achieved in recent years. For example, in the software used in this paper for the computational tests – CPLEX 11.0 (see Ilog, 2007) – several classes of valid inequalities and heuristics are implemented to improve the (lower and upper) bounds during the search of the branch-and-bound tree.

The model inherits the possibility of modeling the features already pointed out by Dyckhoff, as multiple types of pieces (plates) and the value of the pieces (plates) for future use. Furthermore, there is a feature of the proposed model, which is very relevant in practice but is not usually taken into account, which is modeling the length (or time) of the cuts needed to obtain all the final items. This issue as well as an extension for the rotation of pieces will be addressed after the presentation of the model.

This paper is organized as follows. In the next section, the integer programming model for the two-stage and three-stage, exact and non-exact, 2DCS problems is introduced. Upper bounds on the number of constraints and on the number of variables of the model are given. The model is based on the enumeration of all the different cuts and residual plates, for which an algorithm is described. Section 2 ends with a small example. In Section 3, extensions of the developed model are discussed and proposed. Computational results are reported and analyzed in Section 4. In Section 5, the main conclusions of this work are drawn.

2. The integer programming model

2.1. Description

The proposed model is based on explicitly considering how the demanded items can be obtained from plates. The main concept of the proposed model is the *cut*, which consists in taking a plate and obtaining one item from it (through one or two guillotine cuts). In general, a cut results in one item (which will not be further cut, since it is demanded), and two plates which can be further cut. Plates resulting from a cut are named *residual plates*.

In order to illustrate these concepts, a small example is now introduced. Consider one instance of the non-exact two-stage 2DCS problem with initial plates with width and height equal to 6 and two item types, indexed by i, with the widths, heights, and demands given in Table 1.

An example of a *cut* is to obtain one item of type one from an initial plate. In this cut, a stripe of height 3 is generated by a horizontal cut and the item of type one is then obtained by a vertical cut in this stripe. As a result, besides the desired item of type one, two new plates with sizes (6,3) and (2,3) are obtained, as illustrated in Fig. 3, where the bold lines represent the cuts. Another example of a cut is to obtain one item of type two from the residual plate with dimensions (6,3) and is illustrated in Fig. 4. As a result, besides the desired item of type two, two new plates with sizes (6,1) and (4,2) are obtained. The first plate is waste since it has a smaller height than the height of both item types.

The proposed model is based on enumerating all cuts and types of residual plates. Plates of the same type have the same width, height, and belong to the same stage. The stage is relevant when defining the type of a plate because, as illustrated in Figs. 5 and 6, a cut yields different residual plates according to the stage where it is performed.

There is a set of simple rules that allows to define the possible cuts, and thus the width, height and stage of the resulting residual plates. In Section 2.3, the details on the generation of cuts and residual plates will be analysed. In this subsection, it is assumed that all possible cuts and the corresponding types of residual plates are known.

Table 1
Data on the item types of the example.

Item type	Width	Height	Demand
1	4	3	5
2	2	2	5

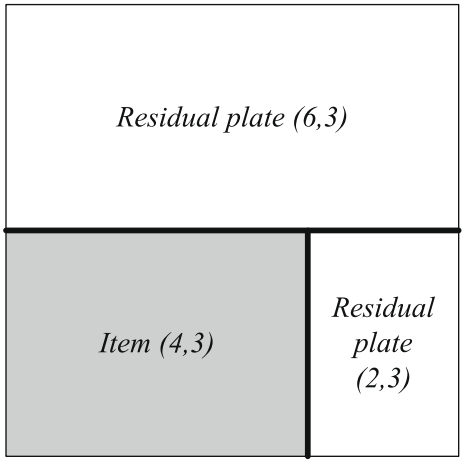

Fig. 3. The cut of one item of type one from an initial plate.

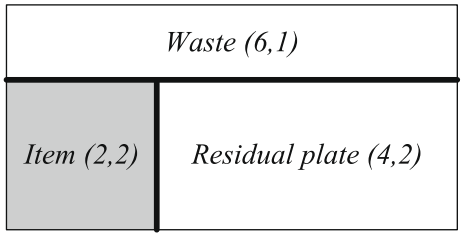

Fig. 4. The cut of one item of type two from a residual plate.

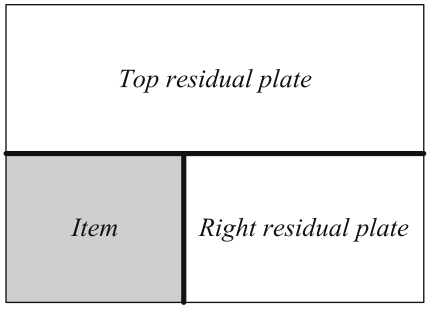

Fig. 5. A cut in the first or third stage.

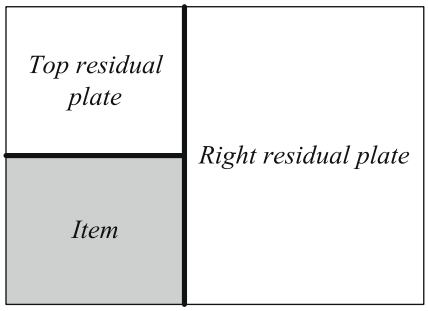

Fig. 6. A cut in the second stage.

The main idea of the proposed integer programming model is to associate one decision variable with each cut, or, more precisely, with the number of times that cut is performed. The constraints assure that the demand of the item types is fulfilled and that the cuts can be made only in existing plates.

The model is now formally presented. The number of item types is defined as n and the number of types of residual plates is defined as m. The plate types are indexed by j, $j = 0,\ldots,m$, with $j = 0$ corresponding to the initial plates and $j = 1,\ldots,m$ corresponding to types of residual plates. The demand of item type i, $i = 1,\ldots,n$, is denoted as d_i.

A decision variable x_{ij}, $i = 1,\ldots,n$, $j = 0,\ldots,m$, is defined as the number of times the item type i is cut from the plate type j. Parameter a_{ijk}, $i = 1,\ldots,n$, $j = 0,\ldots,m$, $k = 1,\ldots,m$, is equal to 1 if plate type k results from cutting item type i from the plate type j, and equal to 0 otherwise.

Using the notation just introduced, the model is the following:

$$\text{Minimize} \quad \sum_{i=1}^{n} x_{i0} \tag{1}$$

$$\text{Subject to} \quad \sum_{j=0}^{m} x_{ij} \geqslant d_i, \quad i = 1,\ldots,n, \tag{2}$$

$$\sum_{j=0}^{m}\sum_{i=1}^{n} a_{ijk}x_{ij} \geqslant \sum_{i=1}^{n} x_{ik}, \quad k = 1,\ldots,m, \tag{3}$$

$$x_{ij} \geqslant 0 \text{ and integer}, \quad i = 1,\ldots,n,\ j = 0,\ldots,m. \tag{4}$$

The objective function (1) minimizes the number of cuts in plates of the initial type. The set of constraints (2) ensure that the demand of each item type is satisfied. In the displayed model, overproduction (exceeding the demands) is allowed. If the greater than or equal to constraints (2) are replaced by equalities, the model does not allow overproduction, i.e., the exact quantities of each item type are cut. Constraints (3) state that, for each type of residual plates, the number of cuts resulting in such a residual plate must be greater than or equal to the number of cuts executed on such residual plates, i.e., only cuts in existing residual plates can be executed.

The size of the proposed model depends on the dimensions of the items and plate and on the number of item types. In general, instances with small items (when compared with the dimensions of the plate) are more difficult to solve (independently of the approach) than instances with larger items because of the combinatorial structure of the problem (with small items, more arrangements in one plate are feasible). In the proposed approach, having small items translates in a large number of cuts and residual plate types, which leads to a large integer programming model. The size of the model depends also on the number of item types. When comparing instances with the same number of items, the proposed model is more suitable for instances with fewer item types (with larger demands) than the opposite, since the demands do not affect the size of the model. Upper bounds on the size of the integer programming model are given in the following subsection.

2.2. Size

In this subsection, three upper bounds on the number of types of residual plates to the two-stage problems and three upper bounds on the number of types of residual plates to the three-stage problems are derived. These upper bounds can easily be used to obtain upper bounds on the number of cuts. Since in the integer programming model the number of constraints and variables are related with the number of types of residual plates and with the number of cuts, respectively, the upper bounds show that the size of the model is pseudo-polynomial.

The following notation is used. As before, n is the number of item types, W and H are the width and height of an initial plate, respectively, and $w'$and h' are the minimum width and the minimum height of an item type, respectively.

An upper bound for the number of types of residual plates for a two-stage problem based on the combinatorial structure of the problem is given in Proposition 1.

Proposition 1. *For the two-stage non-exact problem, an upper bound to the number of types of residual plates is given by*

$$UB2^c = n + n^2 + \cdots + n^{\lfloor \frac{H}{h'} \rfloor} + n + n^2 + \cdots + n^{\lfloor \frac{W}{w'} \rfloor}$$

Proof. There are two types of residual plates: the ones to the right of the item being cut and the ones on the top of it (see Figs. 5 and 6). We will denote them as the *top residual plate* and the *right residual plate*, respectively. Firstly, an upper bound to the number of top residual plates is obtained.

Since a two-stage problem is being considered, top residual plates only result from cuts in the initial plate (Fig. 5) or in other top residual plates. There are n cuts associated with the initial plate each producing one top residual plate. For each top residual plate, an upper bound to the number of cuts is n, each producing n top residual plates. An upper bound to the number of times this process can be repeated is $\left\lfloor \frac{H}{h'} \right\rfloor$. So, an upper bound to the number of top residual plates is $n + n^2 + \cdots + n^{\lfloor \frac{H}{h'} \rfloor}$.

Using the same reasoning, an upper bound to the number of right residual plates is now obtained. Note that, in the worst-case, there are n different heights for right residual plates (one for each item type). Starting from any plate (initial or residual), the number of cuts is n and each produces one right residual plate with the same height. An upper bound to the number of right residual plates is $n + n^2 + \cdots + n^{\lfloor \frac{W}{w'} \rfloor}$. □

Another upper bound for the number of types of residual plates, based on discretization, is given in Proposition 2.

Proposition 2. *For the two-stage non-exact problem, assuming that H, W, h', and w' are nonnegative integers, an upper bound to the number of types of residual plates is given by*

$$UB2^d = (H - h') + (W - w')n.$$

Proof. The upper bound to the number of top residual plates $H - h'$ is straightforward by discretization. The upper bound on the number of right residual plates is also straightforward given that for each of the n possible heights, by discretization, there are $(W - w')$ possible widths. □

Since no upper bound for the number of types of residual plates dominates the other, the best upper bound is given in Proposition 3.

Proposition 3. *For the two-stage non-exact problem, an upper bound to the number of types of residual plates is given by*

$$UB2^b = Min\{UB2^c, UB2^d\}.$$

The next three propositions are for three-stage problems. The maximum height of an item type is denoted by h^*.

Proposition 4. *For the three-stage non-exact problem, an upper bound to the number of types of residual plates is given by*

$$UB3^c = n + n^2 + \cdots + n^{\lfloor \frac{H}{h'} \rfloor} + n + n^2 + \cdots + n^{\lfloor \frac{W}{w'} \rfloor} + n + n^2 + \cdots n^{\lfloor \frac{h^* - h'}{h'} \rfloor}.$$

Proof. All the residual plates considered for the two-stage problem exist for the three-stage non-exact problem. For the three-stage problem, additional top residual plates related to cuts in right residual plates and top residual plates resulting from them must be considered. The number of those top residual plates is given by $n + n^2 + \cdots + n^{\lfloor \frac{h^* - h'}{h'} \rfloor}$. □

Proposition 5. *For the three-stage non-exact problem, assuming that H, W, h', and h^* are nonnegative integers an upper bound to the number of types of residual plates is given by*

$$UB3^d = (H - h') + n(W - W')(h^* - h').$$

Proof. The upper bound to the number of top residual plates in the first stage is straightforward by discretization. For each of the n possible heights, by discretization, there are possible widths of right residual plates (second stage). For each of these widths, in the worst-case there are $h^* - h'$ different heights. □

Proposition 6. *For the three-stage non-exact problem, an upper bound to the number of types of residual plates is given by*

$$UB3^b = Min\{UB3^c, UB3^d\}.$$

Since the number of constraints of the model is n plus the number of types of residual plates, an upper bound for the number of constraints of the model for two-stage problems is $n + UB2^b$ and, for three-stage problems, is $n + UB3^b$.

```
Begin
    Initialize R with the empty set
    Initialize C with the n cuts corresponding to cut each item type from an initial plate
    Mark all cuts in C as non-analysed
    While not all cuts from C have been analysed
        Let c be a non-analysed cut from C associated with item type i and a plate type
        Mark c as analysed
        Add the two plate types (top and right) defined by the width, height and stage given
                by the rules of Table 2 (two-stage problem) or Table 3 (three-stage
                problem) to R if they are not already there
        Forall item types i
            If item type i can be cut from the top plate type
                Add the cut associated with the top plate type and item type i to C and mark
                    it as non-analysed
            Endif
            If item type i can be cut from the right plate type
                Add the cut associated with the right plate type and item type i to C and mark
                    it as non-analysed
            Endif
        Endforall
    Endwhile
End
```

Fig. 7. Algorithm for generating cuts and plate types.

Table 2
Plate types resulting from cuts in different stages for a two-stage problem.

Cuts					Resulting plate types					
Plate type			Item type		Top			Right		
Width	Height	Stage	Width	Height	Width	Height	Stage	Width	Height	Stage
W_r	H_r	1	w_i	h_i	W_r	$H_r - h_i$	1	$W_r - w_i$	h_i	2
W_r	H_r	2	w_i	h_i	w_i	$H_r - h_i$	Waste	$W_r - w_i$	H_r	2

Table 3
Plate types resulting from cuts in different stages for a three-stage problem.

Cuts					Resulting plate types					
Plate type			Item type		Top			Right		
Width	Height	Stage	Width	Height	Width	Height	Stage	Width	Height	Stage
W_r	H_r	1	w_i	h_i	W_r	$H_r - h_i$	1	$W_r - w_i$	h_i	2
W_r	H_r	2	w_i	h_i	w_i	$H_r - h_i$	3	$W_r - w_i$	H_r	2
W_r	H_r	3	w_i	h_i	W_r	$H_r - h_i$	3	$W_r - w_i$	h_i	Waste

Since, in the worst-case, each plate type can be used to cut any item type, an upper bound to the number of decision variables for two-stage problems is $n(1 + UB2^b)$, and, for three-stage problems, is $n(1 + UB3^b)$.

2.3. Generating the cuts and the residual plates

The integer programming model is based on the enumeration of all possible cuts and residual plates. The algorithm for generating all the cuts and residual plates is given in Fig. 7. The list of the types of residual plates is denoted as R, and the list of the cuts is denoted as C. In both lists, repetition is not allowed. A set of rules to define the width, height and stage of the residual plates resulting from a cut are given in Tables 2 (two-stage problems) and 3 (three-stage problems).

When deciding if a cut should be considered (see the *if* conditions inside of the *forall* loop in the algorithm of Fig. 7), if an exact two-stage problem is addressed, only cuts with items having the same height as the one of the residual plate should be considered

	Item type i	1	2	1	2	2	1	2	2	2	2
	Plate type j	0	0	1	1	2	3	3	4	5	6
Item type	1	1		1			1				
	2		1		1	1		1	1	1	1
Plate type	1	1		-1	-1						
	2	1		1		-1	1				
	3		1				-1	-1			
	4		1		1			1	-1	1	
	5							1		-1	
	6								1		-1

Fig. 8. Constraint matrix for the instance of the example.

Table 4
Dimensions and stages of the plate types.

Plate type	Width	Height	Stage
0	6	6	1
1	6	3	1
2	2	3	2
3	6	4	1
4	4	2	2
5	6	2	1
6	2	2	2

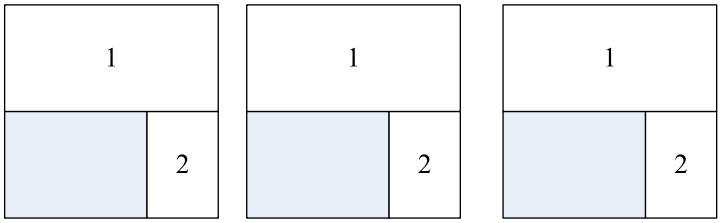

Fig. 9. Cuts on initial plates.

Fig. 10. Cuts on residual plates of type one.

for residual plates of stage two. If an exact three-stage problem is addressed, only cuts with items having the same width as the one of the residual plate should be considered for residual plates of stage three.

The algorithm for enumerating the cuts and the types of residual plates is used to build the integer programming model. As each column of the model corresponds to a cut, a column-wise representation of the non-zeros elements of the constraint matrix is used. Each column has, at most, four non zero elements: a one in the row of the demand of the item type associated with the cut, a minus one in the row of the residual plate associated with the cut, and two ones in the rows of the resulting residual plates. When a cut involves a type of residual plate, whose row was not previously generated, the row is created and added to the constraint matrix, using a dense format. Each cut is considered only once. For each cut, all the types of residual plates previously generated are scanned in order to avoid repetitions.

The *while* cycle depends on the number of cuts and in each iteration it is necessary to check if the generated residual plates are already in the list R. Using basic data structures to implement the algorithm, the worst-case computational complexity of the algorithm is then $0(|C||R|)$. Using the upper bounds on the number of types of residual plates and cuts derived in the previous subsection, the worst-case computational complexity of the algorithm is $0\left(n^{1+max\{\lfloor\frac{H}{h'}\rfloor,\lfloor\frac{W}{w'}\rfloor\}}\right)$ for both two-stage and three-stage problems. In practice, as it is shown by the computational results of Section 4, the time of the construction of the model is always much smaller than the time spent in the optimization of the model.

A procedure for reducing the symmetry of the model (several equivalent solutions corresponding to different values of the decision variables), which is frequently used in models for cutting stock problems, is now given for the non-exact two-stage and three-stage problems. For the non-exact two-stage problem, this procedure can be seen as ordering the items in the stripes by decreasing height. When a cut is executed on a residual plate of stage two, the resulting right residual plate will have the height of the item and not the one of the residual plate. Therefore, all the cuts based on the right residual plate (which is also a residual plate of stage two) will be associated with items with height less than or equal to the height of the item.

A similar reasoning can be applied to the non-exact three-stage problem. In this case, the items in the stacks can be sorted by decreasing width. When a cut is executed on a residual plate of stage three, the resulting top residual plate will have the width of the item and not that of the residual plate. These procedures only affect the order of the items in a stripe (two-stage problem) or stack (three-stage problem) and not the value of a solution.

2.4. Example

The constraint matrix of the integer programming model for the small instance introduced in the beginning of Section 2.1 is provided in Fig. 8. The width, height and stage of the types of residual plates are provided in Table 4. Each column of the matrix of Fig. 8 is associated with a decision variable and has (i) a 1 in the first row of the matrix if the corresponding cut is for item type one or a 1 in the second row if the corresponding cut is for item type two; (ii) a -1 in the row associated with the residual plate type where the cut is performed (if the cut is performed in an initial plate the column does not have a -1); (iii) a value 1 in each row associated with a resulting residual plate type (at most two rows).

For example, the third column (associated with x_{11}) corresponds to the cut associated with item type one and plate type one (width 6, height 3, first stage) and resulting in a plate of type two (width 2, height 3, stage two).

An optimal solution is $x_{10} = 3,\ x_{11} = 2,\ x_{21} = 1,\ x_{22} = 2,\ x_{24} = 1,\ x_{26} = 1$, with all the other decision variables with *0* value. Recall that each x_{ij} is the number of times the item type i is cut from plate type j.

The sequence of cuts starts from the initial plates and proceeds by analysing the cuts on the resulting residual plates. The interpretation of this solution should be made as follows. Since $x_{10} = 3$ and $x_{20} = 0$, three initial plates are used and in each of them, one item of type one is cut. As shown in the constraint matrix and in Fig. 9 (where the numbers are the indexes of the types of residual plates), those cuts lead to three residual plates of type one and three residual plates of type two.

Since $x_{11} = 2$ and $x_{21} = 1$, two items of type 1 are cut from two of the residual plates of type one and one item of type two is cut from the third residual plate of type one. See Fig. 10 for an illustration.

Since $x_{22} = 2$, two of the five residual plates of type two are cut in two items of type two, as illustrated in Fig. 11.

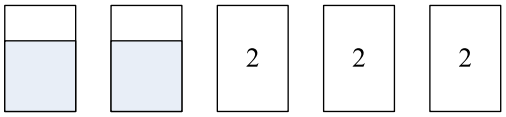

Fig. 11. Cuts on residual plates of type two.

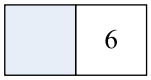

Fig. 12. Cuts on the residual plate of type four.

Since $x_{24} = 1$, one item of type two is cut from the residual plate of type four, as illustrated in Fig. 12.

At last, since $x_{26} = 1$, the residual plate of type six is used as an item. Three residual plates of type two were not further cut.

A solution consisting in cuts can be also represented by patterns and *vice-versa*. For the example, a possible representation in patterns of the solution previously described is given in Fig. 13.

3. Extensions

3.1. Rotation

In some 2DCS problems, rotating the items by 90 degrees is allowed. It is easy to deal with this extension in the model; a cut is now defined by one plate type, one item type, and the orientation of the item. The decision variables become x_{irj}, meaning the number of times that the item i with orientation r ($r = 0$ means that the item is not rotated and $r = 1$ means the item is rotated) is cut from plate type j. The generation of the cuts and residual plates only require minor modifications, taking into account that each item can be cut in two different ways from a plate.

The model is the following:

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{r=0}^{1} x_{ir0}$$

$$\text{subject to} \quad \sum_{j=0}^{m}\sum_{r=0}^{1} x_{irj} \geqslant d_i, \quad i = 1,\ldots,n,$$

$$\sum_{j=0}^{m}\sum_{i=1}^{n}\sum_{r=0}^{1} a_{irjk}x_{irj} \geqslant \sum_{i=1}^{n}\sum_{r=0}^{1} x_{irk}, \quad k = 1,\ldots,m,$$

$$x_{irj} \geqslant 0 \text{ and integer}, \quad i = 1,\ldots,n;\ r = 0,1;\ j = 0,\ldots,m.$$

3.2. Multiple types of plates

The extension to the 2DCS problem with multiple plates' sizes is straightforward. The only significant difference is the initialization of the cuts' list when generating the integer programming model.

3.3. Length of the cuts

The usual objectives considered in cutting stock problems are the minimization of the number of plates used and the minimization of the waste (which will be discussed in the next subsection). Usually, in practical settings, issues related to the length of the cuts are also relevant. For example, the total length of the cuts is related to the time spent in executing the cutting operations, which may be a relevant issue for production management.

An example of two solutions with the same number of plates but different total length of cuts is given in Fig. 14. Assuming that the initial plate has height 4 and width 2, it can be easily seen that a cut length of 8 is needed for the first solution, while the second only needs a cut length of 7.

The model can incorporate this issue, since it is easy to calculate the length of the cutting operations associated with a cut. For a cut in a residual plate of stages one or three (see Fig. 5) the length of the cuts is $W_j + h_i$, where W_j is the width of the residual plate, if the heights and widths of the residual plate and of the item type are different. If the height is the same, then the length of the cut is h_i. If the width is the same, then the length of the cut is W_j. Similar conditions and calculations can be made for cuts in residual plates of stage two (see Fig. 6).

Representing the length of the cuts of cut ij (item i cut from plate type j) by l_{ij}, the total length of the cuts is given by

$$\sum_{i=1}^{n}\sum_{j=0}^{m} l_{ij}x_{ij}.$$

This (linear) function can be used in a constraint to state that the total length of the cuts must be less than or equal to some value or as the objective function. In the latter case, the number of used plates may be included in the model as a constraint.

Another possible use of the above function is to select among alternative (optimal) solutions with the same number of plates. In this case, the function may appear as an additional term with a small weight in the objective function.

3.4. Minimizing waste

In the model, as it was presented so far, at the end of the cut process, there are three types of pieces: items, waste plates (the ones in which it is not possible to cut any item) and empty residual plates (residual plates in which no item is cut). See Fig. 13 for an illustration of these three types of pieces.

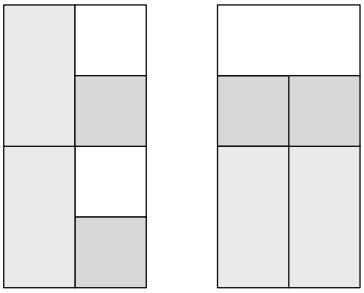

Fig. 14. Two solutions with one plate but with different numbers and lengths of cuts (white spaces are waste).

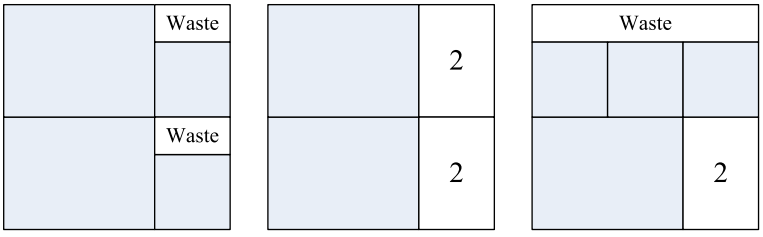

Fig. 13. The solution represented in terms of patterns.

When minimizing the number of used plates, only decision variables related with the residual plates are needed. However, when minimizing the waste, it is necessary to quantify the area of waste plates and the area of empty residual plates. Therefore, for waste minimization, two new sets of decision variables are defined: w_p is the number of waste plates of type $p, w_p \geqslant 0$ and integer, $p = 1,\ldots,q$, where q is the number of types of waste plates; and $s_k, s_k \geqslant 0$, is the number of empty residual plates of type k, $k = 1,\ldots,m$. The latter variables are the surplus variables of the residual plates' constraints, which are now equality constraints:

$$s_k = \sum_{j=0}^{m}\sum_{i=1}^{n} a_{ijk}x_{ij} - \sum_{i=1}^{n} x_{ik}, \quad k = 1,\ldots,m.$$

Defining c_p as the area of a waste plate of type p, $p = 1,\ldots,q$, and c_k as the area of residual plate of type k, $k = 1,\ldots,m$, respectively, the objective function for waste minimization is

$$\text{Min} \quad \sum_{p=1}^{q} a_p w_p + \sum_{k=1}^{m} c_k s_k.$$

3.5. Remaining residual plates

In the model, it is straightforward to treat the pieces that are not cut as stock for the future (next planning period) with a given value.

Using the surplus variables introduced in the previous subsection, a value can be attributed to residual plates and then considered in the objective function. Residual plates not likely to be used in the future, in particular because their area (or width or height) is below some threshold, should have value zero. Values proportional to the area (for example) can be given to the other residual plates. In general, any value can be associated with any residual plate.

4. Computational results

4.1. Computational environment and overview of the tested instances

The construction and optimization of the proposed model was implemented in C++ using the C callable library of the ILOG Cplex 11.0 solver (Ilog, 2007). The same callable library was used to optimize the integer programming models of Lodi et al. (2004) for the two-stage non-exact problem and of Puchinger and Raidl (2007) for the three-stage exact problem. In all cases, no tuning of the solver was attempted and all the default parameters were used. All computational tests were performed on a 1.87 GHz Intel Core Duo processor with 2 GB of RAM memory.

Information on the four sets of instances tested can be found in Tables 1–4 in the e-appendix. Two sets are "real-world" instances provided by two different furniture companies, one with 47 instances and the other with 121 instances. Instances of the first company (A) are heterogeneous, some of them have small demands (few units) and some of them have large demands (dozens). Instances of the second company (B) have large or very large demands (dozens or hundreds, or even thousands).

The other two sets of instances were adapted from instances previously tested on the constrained two-dimensional cutting stock problem (Hifi and Roucairol, 2001; Lodi and Monaci, 2003; Belov and Scheithauer, 2006; Cui, 2008). In that problem, the objective is to maximize the profit of the set of items cut from a single plate. Each item type has a profit and an upper bound on the number of items that can be cut from the plate. In order to adapt those instances to the cutting stock problem addressed in this paper, the profit was neglected and the upper bounds were taken as the demands of the item types. The first set (C) has 30 instances with average demands between 1 and 3 and the second one (D) has 20 instances with average demands between 4 and 6.

From Tables 5–12 of the e-appendix, the optimal value, i.e., the number of used plates (column z), the total (construction of the model plus optimization) CPU time in seconds (column t), the CPU time in seconds required only for building the model (column c), the lower bound given by the linear relaxation (column lb), and the number of nodes of the branch-and-bound tree (column nbb) are displayed for the four problems addressed: exact two-stage (2E), non-exact two-stage (2NE), exact three-stage (3E), and non-exact three-stage (3NE), for each set of instances, for the proposed model (cut model) and the models of Lodi et al. (2004) (proposed for the 2NE problem) and of Puchinger and Raidl (2007) (proposed for the 3E problem). A time limit of 7200 seconds was set for the Cplex solver in all cases. For all sets of instances, the time of the construction of the models is always much smaller than the time spent in their optimization.

When the optimality of the solution was not proved, the value in column z is the value of the incumbent solution, z_{INC}, and the absolute gap, $Z_{INC} - \lceil Z_{LB} \rceil$, where z_{LB} is a lower bound to the optimal value, is given in parenthesis. The lower bound is given directly by the Cplex branch-and-bound solver. In a branch-and-bound algorithm, each unexplored node has a lower bound corresponding to the optimal value of the linear relaxation of its father. A global lower bound to the value of an optimal integer solution is the minimum lower bound value of all unexplored nodes.

4.2. "Real-world" instances

For company A (Tables 5 and 6 of the e-appendix), in a rough analysis, the optimal solution to instances with less than 20 item types is obtained by the proposed model in few seconds, the optimal solution to instances with 20–40 item types is obtained by the proposed model in tens of seconds, the optimal solution to instances from 40 to 100 item types are solved in hundreds or few thousands of seconds (for 3E and 3NE some instances already reached the time limit with incumbent solutions), and *proved* optimal solutions for instances with more than 100 item types are only within reach for the easiest problems (2E and 2NE), although the solutions obtained are optimal or very close to the optimal. Only for the instance with more item types and the most complex problem (instance FA+AA-9_13, 3NE problem), a feasible solution was not obtained by the proposed model.

For this set of instances, the proposed models clearly outperform the ones from the literature. In several instances, the models from the literature reach the time limit without proving optimality or their optimization is aborted because not enough memory is available. The lower bounds given by the linear relaxation of the proposed model are much better than the ones given by the linear relaxation of the models from the literature (improvement of 12% for the 2NE problem and improvement of 13% for the 3E problem). The higher quality of the lower bounds contributes to the much smaller number of branch-and-bound nodes required to obtain an optimal solution by the proposed models.

Only in two occasions (FA+AA-9_13, 2NE and FA+AA-9_6, 3E) with instances from the subset FA+AA which have small demands and a large number of item types, the models from literature obtained better solutions than the proposed models.

For company B (Tables 7 and 8 of the e-appendix), around 90% of the instances are solved by the proposed model in less than 10 seconds (93%, 90%, 86%, and 86% for 2E, 2NE, 3E and 3NE, respectively). An incumbent solution was not found for the same instance (REVAL_140) for three problems (2NE, 3E, and 3NE). This instance is the one, from this set, with the largest number of item types. With the proposed model, a large number of instances from this set is solved in the root node of the branch-and-bound tree,

confirming the good quality of the lower bounds provided by its linear relaxation.

Since for these instances the demands are very large, the models from the literature, although being compact models, are very large and cannot even be constructed for several instances. In a rough analysis, the models from Lodi et al. (2004) and Puchinger and Raidl (2007) are able to obtain optimal solution in instances with up to 200 items and feasible solutions in instances with up to 1500 items. For instances with more items, the models become too large to be computationally tractable.

In general, summing up the results of the instances A and B, the proposed model is able to solve almost instantly "real-world" instances with less than 20 item types and "real-world" instances with up to 40 item types in few minutes. Few instances provided by the two furniture companies have more than 40 item types. For all of those instances, except two, optimal solutions or solutions with a gap of one or two plates (meaning that the solution obtained is either optimal or the optimal solution uses only one or two plates less) were obtained in reasonable amounts of time.

For the non-exact two-stage problem, the proposed model was not able to reach an optimal solution in 5 out of 168 instances, while the model proposed by Lodi et al. (2004) was not able to obtain an optimal solution in 121 out of the same 168 instances. For the exact three-stage problem, the proposed model did not obtain an optimal solution in 6 out of 168 instances, while the model proposed by Puchinger and Raidl (2007) was not able to obtain an optimal solution in 120 out of 168 instances.

4.3. Instances adapted from the literature

For the set of instances C (Tables 9 and 10 of the e-appendix), the proposed models obtained the optimal solution in all except one case (instance CHL6, problem 3E). The same instance for the same problem was not solved also by the model of Puchinger and Raidl (2007).

There are two other cases for which the models from the literature could not obtain an optimal solution (Hchl7s, 2NE and CW1, 3E) due to insufficient memory. It is not obvious what makes these instances more difficult than other instances of the same set. Possibly, the quality of the lower and upper bounds (used in the branch-and-bound) for these instances is worst than for other instances due to geometric aspects.

For the easiest problem (2E) the proposed model solved the set of instances C almost instantly, the maximum computational time was 2.9 seconds for the CHL7 instance.

As it was expected the computational times increased for the non-exact three-stage problem when compared with the other types of problems (2E, 2NE, 3E), but still the proposed model was able to solve all the instances optimally.

The D set of instances (Tables 11 and 12 of the e-appendix) is the most difficult one for all the methods because each plate can accommodate a large number of items (for the 2NE problem, the solutions obtained by the proposed model have around 20 items per plate) and the instances have a considerable number of item types (between 25 and 58 with an average of 40) and a considerable number of items (between 119 and 325 with an average of 202). The number of item types and the number of items are crucial factors for the size of the proposed model and of the models from the literature, respectively.

For the exact two-stage problem, the proposed model provided an optimal solution for all instances and 65% of the instances were solved in less than 30 seconds.

For the non-exact two-stage problem, the results of the proposed model are better than the ones of the model described in Lodi et al. (2004) (a better solution was obtained in 9 out of 20 instances, a solution with the same value but a smaller gap was obtained in three instances and for the eight instances, the value and the gap of the solutions were the same). For the exact three-stage problem, the results of the proposed model are slightly worse than the ones of the model described in Puchinger and Raidl (2007) (a better solution was obtained in 4 out of 20 instances, but a worse solution was obtained in six instances).The improvement of the lower bound of the proposed models in respect to the models from the literature is smaller (3% for 2NE and 2% for 3E) than the improvement in the other sets of instances.

For the non-exact three-stage problem, only 5 out of the 20 instances were solved optimally by the proposed model and a feasible solution was not found (in 7200 seconds) for two instances. For the remaining 13 instances, although optimality was not proved, the solutions obtained are indeed optimal or use only one or two more plates than the optimal solution.

5. Conclusions

In this paper, an integer programming model to solve two-stage and three-stage two-dimensional cutting stock problems exactly was proposed. The model is an extension of the "one-cut" model of Dyckhoff for the one-dimensional cutting stock problem. A cut corresponds to cutting an item from a plate, which can be an initial or a residual plate, and produces more residual plates with different dimensions according to the stage where it is performed.

Although pseudo-polynomial in size, the model optimized by a general-purpose integer programming solver (Cplex 11.0) solved a large number of "real-world" instances in a very reasonable amount of time. The computational results confirm that the proposed model is sensitive to the number of item types of the instance and to the relation between the dimensions of the plate and the dimensions of the items (instances where each plate accommodates fewer items are easier for all solution approaches) but is not sensitive to the number of items.

For the 672 combinations of "real-world" instances and problems tested (168 instances for each problem), only in 29 occasions, corresponding to instances with a large number of item types, an optimal solution could not be found in two hours. In 21 of those occasions, the optimal solution is either the incumbent solution or uses only one plate less than the incumbent. For "real-world" instances, the behaviour of models from the literature (which can also be directly solved by a general-purpose MIP solver) was clearly inferior to the one proposed, since those models are sensitive to the number of items of the instance and some instances have thousands of items. For instances adapted from the literature the behaviour of the two models was similar.

The flexibility of the proposed model allows considering extensions to take into account issues that may have a relevant role in practical problems. In particular, the existence of multiple plate sizes, the incorporation of the values of the remaining plates, and the incorporation of the length of the cuts and the rotation of items were addressed.

Acknowledgments

We would like to thank the anonymous referees for their constructive comments, which led to a clearer presentation of the material.

This work was supported by project SCOOP – Sheet cutting and process optimization for furniture enterprises (Contract No. COOP-CT-2006-032998), financed by the European Commission, 6th Framework Programme on Research, Technological Development and Dissemination, specific actions for SMEs, Cooperative Research Projects. The first author was also supported by Grant SFRH/BD/42259/2007 from Fundação para a Ciência e Tecnologia, Portugal.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2010.01.039.

References

Belov, G., 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, 85–106.

Cintra, G.F., Miyazawa, F.K., Wakabayashi, Y., 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 191, 59–83.

Cui, Y., 2008. Heuristic and exact algorithms for generating homogenous constrained three-staged cutting patterns. Computers and Operations Research 35, 212–225.

Dyckhoff, H., 1981. A new linear programming approach to the cutting stock problem. Operations Research 29 (6), 1092–1104.

Dyckhoff, H., Scheithauer, G., Terno, J., 1997. Cutting and packing. In: Dell'Amico, M., Maffioli, F., Martello, S. (Eds.), Annotated Bibliographies in Combinatorial Optimization. John Wiley and Sons.

Gilmore, P.C., Gomory, R.E., 1965. Multistage cutting stock problems of two and more dimensions. INFORMS, Operations Research 13 (1), 94–120.

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

Ilog, 2007.CPLEX 11.0, User's Manual.

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

Lodi, A., Martello, S., Monaci, M., 2002. Two dimensional packing problems: A survey. European Journal of Operational Research 141, 241–252.

Lodi, A., Martello, S., Vigo, D., 2004. Models and bounds for two-dimensional level packing problems. Journal of Combinatorial Optimization 8, 363–379.

Pisinger, D., Sigurd, M., 2007. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. INFORMS Journal on Computing 19 (1), 36–51.

Puchinger, J., Raidl, G.R., 2007. Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research 183, 1304–1327.

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

Wäscher, G., Haussner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 183 (3), 1109–1130.