ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Logic based Benders' decomposition for orthogonal stock cutting problems

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Delorme, M., Iori, M., Martello, S. (2017). Logic based Benders' decomposition for orthogonal stock cutting problems. COMPUTERS & OPERATIONS RESEARCH, 78, 290-298 [10.1016/j.cor.2016.09.009].

Availability:
This version is available at: https://hdl.handle.net/11585/588348 since: 2020-04-20

Published:

DOI: http://doi.org/10.1016/j.cor.2016.09.009

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

12 October 2026

This is the final peer-reviewed accepted manuscript of:

Maxence Delorme, Manuel Iori, Silvano Martello,

Logic based Benders' decomposition for orthogonal stock cutting problems,

Computers & Operations Research, Volume 78, 2017, Pages 290-298, ISSN 0305-0548.

The final published version is available online at

https://doi.org/10.1016/j.cor.2016.09.009

© 2016 This manuscript version is made available under the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/)

*Manuscript
Click here to view linked References

Logic Based Benders' Decomposition for Orthogonal Stock Cutting Problems

Maxence Delorme[a,*], Manuel Iori[b], Silvano Martello[a]

[a] *DEI "Guglielmo Marconi", University of Bologna*
[b] *DISMI, University of Modena and Reggio Emilia*

Abstract

We consider the problem of packing a set of rectangular items into a strip of fixed width, without overlapping, using minimum height. Items must be packed with their edges parallel to those of the strip, but rotation by 90° is allowed. The problem is usually solved through branch-and-bound algorithms. We propose an alternative method, based on Benders' decomposition. The master problem is solved through a new ILP model based on the arc flow formulation, while constraint programming is used to solve the slave problem. The resulting method is hybridized with a state-of-the-art branch-and-bound algorithm. Computational experiments on classical benchmarks from the literature show the effectiveness of the proposed approach. We additionally show that the algorithm can be successfully used to solve relevant related problems, like rectangle packing and pallet loading.

Keywords: Orthogonal stock cutting problem, Logic based Benders' decomposition, Rectangle packing, Pallet loading

1. Introduction

Given n rectangular *items* of integer width w_j and height h_j $(j = 1, \ldots, n)$ and a *strip* of fixed width W, the *orthogonal Stock Cutting Problem* (SCP) consists in packing all the items into the strip without overlapping and using the minimum strip height. Items must be packed with their edges parallel to those of the strip, but rotation by 90° is allowed. The *oriented* counterpart of the SCP, in which rotation is not allowed, is known in the literature as the *Strip Packing Problem* (SPP, see, e.g., Martello, Monaci, and Vigo [1]). Both the SCP and the SPP are important because they have many real world applications, especially in wood, paper, glass, and metal industries (see, e.g., Lodi, Martello, and Monaci [2]).

It is not difficult to see that the SCP is $\mathcal{NP}$-hard in the strong sense. Consider indeed the famous (one-dimensional) *bin packing problem* (BPP): partition n elements, having values v_j $(j = 1, \ldots, n)$ into the minimum number of subsets so that the sum of the values in each subset does not exceed a given capacity V (see Delorme, Iori, and Martello [3] for a recent exhaustive survey). Any BPP instance can be transformed into an equivalent SCP instance by setting $W = V$ and, for $j = 1, \ldots, n$, $w_j = v_j$ and $h_j = W + 1$. Its solution will consist of the minimum number of "shelves" of height $W + 1$, and hence it will provide the optimal

*Corresponding author `maxence.delorme2@unibo.it`, phone/fax +39 051 2093646/3073

solution to the BPP instance. The BPP is known to be $\mathcal{NP}$-hard in the strong sense, so the same holds for the SCP.

According to the classification introduced by Lodi, Martello, and Vigo [4], the SCP and the SPP can be characterized as 2SP|R|F and 2SP|O|F, respectively, while in the general cutting and packing typology by Wäscher, Haußner, and Schumann [5] both problems are categorized as *two-dimensional open dimension problem*.

In this paper we present an exact algorithm for the SCP. In Section 2 we review successful approaches that have been proposed in the literature for the SCP, the SPP, and related problems. In Section 3 we provide a mathematical model for the SCP. Preprocessing techniques and initial lower and upper bound computations are discussed in Section 4. The proposed algorithm is given in Sections 5, 6, 7, and 8. The outcome of extensive computational experiments is presented in Section 9, where we also discuss some relevant variants of the problem.

2. Literature review

Most of the techniques developed in the literature to solve the SCP and the SPP are combinatorial branch-and-bound algorithms. One of the first branch-and-bound approaches for the SPP was proposed by Martello, Monaci, and Vigo [1] in the early noughties. The algorithm makes use of preprocessing techniques, dominance criteria, and a powerful lower bound based on a problem oriented continuous relaxation. In the same period, Lesh, Marks, McMahon, and Mitzenmacher [6] proposed a branch-and-bound algorithm specifically tailored for *perfect-packing* cases, in which the optimal solution has no loss space, i.e., $\sum_{j=1}^{n} w_j h_j = WH$, where H denotes the height of the optimal solution. The algorithm is based on powerful cuts that fathom nodes for which the current partial packing cannot be filled by the remaining items without inserting a 'hole'. Later on, Alvarez-Valdes, Parreño, and Tamarit [7], as well as Boschetti and Montaletti [8] obtained better branch-and-bound algorithms by improving the dominance criteria and the bounding techniques proposed in [1].

In recent years, new types of exact methods were proposed by Westerlund, Papageorgiou, and Westerlund [9] and by Castro and Oliveira [10], who tried *Mixed Integer Linear Programming* (MILP) models to solve more general classes of problems, that include the SPP. The latter also made computational experiments on SPP instances, but the results were not encouraging, as the proposed approaches generally perform worse than the 'old' Martello, Monaci, and Vigo [1] branch-and-bound algorithm. Côté, Dell'Amico, and Iori [11] found better results by using a Benders' decomposition in which the master problem is a MILP model and the slave is solved through branch-and-bound.

To the best of our knowledge, Jakobs [12] was the first to propose a metaheuristic (genetic) algorithm for the SPP. Since then, many metaheuristics have been proposed, e.g., by Iori, Martello, and Monaci [13] to cite a well-known approach. As the focus of this paper is on exact solutions, we do not give an exhaustive list of heuristic and metaheuristic algorithms: the interested reader can refer to the recent articles by Özcan, Kai, and Drake [14] and by Thomas and Chaudhari [15] for updated reference lists.

Coming to the SCP, the literature particularly focused on heuristics and metaheuristics rather than on exact methods: see, e.g., Burke, Kendall, and Whitwel [16] for a classical approach, or Wei, Oon, and Lim [17] for an extensive literature review. To the best of our

knowledge, the only exact approaches for the SCP were proposed by Kenmochi, Imamichi, Nonobe, Yagiura, and Nagamochi [18] and by Arahori, Imamichi, and Nagamochi [19]. Both approaches are based on branch-and-bound techniques. The former algorithm transforms a general instance into a perfect-packing one by adding small items of size 1×1. It then applies a branch-and-bound method that fathoms nodes through powerful cuts derived from those proposed in [6] (see above). The latter algorithm improves the enumeration scheme by adding innovative cuts and strong fathoming criteria. According to the computational experiments presented in [19] this can be considered the state of the art of the exact algorithms for the SCP. Note that these two algorithms are also used for the SPP, and that both present very competitive results on this problem variant too.

In the last decades, several problems related to the SCP were also studied. We cite in particular the problems of *Packing Squares into a Square* (PSS) and of *Packing Rectangles into a Square with Rotation* (PRSR), which ask for the minimum area square required to pack a given set of items. As we will see, the approach we propose can be easily applied to solve them. In the early nineties, Leung, Tam, Wong, Young, and Chin [6] proved that the PSS is strongly $\mathcal{NP}$-hard, hence the same holds for PRSR. The two problems have been studied by Picouleau [20], who gave some simple heuristics for the PSS, by Caprara, Lodi, Martello, and Monaci [21], who proposed lower bounds and determined their worst-case performance, and by Correa [22], who presented a polynomial-time approximation scheme for the PRSR. To the best of our knowledge, the only exact approach for the PSS and the PRSR is the one proposed by Martello and Monaci [23].

Another related problem to which our approach can be extended is the *Pallet Loading Problem* (PLP), which consists in packing the maximum number of identical rectangles into a given larger rectangle. This problem, whose complexity status is currently unknown, was introduced in the late sixties by Barnett and Kynch [24] and then extensively studied by many researchers, who proposed both exact methods (see, e.g., Dowsland [25], Alvarez-Valdes, Parreño, and Tamarit [26], or Ahn, Park, and Yoon [27]) and powerful heuristics (see, e.g., Lins, Lins, and Morabito [28] or Birgin, Lobato, and Morabito [29]). The number of references being huge, we refer the interested reader to the recent, very complete survey by Silva, Oliveira, and Wäscher [30].

The exact approach we propose is based on the logic based Benders' decomposition, that was formally developed by Hooker [31], and applied with success by Hooker and Ottosson [32] to 0-1 programming and by Hooker [33] to planning and scheduling problems. The approach was later specialized to mixed integer programming by Codato and Fischetti [34] who introduced the so-called combinatorial Benders' cuts. Such cuts were successfully used by Côté, Dell'Amico, and Iori [11] for solving the SPP. In the *logic based Benders' decomposition*, the classical master problem is frequently solved as a MILP and the slave through constraint programming (although other solution techniques can be used). Constraint programming techniques were used by Clautiaux, Jouglet, Carlier, and Moukrim [35] to solve the recognition version of the SPP.

3. Mathematical model

We assume the existence of a coordinate system with origin in the bottom-left corner of the strip, having integer coordinates on the horizontal (width) and the vertical (height) axis.

For any integer coordinate i on the width axis, we call the unit-width vertical space above interval $[i, i+1)$ a *column*. We say that an item j *engages* a column q whenever j is packed so as to occupy a portion of column q. We will use the following notation

- $\mathcal{N} = \{1, 2, \ldots, n, n+1, \ldots, 2n\}$ = set of the given n items ($j = 1, 2, \ldots, n$) followed by a copy of each of them rotated by 90°, i.e., such that $w_j = h_{j-n}$ and $h_j = w_{j-n}$ ($j = n+1, n+2, \ldots, 2n$);
- $\mathcal{W} = \{0, 1, \ldots, W-1\}$ = set of all integer width coordinates (abscissae) where the bottom left corner of an item can be packed;
- $\mathcal{W}(j, q) = \{q - w_j + 1, \ldots, q\}$ = set of integer abscissae a such that, if item j is packed with its bottom-left corner at a (at any ordinate), then it engages column q.

From now on, when no confusion arises, we will use the term 'item' to denote either an input item or a rotated copy. By introducing decision variables

- x_{jp} = binary variable taking the value 1 if the bottom-left corner of item j is packed at abscissa p, and the value 0 otherwise ($j \in \mathcal{N}, p \in \mathcal{W}$);
- y_j = ordinate at which the bottom edge of item j is packed ($j \in \mathcal{N}$);
- z = overall height of the packing,

the SCP can be modeled as:

$$\min \quad z \tag{1}$$

$$\sum_{p \in \mathcal{W}} x_{jp} + x_{(n+j)p} = 1 \qquad j = 1, 2, \ldots, n, \tag{2}$$

$$y_j + h_j \leq z \qquad j \in \mathcal{N}, \tag{3}$$

$$\text{nonoverlap} \left\{ [y_j, y_j + h_j], j \in \mathcal{N} : \sum_{p \in \mathcal{W}(j,q)} x_{jp} = 1 \right\} \qquad q \in \mathcal{W}, \tag{4}$$

$$x_{jp} \in \{0, 1\} \qquad j \in \mathcal{N}, p \in \mathcal{W}, \tag{5}$$

$$y_j \geq 0 \qquad j \in \mathcal{N}, \text{ integer}. \tag{6}$$

Constraints (2) ensure that each item is packed exactly once, either rotated or not. Constraints (3) impose that no item exceeds the height of the strip. Logical constraints (4) (of "constraint programming" flavor) prevent items from overlapping by imposing, for each column q, that the vertical intervals $[y_j, y_j + h_j)$ associated with all items j that engage column q do not overlap.

Constraints (4) can equivalently be expressed through linear inequalities, as done by Baldacci and Boschetti [36] and by Castro and Oliveira [10]. However computational experiments (see, e.g., [10]) show that the direct use of the resulting mathematical model with an MILP solver is not very effective. In the next sections we show that decomposition techniques produce instead reduced models that can be solved more easily.

4. Preprocessing

We adapted a number of techniques from the literature to preprocess SCP instances through reduction, heuristic initialization, and lower bound computations.

A size reduction of strip and item width is attempted through an adaptation to the SCP of two techniques proposed by Boschetti and Montaletti [8] for the SPP:

(a) determine, through dynamic programming, the maximum width $W' \leq W$ that can be obtained by packing side by side any subset of the given items: if $W' < W$, then the strip width can be set to W';

(b) for each item $j \in \mathcal{N}$ compute, through dynamic programming, the maximum width w'_j that a subset of items can take when packed side by side with j: if $w_j + w'_j < W$, then the item width can be increased to $W - w'_j$. Note that, after this process, the width (resp. height) of an item $j > n$ is no longer necessarily equal to the height (resp. width) of item $j - n$ (see Section 3).

A third reduction technique, also presented in [8] but developed by Martello, Monaci, and Vigo [1] for the SPP, is based on a dominance criterion that packs at the bottom of the strip some large items and all the small items that would fit side by side with the large ones, provided such a packing is possible. However, its extension to the SCP appears to have little usefulness, due to difficulty in handling the possible rotation of the considered large items.

As mentioned in Section 2, several heuristics exist for the SCP. We adopted the *best-fit heuristic* by Burke, Kendall, and Whitwel [16], which is relatively simple and has a good average performance. We implemented the three versions proposed in [16], that differ from each other in the position where the next item is packed (*Leftmost*, *Next to Tallest Neighbor*, *Next to Shortest Neighbor*).

Concerning lower bounds, we computed a value LB as the best (higher) between

(i) the simple bound $L_1 = \max\left(\lceil \sum_{j=1}^{n} w_j h_j / W \rceil, \max_{j=1,\ldots,n}\{\min(w_j, h_j)\}\right)$, and

(ii) L_2 = rounded up solution value of the continuous relaxation of the *Integer Linear Programming* (ILP) model that will be introduced in Section 6.

Once LB has been computed, we execute a *truncated* (heuristic) version of the branch-and-bound algorithm proposed by Kenmochi, Imamichi, Nonobe, Yagiura, and Nagamochi [18], which looks for a feasible solution of value LB: we selected the so called *G-staircase placement* with *DP cuts*, as its performance on particular types of instance (where the unused space is very small) was shown to be very good. Such strategy maintains, at each decision node, a staircase-shape envelope that separates the two regions where the next items may or may not be placed. We limited the exploration of the branch-decision tree by: (i) heuristically killing decision nodes that are unlikely to lead to an optimal solution, and (ii) cutting out "small" portions of the strip region available to the next packings with the objective of maintaining a low number of steps in the staircase.

5. Decomposition algorithm

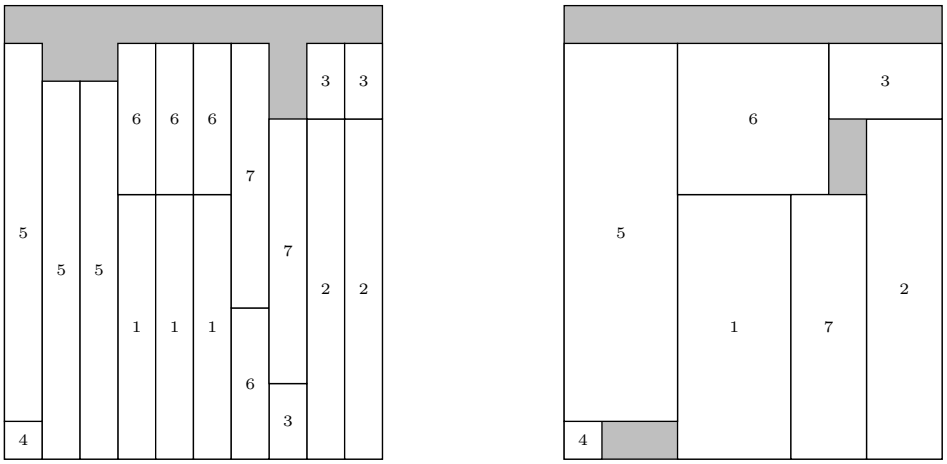
The decomposition introduced by Benders [37] solves a difficult problem through the iterative solution of two subproblems: the *master problem* and the *slave problem*. The master is generally a relaxation of the given problem, while the slave either provides an overall optimal solution or generates cuts to be added to the master at the next iteration. The algorithm we propose is derived from relatively recent developments of this technique: the logic based Benders' decomposition (Hooker [31]) and the combinatorial Benders' cuts (Codato and Fischetti [34]). The latter method was used by Côté, Dell'Amico and Iori [11] for the SPP.

Our master problem is an extension of the *One-Dimensional Contiguous Bin-Packing Problem* (1CBP), a relaxation that was introduced by Martello, Monaci, and Vigo [1] for the SPP. For the SPP, the 1CBP is obtained by horizontally 'cutting' each two-dimensional $w_j \times h_j$ item into h_j unit height *slices* of length w_j, and the objective is to pack all resulting slices into the minimum number of one-dimensional bins of capacity W so that slices belonging to the same item are packed into contiguous bins.

The master problem we developed solves a variant of the 1CBP in which each item j is vertically cut into w_j unit width slices of height h_j. It then looks for a feasible solution of threshold value LB (the capacity of the one-dimensional bin) that contiguously packs the slices into at most W bins. In addition, it is necessary to take into account the possibility of rotating the items, and hence both copies of each two-dimensional item (see Section 3) are cut into slices, and we impose that the solution packs only one of the two sets of slices. Figure 1(a) provides a pictorial representation of a master problem solution.

In Section 6 we describe the approach we developed for solving the master problem. Once the master has been solved, the slave must check the feasibility of the resulting solution for the original SCP by looking for a rearrangement of the slices that reconstructs the original two-dimensional items without exceeding the height of the strip produced by the master.

Figure 1 (a) master problem solution; (b) corresponding SCP solution



(a) (b)

Figure 1(b) shows an SCP solution corresponding to the master solution of Figure 1(a). In Section 7 we discuss the approach we adopted for its solution.

The overall approach we propose starts with preprocessing and sets a possible strip height at a threshold provided by a lower bound. At each iteration, it looks for a feasible solution of value equal to the current threshold through the above master-slave method. If it can prove that no such solution exists, then the threshold is conveniently increased. The method can be summarized as follows.

Algorithm SINGLE:
1. reduce the instance (see Section 4);
2. execute the best-fit heuristics to obtain an upper bound UB, and compute lower bound LB (see Section 4);
3. execute the truncated branch-and-bound heuristic (see Section 4);
4. **if** a solution of value LB has been obtained **then** $UB := LB$ and **terminate**;
while $UB > LB$ **do**
5. exactly solve the master problem with threshold height LB (see Section 4);
6. **if** no solution of value LB is found **then** remove all cuts, increase LB, and **continue**;
7. exactly solve the slave problem;
8. **if** a solution of value LB is obtained **then** $UB := LB$ and **terminate**;
9. add improved Benders' cuts to the master problem (see Section 7)
end while.

In many cases, the value of UB is not much higher than that of LB, and hence increasing LB by one at Step 6 is a reasonable choice. (In our computational experiments, we rarely found instances with $z > LB + 1$.) For different cases, a binary search between LB and UB could be preferable.

6. Master problem

As mentioned in Section 5, the aim of the master problem is to find a feasible solution of given value LB (threshold) to an adaptation of the one-dimensional bin packing problem with contiguity constraints. In [11], the master arising from the SPP was solved using two exact algorithms: a combinatorial branch-and-bound and, when it fails due to time limit, the Benders' decomposition of a mathematical model given by the oriented version of (1), (2), and (5), with an additional constraint imposing that, for any column, the sum of the heights of the items engaging it does not exceed z.

Our master was solved through an ILP model of an adaptation of the 1CBP introduced in [1] for the SPP (see Section 5). The type of modeling we adopted has its roots in the ARCFLOW model proposed by Valério de Carvalho [38] for the one-dimensional cutting stock problem. (Note that, in spite of its similar name, this problem is totally different from the two-dimensional problem we are considering.) We make use of a directed graph with $W + 1$ vertices $0, 1, \ldots, W$, and arc set $A = A_1 \cup \cdots \cup A_{2n} \cup A_0$ where

$$A_j = \{(d, e) : 0 \leq d < e \leq W \text{ and } e - d = w_j\} \ (j = 1, \ldots, 2n) \tag{7}$$

and $A_0 = \{(d, d + 1) : 0 \leq d < W\}$. In other words, there is an arc between two vertices d and e if there is an item (either an input item or a rotated copy) whose width is equal to

$e-d$. In addition, there is a set A_0 of *loss arcs*, corresponding to identical dummy items having unit width and height, used for ensuring flow conservation, as it will be clear from the model.

Consider the following numerical example: $n=3$, $W=5$, $w_1=5$, $h_1=1$, $w_2=h_2=3$, $w_3=h_3=2$: Figure 2 shows the resulting graph (disregard by the moment the values on the arcs, which give the item heights). The topmost arc corresponds to input item 1, non rotated (A_1). Then we have the three arcs corresponding to input item 2 ($A_2 \equiv A_5$) and the four arcs corresponding to input item 3 ($A_3 \equiv A_6$). Finally , the loss arcs (A_0, dotted) and, on the bottom, the five arcs corresponding to the rotated copy of item 1 (A_4). (Note that, formally, one should also have a second, useless, copy of the arc sets corresponding to input items 2 and 3, for which, however, rotation does not make sense.)

Let us introduce variables $\bar{x}_{jde}$ ($j=0,\ldots,2n$, $d=0,\ldots,W-1$, $e=1,\ldots,W$) to represent the number of times arc $(d,e)\in A_j$ is selected. For $j=1,2,\ldots,2n$, binary variable $\bar{x}_{jde}$ takes the value 1 iff item j is selected and packed with its bottom left corner at abscissa d, engaging columns $d,d+1,\ldots,e-1$. For $j=0$, $\bar{x}_{jde}$ is a non-negative integer variable giving the number of 1×1 dummy items packed at abscissa d. Let us denote by $\delta_j^-(e)$ (resp. $\delta_j^+(e)$) the set of arcs of A_j entering (resp. emanating from) vertex e. The master problem is then to find a feasible solution to

$$\sum_{j=0}^{2n}\left(-\sum_{(d,e)\in\delta_j^-(e)} h_j\bar{x}_{jde}+\sum_{(e,f)\in\delta_j^+(e)} h_j\bar{x}_{jef}\right)=\begin{cases} LB & \text{if } e=0;\\ -LB & \text{if } e=W;\\ 0 & \text{otherwise,}\end{cases} \quad e=0,1,\ldots,W, \tag{8}$$

$$\sum_{(d,e)\in A_j}\bar{x}_{jde}+\sum_{(d,e)\in A_{n+j}}\bar{x}_{(n+j)de}=1 \qquad j=1,2,\ldots,n, \tag{9}$$

$$\bar{x}_{jde}\in\{0,1\} \qquad j=1,2\ldots,2n;(d,e)\in A_j, \tag{10}$$

$$\bar{x}_{0de}\geq 0,\text{integer} \qquad (d,e)\in A_0. \tag{11}$$

Constraints (8) impose: (i) the solution value LB to the flow emanating from node 0 and to that entering node W (hence ensuring that every item is entirely packed within the strip), and (ii) the flow conservation at nodes $1,\ldots,W-1$. Constraints (9) impose that, for each item j, either the original item or its rotated version is packed. The model has $W+n+1$ constraints and $O(nW)$ variables (as, from (7), each arc set A_j has no more than W arcs).

In Figure 2, the values on the arcs represent the height of the corresponding items, i.e., in the model, the flow that circulates along the arc every time it is selected. For the previous numerical example, Figure 3 shows the arcs selected in a feasible solution of threshold value

Figure 2 Arcs generated for the example instance

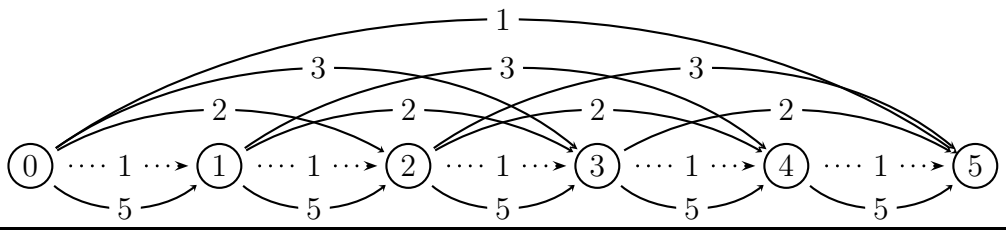
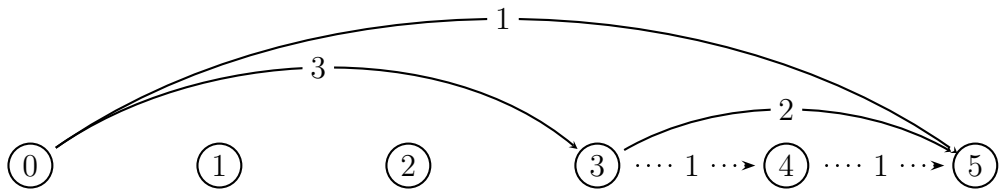


Figure 3 Set of arcs selected in the optimal solution of the master for the example instance



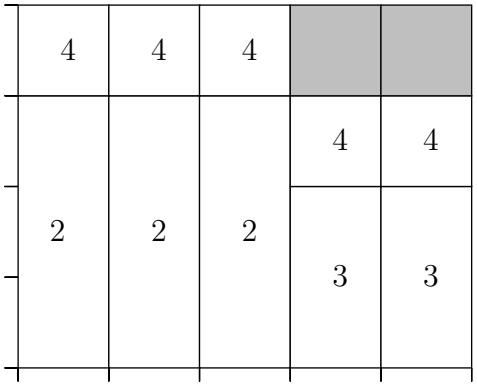
3, while Figure 4 provides a possible graphical representation of such solution. The non-zero $\bar{x}_{jde}$ values are

- $\bar{x}_{2,0,3} = 1$, i.e., item 2 has its bottom left corner packed at abscissa 0, it engages columns $0, 1, 2$, occupying three height units in each of them;
- $\bar{x}_{3,3,5} = 1$, i.e., item 3 has its bottom left corner packed at abscissa 3, it engages columns $3, 4$, occupying two height units in each of them;
- $\bar{x}_{4,0,5} = 1$, i.e., item 4 (the rotated copy of item 1) has its bottom left corner packed at abscissa 0, it engages columns $0, 1, 2, 3, 4$, occupying one height unit in each of them;
- $\bar{x}_{0,3,4} = \bar{x}_{0,4,5} = 1$, i.e., loss arcs engage one height unit (waste) in columns 3 and 4 (to satisfy the flow conservation constraints).

7. Slave problem and cut generation

Let $[\bar{x}^s_{jde}]$ be the solution to the current master problem, and z^s (equal to the current threshold LB) its value. The solution identifies a subset $\mathcal{N}^s \subset \mathcal{N}$ containing one copy (either rotated or not) of each of the n input items, together with the corresponding abscissa, and hence the columns engaged by the item. The slave problem, *y-check* in the following, is then to decide if there exists an ordinate for each item $j \in \mathcal{N}^s$ such that all items are packed without overlapping and without exceeding the height of the strip. This problem is known to be $\mathcal{NP}$-hard, as shown by Côté, Dell'Amico, and Iori [11].

Figure 4 Graphical representation of an optimal master solution for the example instance



The y-check problem arising for the SPP was attacked in [11] through a specialized branch-and-bound algorithm. We preferred to explore the possibility of obtaining an effective overall algorithm by solving the slave through a simpler, non-tailored constraint programming approach. Remind that, for each item j, w_j and h_j give its width and height in the selected orientation. The slave problem is then to find ordinates y_j ($j \in \mathcal{N}^s$) satisfying (see (1)-(6))

$$y_j + h_j \leq z^s \qquad j \in \mathcal{N}^s, \tag{12}$$

$$\text{nonoverlap} \left\{ [y_j, y_j + h_j], j \in \mathcal{N}^s : \sum_{p \in \mathcal{W}(j,q)} \bar{x}^s_{jp(p+w_j)} = 1 \right\} q \in \mathcal{W}, \tag{13}$$

$$y_j \geq 0, \text{ integer} \qquad j \in \mathcal{N}^s. \tag{14}$$

Constraints (12) impose that no item exceeds the height of the strip, while constraints (13) impose that no two items overlap. If the slave problem returns a solution satisfying (12)-(14), then we know that the original SCP instance has been optimally solved.

If instead the slave can prove that no feasible solution to (12)-(14) exists, we generate a set of lifted combinatorial Benders' cuts, following the method developed in [11] for the SPP:

(i) heuristically find a (possibly small) subset $\widetilde{\mathcal{N}}^s \subseteq \mathcal{N}^s$ of items such that any solution including them at their current abscissa is infeasible for the slave;

(ii) solve an LP to identify, for each item $j \in \widetilde{\mathcal{N}}^s$, an interval $[l^s_j, r^s_j]$ that includes its current abscissa, and has the following property: If all the items $j \in \widetilde{\mathcal{N}}^s$ are in the master solution, each with the abscissa in $[l^s_j, r^s_j]$, then the same (infeasible) slave problem will be obtained;

(iii) add a cut to prohibit this, i.e.,

$$\sum_{j \in \widetilde{\mathcal{N}}^s} \sum_{(d,e) \in A_j, l^s_j \leq d \leq r^s_j} \bar{x}_{jde} \leq |\widetilde{\mathcal{N}}^s| - 1. \tag{15}$$

Observe that cuts (15) are only valid for a given threshold value $z^s = LB$: a certain master solution $[\bar{x}^s_{jde}]$ can be infeasible for LB, but feasible, at a subsequent iteration, for a higher threshold value. This corrects an imprecision in the overall descriptive model of Section 2.3 in [11], that was pointed out by Hashimoto, Hu, Imahori, and Yagiura [39].

8. The case of identical item copies

The approach we have introduced so far is obviously valid if the input instance includes a number of identical items. For such cases however, a more effective algorithm, called MULTI in the following, can be obtained through simple modifications of SINGLE:

- at Step **1**, the item width increase (b) of Section 4 is deactivated in order to avoid the creation of dissimilarities among copies of identical items;

- at Step **5**, we can group together identical items: let m denote the number of items that are different from each other and b_j the number of copies of item j ($j = 1, 2, \ldots, m$). We then perform a modified version of the master (8)-(11), in which n is replaced by m, constraints (9) become

$$\sum_{(d,e)\in A_j} \bar{x}_{jde} + \sum_{(d,e)\in A_{m+j}} \bar{x}_{(m+j)de} = b_j \quad j = 1, 2, \ldots, m, \tag{16}$$

  while (10) and (11) merge to

$$\bar{x}_{jde} \geq 0, \text{integer} \quad j = 0, 1, \ldots, 2m; (d, e) \in A_j. \tag{17}$$

  For $j = 1, 2, \ldots, 2m$, $\bar{x}_{jde}$ is now a non-negative integer variable giving the number of copies of item j that are selected and packed with their bottom left corner at abscissa d, engaging columns $d, d+1, \ldots, e-1$. The meaning of $\bar{x}_{0de}$ does not change. Once the master problem has been solved, we map its solution into the original instance;

- at Step **7**, for pairs of identical items (j, k) ($j, k \in \mathcal{N}^s$, $j < k$) assigned to the same abscissa, it is imposed that $y_j < y_k$, in order to avoid symmetries in the slave problem;

- at Step **9**, as constraint (15) does not extend to general integer variables, when the slave proves that (12)-(14) has no solution, instead of adding a cut, we kill the decision node producing $\bar{x}_{jde}$ in the master enumeration tree.

Our overall approach, referred to as DIM in the following, executes algorithm SINGLE of Section 5 when all items are different, and algorithm MULTI otherwise.

9. Computational experiments

In order to test the effectiveness of the proposed approach we performed extensive computational experiments on classical SCP instances from the literature. As we will see, our approach can be easily modified to handle relevant variants of the problem. We thus also tested it on literature instances of the square and rectangle packing problems PSS and PRSR, as well as of the pallet loading problem PLP (see Section 2).

We used CPLEX 12.6.0 for all steps of the algorithm: master, slave, and cuts. The logical constraints used in the slave were `IloNoOverlap` (which models constraints (13)) and in addition, for MULTI, `IloEndBeforeStart` (to avoid symmetries). In the implementation of SINGLE, the slave problem was invoked within the `lazy callback` procedure, while in that of MULTI it was invoked within the `incumbent callback` procedure. Both procedures are available in the branch-and-cut framework, so, as soon as the master produces a decision node with integer solution, the slave checks it for feasibility: if the check fails, SINGLE adds new cuts (15) to the master, while MULTI just kills the corresponding decision node. This implementation of the logic based Benders' decomposition is sometimes called *branch-and-check* (see Thorsteinsson [40]).

The experiments were performed on an Intel Xeon E3-1220 3.10 GHz with 8 GB RAM, equipped with four cores. In order to have a fair comparison with other algorithms and

machines, we always used a single core. The time limit was set to 3 600 seconds per instance, with a limit of 10 seconds for the execution of the truncated heuristic of Section 4.

The performance of DIM was compared with that of other algorithms, in most cases using the results published by their authors. As the involved computers used have different speeds, we evaluated the CPU performances using the indicators given by PassMark© Software (see `https://www.cpubenchmark.net/`). The indicator for our computer is 6 106. Concerning G-Staircase, the best algorithm among those presented by Kenmochi, Imamichi, Nonobe, Yagiura, and Nagamochi [18], the experiments were re-run using a working copy of the original code, kindly provided by the authors. (These experiments confirmed a good reliability of the PassMark indicators.)

Furthermore, different authors adopted different ways for handling the cases of time limit in the evaluation of the average CPU time: some included the time limits in the computations, some did not. In order to allow comparisons, our tables provide, for each algorithm, the average CPU times (in seconds) relative to the instances solved to proven optimality, and their number (# opt). The highest number of solved instances is highlighted in bold.

All the instances we used for our experiments can be downloaded from the library of codes and instances of the University of Bologna Operations Research Group, `http://or.dei.unibo.it/library`. In the following, we examine the outcome of the computational experiments on the different instance classes.

9.1. SCP instances

This is the main benchmark, as it refers to instances of the problem DIM is tailored for. We used the following classical two dimensional instances from the literature:

- NGCUT: a set of 12 knapsack instances proposed by Beasley [41];
- CGCUT: a set of 3 knapsack instances proposed by Christofides and Whitlock [42];
- GCUT: a set of 13 knapsack instances proposed by Beasley [43];
- BENG: a set of 10 packing instances proposed by Bengtsson [44];
- HT: a set of 9 SCP instances proposed by Hopper and Turton [45];
- BKW: a set of 13 SCP instances proposed by Burke, Kendall, and Whitwell [16];
- CLASS01, ..., CLASS10: ten sets of 50 bin packing instances each (six proposed by Berkey and Wang [46], and four later proposed by Martello and Vigo [47]). Each class is composed by five groups of 10 instances each, with $n = 20$, 40, 60, 80, and 100, respectively.

For all cases, but HT and BKW, the strip width W was set to the width of the original two-dimensional container (knapsack or bin).

The performance of Algorithm DIM was compared with that of the algorithms by Kenmochi, Imamichi, Nonobe, Yagiura, and Nagamochi [18] and by Arahori, Imamichi, and Nagamochi [19]. The experiments reported in the latter paper were made on a Pentium 4 3.0GHz, with a time limit of 3 600 seconds. The performance indicators of this machine

is 357, i.e., the entries in Table 1 for the algorithm in [19] should be multiplied by 0.058. For the instances for which a comparison with both other SCP algorithms can be done, it turns out that DIM always solved at least all the instances solved by the best between the algorithms in [18] and [19], sometimes with higher, sometimes with smaller CPU times. Similar results were obtained for instances CLASS*, that could only be compared with the working code provided by the authors of [18]. It can be observed that, for both algorithms, CLASS01 and CLASS02 are relatively easy, CLASS03 and CLASS04 are difficult, while CLASS05, CLASS06, CLASS07, CLASS08, and CLASS10 are extremely hard. The most relevant difference concerns CLASS09, for which DIM appears especially powerful. This could be explained by the fact that the instances of such class have a number of items with large width and height: their packing frequently creates large "holes" in the strip, and hence lower bounds are not tight, while the ARCFLOW model performs well because the number of arcs is small.

By comparing our experiments with those in [11], we can observe that allowing item rotation considerably increases the difficulty of the problem. For example, the instances of CLASS07 (which were all solved in the oriented case) have many oblong horizontal items: this allows a powerful initial reduction through dominance criteria when the items are oriented (which makes the instance quite easy to solve), but no reduction at all when they can be rotated.

Worth is mentioning that we solved for the first time instance GCUT02: The optimal solution is reported in Figure 5, where item numbers and (possibly rotated) sizes are provided within the rectangles. The solution was obtained in 108 CPU seconds.

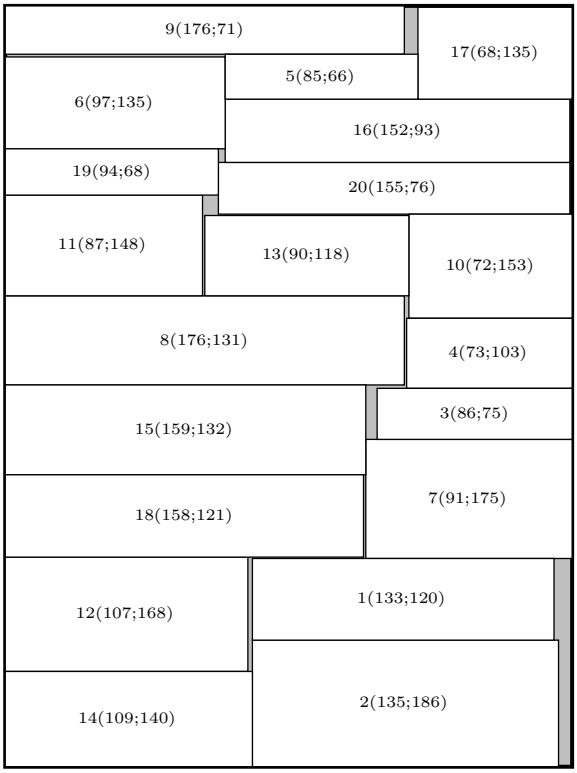
9.2. Rectangle packings

In this section we deal with the rectangle packing problems mentioned in Section 2. Given n rectangular items of integer width w_j and height h_j $(j = 1, \ldots, n)$, problem PRSR

Table 1: SCP instances. CPU times to be multiplied by 0.058 for [19]

Name	# inst.	Arahori et. al [19]		Kenmochi et. al [18]		Algorithm DIM	
		# opt.	Avg. time (s)	# opt.	Avg. time (s)	# opt.	Avg. time (s)
NGCUT	12	**12**	0.4	11	1.0	**12**	25.8
CGCUT	3	**2**	0.0	**2**	0.2	**2**	0.2
GCUT1-4	4	1	0.8	1	141.6	**2**	59.2
GCUT5-13	9	-	-	2	425.7	**5**	35.9
BENG	10	**10**	0.1	**10**	0.2	**10**	0.2
HT	9	**9**	0.0	**9**	0.1	**9**	0.2
BKW1-12	12	-	-	**9**	6.7	**9**	1.0
BKW13	1	-	-	0	-	0	-
CLASS01	50	-	-	44	2	**50**	0.4
CLASS02	50	-	-	**50**	0	**50**	0.3
CLASS03	50	-	-	2	82.9	**9**	846.2
CLASS04	50	-	-	18	35.6	**19**	145.3
CLASS05	50	-	-	0	-	**1**	1289.4
CLASS06	50	-	-	0	-	0	-
CLASS07	50	-	-	0	-	0	-
CLASS08	50	-	-	0	-	0	-
CLASS09	50	-	-	0	-	**45**	293.3
CLASS10	50	-	-	0	-	**2**	1400.4

Figure 5 Optimal solution found for GCUT02 ($W = 250$, $z_{opt} = 1118$)



asks for the minimum area square needed to pack all the items without overlapping. The special case in which $w_j = h_j$ for all j is denoted as PSS. Both problems can be solved through Algorithm DIM as follows. We set $W = LB = \left\lceil \sqrt{\sum_{j=1}^{n} w_j h_j} \right\rceil$, and we execute DIM. While no feasible solution is found, we increase both W and LB by one unit, and iterate. We evaluated this approach on classical instances from the literature, and compared our results with those obtained by Martello and Monaci [23] through a specialized algorithm that attempts squares of different size through binary search, namely:

- NGCUT, CGCUT, GCUT, BENG: see Section 9.1;
- GARD: PSS instances with $w_j = h_j = j$ for $j = 1, \ldots, n$, proposed by Gardner [48];
- KORF: PRSR instances with $w_j = j$, $h_j = j + 1$ for $j = 1, \ldots, n$, proposed by Korf, Moffitt, and Pollack [49].
- RND_S: 200 PSS instances proposed by Martello and Monaci [23];
- RND_R: 200 PRSR instances proposed by Martello and Monaci [23].

Both RND_S and RND_R consist of four groups of 50 instances each, with $n = 5$, 10, 15, and 20, respectively.

The experiments in [23] were made on an Intel i5-750 CPU (2.67 GHz), using IBM-ILOG Cplex 12.5.1 and Gurobi 5.6, with a time limit of 3 600 seconds per instance. The performance

indicator for such computer is 3 732, so the entries given in Table 2 for this algorithm should be multiplied by 0.611. The results show that Algorithm DIM always solved at least all the instances solved in [23], in several cases with smaller CPU times. In particular, all BENG instances were solved to proven optimality, and 5 more GARD instances were solved with respect to [23]. (Note however that 6 additional instances were solved by Korf, Moffitt, and Pollack [49] through a highly specialized algorithm.) Worth is mentioning that instances RND_* with $n \geq 15$ confirm to be very difficult to solve exactly. We tested DIM with a larger time limit on the only RND_R 10 instance it could not solve: a proven optimal solution was found in 3 918 seconds.

Table 2: Rectangle packing instances. CPU times to be multiplied by 0.611 for [23]

Name	# inst.	Martello and Monaci [23]		Algorithm DIM	
		# opt.	Avg. time (s)	# opt.	Avg. time (s)
NGCUT	12	10	357.1	**12**	67.1
BENG	10	3	0.0	**10**	0.2
CGCUT	3	0	-	**1**	734.6
GCUT	13	**3**	560	**3**	742.7
KORF	40	17	73.6	**24**	2.2
GARD	40	16	2.2	**21**	191.1
RND_S 05	50	**50**	0.0	**50**	0.3
RND_S 10	50	**50**	341.2	**50**	81.8
RND_S 15	50	**5**	186.9	**5**	894.3
RND_S 20	50	0	-	0	-
RND_R 05	50	**50**	0.0	**50**	0.4
RND_R 10	50	39	384.2	**49**	464.2
RND_R 15	50	0	-	0	-
RND_R 20	50	0	-	0	-

9.3. Pallet loading

Our last set of experiments was performed on instances of the pallet loading problem (PLP). Given a rectangle of width W and height H, the problem is to pack into it, without overlapping, the maximum number of identical rectangular items of integer width w_1 and height h_1. We attacked the problem through Algorithm DIM as follows. We initialize n to $\left\lfloor \frac{WH}{w_1 h_1} \right\rfloor$ (an obvious upper bound on the solution value), we define $w_j = w_1$, $h_j = h_1$ for $j = 2, \ldots, n$, we set $LB = H$, and we execute DIM. While no feasible solution is found, we decrease n by one, and iterate.

The PLP has been intensively studied since the sixties, and many specialized algorithms have been proposed for its solution. The comprehensive recent survey by Silva, Oliveira, and Wäscher [30] has a bibliography of over fifty references, and examines computational experiments from the literature involving some three million instances. Additional experiments are reported by Ahn, Park, and Yoon [27]. (The two articles appeared almost at the same time, so they do not cite each other.).

Although the MULTI version of our approach exploits the identical item property, it is not guided in any way by the very strong property that *all* items are identical, nor it uses the powerful heuristics and upper bounds that such property exploits. Our objective here was not to beat very specialized algorithms, but to see if we could solve to proven optimality some previously unsolved, or very difficult to solve, instances from the literature. We thus tested MULTI on

- GROUP1, ..., GROUP5: five sets, each containing between 10 and 21 instances, used by Ahn, Park, and Yoon [27];
- HARD II: a set of 241 instances that were identified by Alvarez-Valdes, Parreño, and Tamarit [26] as the most difficult ones among the 40 609 "Cover II" instances they tested. (All these instances have been solved to proven optimality (see Birgin, Lobato, and Morabito [29]);
- HARD III: a subset of 970 instances for which optimality is hard to prove, among the 98016 initially proposed in set "Cover III" by Alvarez-Valdes, Parreño, and Tamarit [50]. Attempts to exactly solve such instances, identified by Birgin, Lobato, and Morabito [29], is currently underway (see the web page `http://lagrange.ime.usp.br/~lobato/packing/cover3.php` they maintain);
- SAMPLE59: a set of 59 instances selected by Silva, Oliveira, and Wäscher [30] as particularly significant, as they have been used as benchmark instances in at least two numerical experiments from the literature.

For all instances but most of those in HARD III, the optimal solution is known. The available information about the solution times is not uniform. For GROUP*, Ahn, Park, and Yoon [27] provide the average CPU times for their exact solution, obtained on an Intel I5-750 CPU, 3 GB, with performance indicator 4 277 (i.e., their times should be multiplied by 0.7). For HARD II, Alvarez-Valdes, Parreño and Tamarit [26] provide the median CPU time obtained by using CPLEX 7.0 on a PC Pentium III 850 MHz, CPLEX 7.0. We found no precise indicator for such computer: on the basis of indicators for similar machines, it should be very low (below 300), so their times should be multiplied by 0.05. For HARD III and SAMPLE59, no complete information on the solution times is available. In Table 3 we provide the comparable information for our approach.

The results show a good performance of DIM on all GROUP* instances but GROUP4 (probably because such instances are characterized by very large W and H values). The algorithm was also successful for HARD II instances, although no significant comparison with [26] can be done because of the big difference between the two CPLEX versions used. We solved to proven optimality the great majority of HARD III and SAMPLE59 instances within reasonable CPU times. The only unsolved SAMPLE59 instance is #58. Worth is mentioning that we solved 28 out of the 29 instances that have been indicated by Silva, Oliveira, and Wäscher [30] as those on which exact algorithms should be tested. Such instances (in parentheses the CPU times) are: #18 (7.4 s), #40 (0.6 s), #42 to #45 (0.3 s, 15.1 s, 1.5 s, 80.5 s), #26 to #34 (each in less than 0.7 s), #46 to #51 (each in less than 1.0 s), #52 (68.0 s), #53 (0.2 s), #54 (236.8 s), #55 (697.2 s), #56 (2.1 s), #57 (0.7 s), #58 (unsolved after 3 600 s), and #59 (5.5 s).

The results obtained on HARD III are particularly relevant, as optimality was still not proven for most (683) of these 970 instances. We were able to solve exactly 849 instances. In all such cases, it turned out that the lower bound found by the most powerful PLP heuristics was the optimal solution value, enforcing the conjecture of Birgin, Lobato, and Morabito [29] that their recursive partitioning approach always finds the optimal solution.

Table 3: Pallet loading instances. CPU times to be multiplied by 0.700 for [27], and by 0.05 for [26]

Name	# inst.	Ahn et. al [27]		Alvarez-Valdes et. al [26]		Algorithm DIM		
		# opt.	Avg. time (s)	# opt.	Med. time (s)	# opt.	Avg. time (s)	Med. time (s)
G1	17	**17**	0.0			**17**	0.3	0.3
G2	21	20	2.3			**21**	4.6	0.4
G3	17	**17**	2.9			**17**	8.3	0.4
G4	15	**14**	530.5			9	11.3	39.9
G5	10	8	278.8			**10**	20.1	7.3
HARD II	241			204	506.9	**237**	17.9	1.3
HARD III	970					**849**	282.8	0.3
SAMPLE59	59					**58**	19.5	0.2

10. Conclusion

We have presented a logic based Benders' decomposition approach for the orthogonal stock cutting problem. We initially consider as the master a relaxation of the problem, that consists of a one-dimensional bin packing problem with contiguity constraints. We solve it through an ILP model based on the ARCFLOW formulation. The slave then checks, through constraint programming, if the solution obtained can lead to a feasible solution for the original problem. If the attempt is unsuccessful, we prevent the master to replicate the current solution, either by adding cuts or by killing the corresponding decision node, and iterate. Computational experiments on classical benchmarks from the literature show that the algorithm compares favorably with state-of-the art approaches. In particular, it solved for the first time instance GCUT02. We also tested an adaptation of the algorithm to other relevant problems. For rectangle packing problems, the algorithm compared favorably with a recent specialized algorithm. It also showed a good performance on hard instances of the pallet loading problem, solving 849 out of 970 instances for most of which the known solutions were not proved to be optimal.

Acknowledgements

We are grateful to Mutsunori Yagiura for useful discussions. We thank him and his co-authors for providing a working copy of their computer code developed for [18]. We thank an anonymous referee for helpful comments. The second author acknowledges financial support from Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), under grant PVE no. A007/2013.

References

[1] S. Martello, M. Monaci, D. Vigo, An exact approach to the strip-packing problem, INFORMS Journal on Computing 15 (3) (2003) 310–319.

[2] A. Lodi, S. Martello, M. Monaci, Two-dimensional packing problems: A survey, European Journal of Operational Research 141 (2) (2002) 241–252.

[3] M. Delorme, M. Iori, S. Martello, Bin packing and cutting stock problems: Mathematical models and exact algorithms, European Journal of Operational Research 255 (1) (2016) 1 – 20.

[4] A. Lodi, S. Martello, D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS Journal on Computing 11 (4) (1999) 345–357.

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

[6] N. Lesh, J. Marks, A. McMahon, M. Mitzenmacher, Exhaustive approaches to 2D rectangular perfect packings, Information Processing Letters 90 (1) (2004) 7–14.

[7] R. Alvarez-Valdes, F. Parreño, J. Tamarit, A branch and bound algorithm for the strip packing problem, OR Spectrum 31 (2) (2009) 431–459.

[8] M. Boschetti, L. Montaletti, An exact algorithm for the two-dimensional strip-packing problem, Operations Research 58 (6) (2010) 1774–1791.

[9] J. Westerlund, L. Papageorgiou, T. Westerlund, A MILP model for N-dimensional allocation, Computers & Chemical Engineering 31 (12) (2007) 1702–1714.

[10] P. Castro, J. Oliveira, Scheduling inspired models for two-dimensional packing problems, European Journal of Operational Research 215 (1) (2011) 45–56.

[11] J. Côté, M. Dell'Amico, M. Iori, Combinatorial Benders' cuts for the strip packing problem, Operations Research 62 (3) (2014) 643–661.

[12] S. Jakobs, On genetic algorithms for the packing of polygons, European Journal of Operational Research 88 (1) (1996) 165–181.

[13] M. Iori, S. Martello, M. Monaci, Metaheuristic algorithms for the strip packing problem, in: P. Pardalos, V. Korotkich (Eds.), Optimization and Industry: New Frontiers, Kluwer, Boston, 2003, pp. 159–179.

[14] E. Özcan, Z. Kai, J. Drake, Bidirectional best-fit heuristic considering compound placement for two dimensional orthogonal rectangular strip packing, Expert Systems with Applications 40 (10) (2013) 4035–4043.

[15] J. Thomas, N. Chaudhari, A new metaheuristic genetic-based placement algorithm for 2D strip packing, Journal of Industrial Engineering International 10 (1) (2014) 1–16.

[16] E. Burke, G. Kendall, G. Whitwell, A new placement heuristic for the orthogonal stock-cutting problem, Operations Research 52 (4) (2004) 655–671.

[17] L. Wei, W. Oon, W. Zhu, A. Lim, A skyline heuristic for the 2D rectangular packing and strip packing problems, European Journal of Operational Research 215 (2) (2011) 337–346.

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

[19] Y. Arahori, T. Imamichi, H. Nagamochi, An exact strip packing algorithm based on canonical forms, Computers & Operations Research 39 (12) (2012) 2991–3011.

[20] C. Picouleau, Worst-case analysis of fast heuristics for packing squares into a square, Theoretical Computer Science 164 (1-2) (1996) 59–72.

[21] A. Caprara, A. Lodi, S. Martello, M. Monaci, Packing into the smallest square: Worst-case analysis of lower bounds, Discrete Optimization 3 (4) (2006) 317–326.

[22] J. Correa, Resource augmentation in two-dimensional packing with orthogonal rotations, Operations Research Letters 34 (1) (2006) 85–93.

[23] S. Martello, M. Monaci, Models and algorithms for packing rectangles into the smallest square, Computers & Operations Research 63 (2015) 161–171.

[24] S. Barnett, G. Kynch, Exact solution of a simple cutting problem, Operations Research 15 (6) (1967) 1051–1056.

[25] K. Dowsland, An exact algorithm for the pallet loading problem, European Journal of Operational Research 31 (1) (1987) 78–84.

[26] R. Alvarez-Valdes, F. Parreño, J. Tamarit, A branch-and-cut algorithm for the pallet loading problem, Computers & Operations Research 32 (11) (2005) 3007–3029.

[27] S. Ahn, C. Park, K. Yoon, An improved best-first branch and bound algorithm for the pallet-loading problem using a staircase structure, Expert Systems with Applications 42 (21) (2015) 7676–7683.

[28] L. Lins, S. Lins, R. Morabito, An l-approach for packing (l,w)-rectangles into rectangular and l-shaped pieces, Journal of the Operational Research Society 54 (7) (2003) 777–789.

[29] E. Birgin, R. Lobato, R. Morabito, An effective recursive partitioning approach for the packing of identical rectangles in a rectangle, Journal of the Operational Research Society 61 (2) (2010) 306–320.

[30] E. Silva, J. Oliveira, G. Wäscher, The pallet loading problem: a review of solution methods and computational experiments, International Transactions in Operational Research 23 (1–2) (2016) 147–172.

[31] J. Hooker, Logic-based methods for optimization: combining optimization and constraint satisfaction, John Wiley & Sons, 2000.

[32] J. Hooker, G. Ottosson, Logic-based Benders decomposition, Mathematical Programming 96 (1) (2003) 33–60.

[33] J. Hooker, Planning and scheduling by logic-based Benders decomposition, Operations Research 55 (3) (2007) 588–602.

[34] G. Codato, M. Fischetti, Combinatorial Benders' cuts for mixed-integer linear programming, Operations Research 54 (4) (2006) 756–766.

[35] F. Clautiaux, A. Jouglet, J. Carlier, A. Moukrim, A new constraint programming approach for the orthogonal packing problem, Computers & Operations Research 35 (3) (2008) 944–959.

[36] R. Baldacci, M. Boschetti, A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem, European Journal of Operational Research 183 (3) (2007) 1136 – 1149.

[37] J. Benders, Partitioning procedures for solving mixed-variables programming problems, Numerische Mathematik 4 (1) (1962) 238–252.

[38] J. Valério de Carvalho, Exact solution of bin packing problems using column generation and branch and bound, Annals of Operations Research 86 (1999) 629–659.

[39] H. Hashimoto, Y. Hu, S. Imahori, M. Yagiura, Private communication (2015).

[40] E. Thorsteinsson, Branch and check: A hybrid framework integrating mixed integer programming and constraint programming, in: Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming (CP2001), Springer-Verlag, Berlin, 2001, pp. 16–30.

[41] J. Beasley, An exact two-dimensional non-guillotine cutting tree search procedure, Operations Research 33 (1) (1985) 49–64.

[42] N. Christofides, C. Whitlock, An algorithm for two-dimensional cutting problems, Operations Research 25 (1) (1977) 30–44.

[43] J. Beasley, Algorithms for unconstrained two-dimensional guillotine cutting, Journal of the Operational Research Society 36 (4) (1985) 297–306.

[44] B. Bengtsson, Packing rectangular pieces-a heuristic approach, The Computer Journal 25 (3) (1982) 353–357.

[45] E. Hopper, B. Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, European Journal of Operational Research 128 (1) (2001) 34–57.

[46] J. Berkey, P. Wang, Two-dimensional finite bin-packing algorithms, Journal of the Operational Research Society 38 (5) (1987) 423–429.

[47] S. Martello, D. Vigo, Exact solution of the two-dimensional finite bin packing problem, Management Science 44 (3) (1998) 388–399.

[48] G. M., Mathematical games: the problem of Mrs. Perkins' quilt, and answers to last month's puzzles, Scientific American 215 (3) (1966) 264–272.

[49] R. Korf, M. Moffitt, M. Pollack, Optimal rectangle packing, Annals of Operations Research 179 (1) (2010) 261–295.

[50] R. Alvarez-Valdes, F. Parreño, J. Tamarit, A tabu search algorithm for the pallet loading problem, OR Spectrum 27 (1) (2005) 43–61.