Combinatorial Benders' Decomposition for the Orthogonal Stock Cutting Problem

Maxence Delorme
DEI "Guglielmo Marconi", University of Bologna
Viale Risorgimento 2, 40136, Bologna, Italy
maxence.delorme2@unibo.it

Manuel Iori
DISMI, University of Modena and Reggio Emilia
Via Amendola 2, Pad. Buccola, 42122, Reggio Emilia, Italy
manuel.iori@unimore.it

Silvano Martello
DEI "Guglielmo Marconi", University of Bologna
Viale Risorgimento 2, 40136, Bologna, Italy
silvano.martello@unibo.it

ABSTRACT

We study the orthogonal Stock Cutting Problem (SCP), whose aim is to pack a set of rectangular items into a strip of fixed width without overlapping and using the minimum height. Items must be packed with their edges parallel to those of the strip, but rotation by 90° is allowed. The problem is important because it has many real world applications, especially in wood, paper, glass, and metal industries, just to cite some. The SCP is usually solved through branch-and-bound algorithms. We propose an alternative method, based on Benders' decomposition, that was previously used with success for the Strip Packing Problem (the counterpart of the SCP in which rotation is not allowed). We describe the steps of our algorithm, expose some preliminary computational experiments, and compare them with the state of the art exact approaches introduced in the literature. Preliminary results show that our approach provides very competitive results when it is hybridized with a powerful branch-and-bound algorithm. Moreover, it solves for the first time to proven optimality instance "gcut02", which has been an open problem for quite a long time.

KEYWORDS. Orthogonal Stock Cutting Problem. Bender's decomposition. Exact algorithm.

Main Area: Cutting and Packing Problems

1. Introduction

Given n rectangular items of 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 height. Items must be packed with their edges parallel to those of the strip, but rotation by 90° is allowed. The 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 [18]). Both the SCP and the SPP are important because they have many real world applications, especially in wood, paper, glass, and metal industries, just to cite some.

The two problems can possibly lead to different optimal solutions, as it is shown in Figure 1. Figure 1(a) depicts a solution for which rotation is not allowed, whose optimal height is 9, whereas Figure 1(b) shows the case where rotation is allowed and the optimal solution value is 8. It is therefore not surprising that most of the techniques proposed in the literature to solve these problems focus either on the case with rotation or on the case without it.

A recent approach proposed by Côté, Dell'Amico and Iori [11] uses the so-called *Combinatorial Benders' decomposition* to solve the SPP and shows interesting results. The main idea of this method, originally introduced by Benders [5], is to solve a difficult problem by means of the iterative solution of two subproblems, called *master problem* and *slave problem*. The master problem, which is usually a relaxed version of the original difficult problem, is in this case a *bin packing with contiguity constraints* (CBP). In the CBP, that was introduced by Martello, Monaci, and Vigo [18], each item is horizontally cut into unit height slices, and the aim is to pack all items into the minimum number of bins of capacity w so that slices belonging to the same item are packed into contiguous bins. The aim of the slave problem is to check if the optimal solution provided by the master is feasible for the original problem. If it is not, then a valid cut is added to the master to prevent such a solution to be regenerated. The process is iterated until the optimal solution found by the master is validated by the slave problem. Note that in the original approach by Benders [5] the subproblem is a continuous linear program, but later algorithms treated the case where the subproblem is an integer (possibly difficult) program (see, e.g., Hooker [13] and Codato and Fischetti [10]).

The aim of this paper is to continue this line of research and to propose a decomposition method that takes into account rotation of the items and solves to optimality the SCP. We propose a new ILP model for solving the CBP, based on an extension of the well-known ARCFLOW formulation introduced by Valério de Carvalho [19], and an original way to solve the slave problem. The resulting algorithm provides very encouraging results, closing the instance "gcut02", one of the instances introduced by Beasley [3], in the case where item rotation is allowed.

The paper is organized as follows. In Section 2 we review some successful approaches that have been proposed in the literature for the SCP and the SPP. Section 3 presents in details the different steps of our algorithm, and some experimental results are provided in Section 4.

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 [18] in

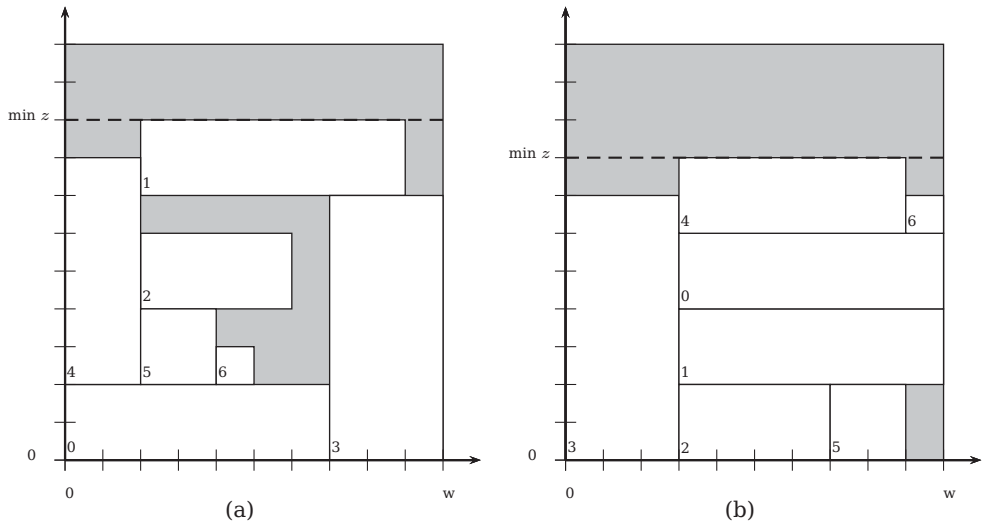

Figure 1: (a) an optimal SPP solution; (b) an optimal SCP solution

the early noughties. This algorithm makes use of preprocessing techniques, dominance criteria and a powerful lower bound (based on the CBP relaxation previously discussed). In the same period, Lesh, Marks, McMahon, and Mitzenmacher [17] proposed a branch-and-bound algorithm specifically tailored for perfect packing, i.e., for those cases in which the optimal solution has no loss space. 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 [1], and Boschetti and Montaletti [6] obtained better branch-and-bound algorithms by improving the dominance criteria and the bounding techniques proposed in [18].

Recently, new types of exact methods were proposed by Westerlund, Papageorgiou, and Westerlundtried [21] and Castro and Oliveira [8], who tried *Mixed Integer Linear Programming* (MILP) models to solve the SPP. Côté, Dell'Amico and Iori [11] introduced a 3-phase algorithm that uses Benders' decomposition to solve the problem. Starting from Iori, Martello and Monaci [15], several metaheuristic algorithms have also been proposed.

The literature on the SCP focused more on heuristics and metaheuristics rather than on exact methods, see, e.g., Burke, Kendall and Whitwel [7] or Wei, Oon and Lim [20]. To the best of our knowledge, the only exact approaches developed to tackle the SCP were proposed by Kenmochi, Imamichi, Nonobe, Yagiura, and Nagamochi [16] and Arahori, Imamichi, and Nagamochi [2], both based on branch-and-bound. The former algorithm transforms an instance into a perfect packing by adding some small items of size 1 x 1, and then applies a branch-and-bound using "DP cuts", based on the idea exposed in [17]. The latter algorithm adopts a more classical enumeration scheme and enriches it by adding innovative cuts. According to the computational experiments presented in [2] 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.

3. Algorithm

The aim of this section is to present the details of the algorithm that we implemented to solve the SCP. To this purpose we first need some notation. Let us define

- $N = \{1, 2, \ldots, n, n+1, \ldots, 2n\}$ = set of n items $(1, 2, \ldots, n)$ followed by a copy of them, rotated by 90° $(n+1, n+2, \ldots, 2n)$;
- $W = \{0, 1, \ldots, w-1\}$ = set of positions (also called columns in the following) along the width where the bottom left corner of an item can be packed;
- $W(j, q) = \{q, q+1, \ldots, q+w_j-1\}$ = columns occupied by item j when its bottom-left corner is packed in column q.

By introducing decision variables

- x_{jp} = binary variable taking the value 1 if the bottom-left corner of item j is packed in column p, and the value 0 otherwise;
- y_j = height at which the bottom edge of item j is packed;
- z = height at which the top edge of the topmost item is packed,

the SCP can be modeled as the following logical model:

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

$$\text{s.t.} \quad \sum_{p \in W} x_{jp} + x_{n+j,p} = 1 \qquad j = 1, 2, \ldots, n, \tag{2}$$

$$\sum_{j \in N} \sum_{p \in W(j,q)} h_j x_{jp} \leq z \qquad q \in W, \tag{3}$$

$$y_j + h_j \leq z \qquad j \in N, \tag{4}$$

$$\text{nonoverlap} \left\{ [y_j, y_j + h_j], j \in N : \sum_{p \in W(j,q)} x_{jp} = 1 \right\} \qquad q \in W, \tag{5}$$

$$x_{jp} \in \{0, 1\} \qquad j \in N, p \in W, \tag{6}$$

$$y_j \geq 0. \qquad j \in N. \tag{7}$$

Constraints (2) ensure that each item is packed, either rotated or not. Constraints (3) and (4) impose that no item go over the height of the strip. Note that constraints (3) are not strictly necessary, but are useful for our decomposition approach. Finally, logical constraints (5) prevent items to overlap. They can be used in constraint programming approaches and enforce items to be placed so that they do not intersect one another.

The above model is impractical for direct use (e.g., with a MILP solver, provided logical constraints (5) are transformed into valid MILP constraints) because the number of constraints (5) is generally very high. However, a decomposition may lead to reduced models that can be solved more easily. In the following, we propose an algorithm that makes use of this idea.

3.1. Overview of the algorithm

What is usually done for the SPP and the SCP [16, 12] is to fix the height of the strip, and to try and find a feasible solution. If the algorithm proves that no such solution exists, then the height of the strip is conveniently increased. Following this approach, our algorithm works as follows:

Phase 1 : Apply preprocessing to the widths of the strip and of the items.

Phase 2 : Compute a lower bound LB and an upper bound UB. If they are equal, then terminate.
Phase 3 : Solve the Master Problem with a fixed height equal to LB. If no solution is found, increase LB by one unit, and iterate Phase 3.
Phase 4 : Check the solution produced by the Slave Problem. If it is feasible, terminate with a proven optimal solution.
Phase 5 : Look for Benders' Cuts that violate the current solution, improve them and add them to the Master Problem. Go to Phase 3.

3.2. Preprocessing

The algorithm makes use of the preprocessing techniques proposed by Boschetti and Montaletti [6]. First, it computes the maximum total strip width $w' \leq w$ that can be obtained by packing side by side any subset of items:

$$\max \quad w' \tag{8}$$

$$\text{s.t.} \quad w' \leq w, \tag{9}$$

$$\sum_{j \in N} t_j w_j = w', \tag{10}$$

$$t_j + t_{n+j} \leq 1 \qquad j = 1, 2, \ldots, n, \tag{11}$$

$$t_j \in \{0, 1\}. \tag{12}$$

The optimal w' value can be computed through a standard dynamic programming. If w' is strictly less than w, then we reduce the strip width as $w := w'$. Secondly, our algorithm computes, in sequence for each item j, the maximum total width w'_j that can be packed side by side with item j without exceeding w. If $w_j + w'_j < w$, we can increase w_j to $w - w'_j$. This too can be obtained through dynamic programming. We also tried a third reduction procedure presented in [6] for the SPP, 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 direct application to the case with rotation appears to have little usefulness.

3.3. Initial lower and upper bounds

As mentioned in Section 1, several heuristics exist for the SCP. From the non-exhaustive list that we provided, we decided to use the Best-Fit heuristic of [7], for its simplicity and relative good performance. We implemented the versions proposed in [7], that differ from each other in the position where the next item is packed (left-most, rightmost, or a variant). We additionally implemented a simplified version of the branch-and-bound algorithm proposed in [16] using the so called *G-staircase placement* and the *DP cuts*, as its performance on certain type of instances was shown to be very efficient. As far as lower bounds are concerned, the simple lower bound $L_1 = \max(\lceil \sum_{j=1}^{n} w_j h_j / w \rceil, \max_{j=1,\ldots,n} \min\{w_j; h_j\})$ was used. We also computed an improved lower bound, called L_2, which is obtained by solving the continuous relaxation of the MILP described in the next subsection and rounding up to the next integer its solution value.

3.4. Master problem

As mentioned in Section 1, the aim of the master problem is to solve a bin packing with contiguity constraints. In other words, each item is horizontally cut into unit height slices, and the aim is to pack all items into the minimum number of bins

of capacity w so that slices belonging to the same item are packed into contiguous bins. In [11], this problem was solved either using model (1)-(3) and (6) or through a branch-and-bound algorithm. We decided to use a MILP model, but we transformed it to make it similar to an ARCFLOW model [19] with an additional constraint. This improves the efficiency of the model. To do so, we first create a set of arcs A, that are defined by (i) the item j they are associated with, and (ii) a starting point p along the width of the strip. We introduce variables x'_{jp} thats represent the number of times the corresponding arc is selected. We finally add *loss arcs* associated with dummy items 1x1, obtaining the following model:

$$\min \quad z \tag{13}$$

$$\text{s.t.} \quad -\sum_{(j,q)\in\delta^-(p)} h_j x'_{jq} + \sum_{(j,p)\in\delta^+(p)} h_j x'_{jp} = \begin{cases} z & \text{if } p = 0; \\ -z & \text{if } p = w; \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

$$\sum_{p\in W} x'_{jp} + x'_{n+j,p} = 1 \qquad j = 1, 2, \ldots, n, \tag{15}$$

$$x'_{jp} \geq 0, \text{integer} \qquad (j,p) \in A, \tag{16}$$

where $\delta^-(p)$ (resp. $\delta^+(p)$) denotes the set of arcs entering (resp. emanating from) p. Constraints (14) impose the flow conservation at all nodes, from $p = 1$ to $p = w - 1$. They also impose that the flow emanating from node 0 and the flow entering in node w are both equal to z, ensuring that no item is packed over the strip. Constraints (15) impose that, for each item j, either the original item or its rotated version is packed.

An illustrative example is provided in Figures 2 – 4. Considering the SCP instance I given by items having sizes $w_1 = 5, w_2 = 3, w_3 = 2, h_1 = 1, h_2 = 3, h_3 = 2$ and a strip of width w = 5, Figure 2 represents the set of arcs generated to solve the CBP problem. Figure 3 shows the arcs selected in the optimal solution. Figure 4 provides a graphical representation of the solution: the x'_{jp} values are

- $x'_{0,2}$ = 1, i.e., item 2 has its bottom left corner packed in column 0, occupies columns $0, 1, 2$, and takes 3 height units in each column.
- $x'_{3,3}$ = 1, i.e., item 3 has its bottom left corner packed in column 3, occupies columns $3, 4$, and takes 2 height units in each column.
- $x'_{0,4}$ = 1, i.e., item 4 (the rotated version of item 1) has its bottom left corner packed in column 0, occupies columns $0, 1, 2, 3, 4$, and takes 1 height unit in each column.
- $x'_{3,loss}$ = $x'_{4,loss}$ = 1, i.e., loss arcs take 1 height unit in columns 3 and 4 to satisfy the flow conservation constraints.

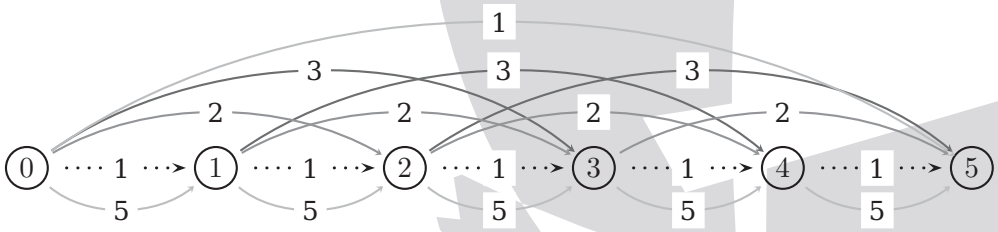

Figure 2: Set of arcs generated to solve the CBP instance I

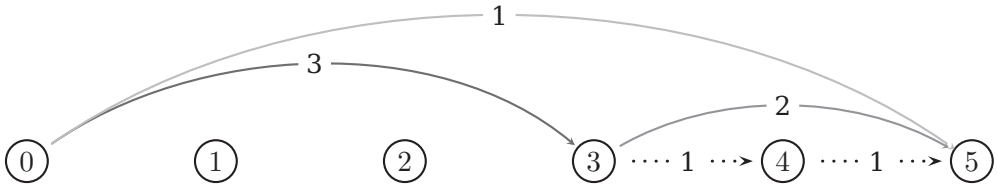

Figure 3: Set of arcs selected in the optimal solution of the CBP instance I

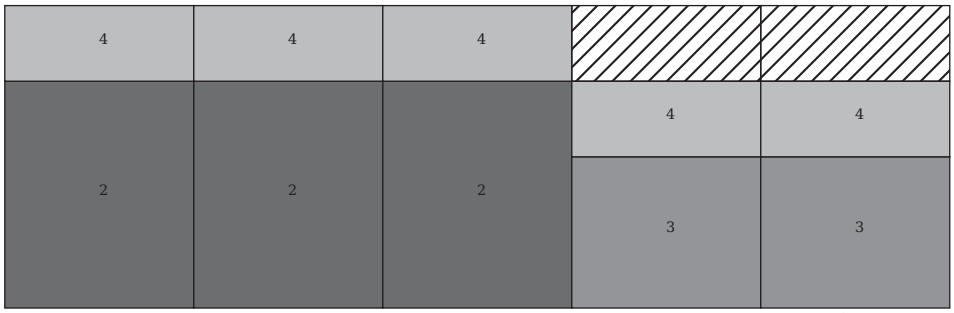

Figure 4: Graphical representation of the optimal solution of the CBP instance I

To take into account the fact that a solution may be forbidden by the slave problem, the following constraints may be added to the above model:

$$\sum_{(j,p)\in C^s} x'_{jp} \leq |C^s| - 1 \quad \forall C^s \text{ infeasible for the slave problem,} \tag{17}$$

where s denotes a problem solution and C^s denotes a set of arcs that leads to infeasibility in the slave problem when taken. Following the definition by Codato and Fischetti [10], constraints (17) are known as Combinatorial Benders' cuts. Here the two main issues are to develop an algorithm to efficiently solve the slave problem, and to possibly find the smallest set C^s that causes infeasibility. Both issues are explained in the next subsection.

3.5. Slave problem and cut generation

Once the master problem has been solved, we have a subset of selected items N' (each with a given orientation) and their abscissae. The slave problem is then invoked to check if it is possible to find the ordinates for each of the selected items such that all items are packed without exceeding the height of the strip and without overlapping with each other. The problem is known to be NP-hard (Côté, Dell'Amico and Iori [11]). To solve this problem, we use a constraint programming approach, whose logical model is as follows:

$$y_j + h_j \leq \bar{z} \quad j \in N', \tag{18}$$

$$\text{nonoverlap}\left\{[y_j, y_j + h_j], j \in N' : \sum_{p\in W(j,q)} \bar{x}'_{jp} = 1\right\} \quad q \in W, \tag{19}$$

$$y_j \geq 0. \quad j \in N'.*, \tag{20}$$

where $\bar{x}$ denote the current solution and $\bar{z}$ its value (height).

Constraints (18) impose that no item goes over the height of the strip, while constraints (19) impose that no two items overlap.

At this point, if the slave problem returns a feasible solution, then the instance has been optimally solved. If instead it proves that no feasible solution exists, then we generate a set of combinatorial Benders' cuts through the improved and lifted Benders' cuts of [11].

4. Preliminary computational experiments

To test our algorithm, we used instances from the literature and compared our results with those in Kenmochi, Imamichi, Nonobe, Yagiura, and Nagamochi [16] and Arahori, Imamichi, and Nagamochi [2]. The instances used are

- *NGCUT*, a set of 12 instances proposed by Beasly in [4];
- *GCUT*, a set of 13 instances proposed by Beasley in [3];
- *CGCUT*, a set of 3 instances proposed by Christofides and Whitlock in [9];
- *HT*, the 9 first instances out of 21 proposed by Hopper and Turton in [14].

The experiments were made on an Intel Xeon 3.10 GHz with 8 GB RAM, equipped with four cores. All our experiments were performed with a single core (the number of threads was set to one) and the time limit was set to 3600 seconds. The solver used was CPLEX 12.6.0 for the entire algorithm. The logical constraint used in the slave problem was "IloNoOverlap". For comparison, the experiments of [16] were made on a Pentium 4 (3.0GHz) and [2] used a Xeon X5260 (3.3GHz). Both used a time limit of 3600 seconds.

We start by calling our implementation of [16] for 10 seconds and switch to the algorithm presented above if the optimal solution is not found.

As shown in Table 1, the hybridization of our method with [16] gives pretty good results: we can solve 30 instances out of the 37 tried (instances solved to optimality appear in bold), and we obtain an optimal solution of value 1118 for the instance "gcut02". To the best of our knowledge, this is the first time that an algorithm provides a proven optimal solution for this instance, at least for the case with rotation, and this makes these preliminary results quite encouraging. The solution obtained is depicted in Figure 5. We can notice that the structure of the packing is very complex and cannot be obtained by guillotine cuts (a guillotine cut is a cut that goes from one side of the bin to the other).

5. Conclusion

We presented a Benders' decomposition approach for the orthogonal Stock Cutting Problem that results in a 3-phase algorithm: (i) we solve a continuous bin packing problem for which we introduce an ILP model based on the ARCFLOW formulation; (ii) we check, through constraint programming, if the solution obtained is feasible for the whole problem; (iii) we generate cuts if it is not. The algorithm presents encouraging results, solving for the first time "gcut02" in the case where rotation is allowed. When it is hybridized with a powerful branch-and-band algorithm, it can provide competitive results for most of the considered instances. The main drawback of our method is that the ILP of the master problem can be quite heavy, especially when the number of items increases. Our next direction of research is to focus on the master

Table 1: Preliminary results and comparison with state of the art SCP algorithms

Name	Our approach			[16]			[2]		
	Time	LB	UB	Time	LB	UB	Time	LB	UB
NGCUT01	**0.11**	**20**	**20**	**0.14**	**20**	**20**	**0.01**	**20**	**20**
NGCUT02	**0.02**	**28**	**28**	**0.14**	**28**	**28**	**0.01**	**28**	**28**
NGCUT03	**0.02**	**28**	**28**	**0.11**	**28**	**28**	**0.00**	**28**	**28**
NGCUT04	**0.02**	**18**	**18**	**0.15**	**18**	**18**	**0.00**	**18**	**18**
NGCUT05	**0.02**	**36**	**36**	**0.08**	**36**	**36**	**0.00**	**36**	**36**
NGCUT06	**0.03**	**29**	**29**	**0.06**	**29**	**29**	**0.01**	**29**	**29**
NGCUT07	**0.04**	**10**	**10**	**0.13**	**10**	**10**	**0.00**	**10**	**10**
NGCUT08	**4.30**	**33**	**33**	**8.80**	**33**	**33**	**0.52**	**33**	**33**
NGCUT09	**0.06**	**49**	**49**	**0.10**	**49**	**49**	**0.01**	**49**	**49**
NGCUT10	**1.02**	**59**	**59**	**2.28**	**59**	**59**	**0.04**	**59**	**59**
NGCUT11	**224.24**	**51**	**51**	3600	51	-	**3.94**	**51**	**51**
NGCUT12	**1.60**	**77**	**77**	**12.66**	**77**	**77**	**0.18**	**77**	**77**
GCUT01	**11.84**	**696**	**696**	**-**	**-**	**-**	**0.80**	**696**	**696**
GCUT02	**294.09**	**1118**	**1118**	**-**	**-**	**-**	3600	1099	-
GCUT03	3600	1644	1693	**-**	**-**	**-**	3600	1631	-
GCUT04	3600	2929	3054	**-**	**-**	**-**	3600	2926	-
GCUT05	**13.05**	**1148**	**1148**	**-**	**-**	**-**	**-**	**-**	**-**
GCUT06	**236.53**	**2503**	**2503**	**-**	**-**	**-**	**-**	**-**	**-**
GCUT07	**793.02**	**4068**	**4068**	**-**	**-**	**-**	**-**	**-**	**-**
GCUT08	3600	5624	5868	**-**	**-**	**-**	**-**	**-**	**-**
GCUT09	**11.72**	**2076**	**2076**	**-**	**-**	**-**	**-**	**-**	**-**
GCUT10	**80.23**	**5462**	**5462**	**-**	**-**	**-**	**-**	**-**	**-**
GCUT11	3600	6606	6914	**-**	**-**	**-**	**-**	**-**	**-**
GCUT12	3600	12568	13556	**-**	**-**	**-**	**-**	**-**	**-**
GCUT13	3600	4772	5240	**-**	**-**	**-**	**-**	**-**	**-**
CGCUT01	**0.04**	**23**	**23**	**0.10**	**23**	**23**	**0.00**	**23**	**23**
CGCUT02	**0.31**	**63**	**63**	**0.75**	**63**	**63**	**0.01**	**63**	**63**
CGCUT03	3600	636	652	3600	636	-	3600	636	-
HT01	**0.09**	**20**	**20**	**0.07**	**20**	**20**	**0.00**	**20**	**20**
HT02	**0.05**	**20**	**20**	**0.10**	**20**	**20**	**0.00**	**20**	**20**
HT03	**0.06**	**20**	**20**	**0.05**	**20**	**20**	**0.01**	**20**	**20**
HT04	**0.25**	**15**	**15**	**0.12**	**15**	**15**	**0.00**	**15**	**15**
HT05	**0.40**	**15**	**15**	**0.09**	**15**	**15**	**0.00**	**15**	**15**
HT06	**0.40**	**15**	**15**	**0.11**	**15**	**15**	**0.01**	**15**	**15**
HT07	**0.90**	**30**	**30**	**0.13**	**30**	**30**	**0.10**	**30**	**30**
HT08	**0.91**	**30**	**30**	**0.14**	**30**	**30**	**0.08**	**30**	**30**
HT09	**0.91**	**30**	**30**	**0.10**	**30**	**30**	**0.01**	**30**	**30**

problem trying to reduce the set of arcs used in the ILP. It could also be interesting to produce a good starting metaheuristic algorithm. As our lower bounds appear to be of very good quality, finding good feasible solutions could largely decrease the overall computational effort.

Acknowledgements

The authors would like to acknowledge financial support from Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), under grant PVE no. A007/2013.

References

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

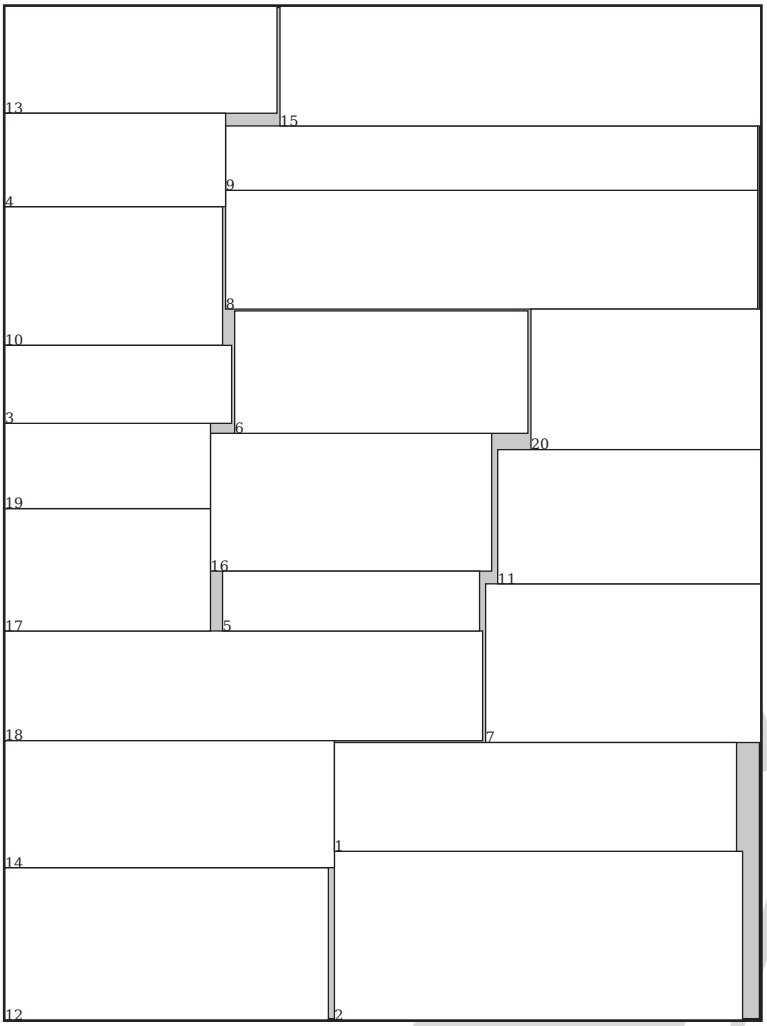

Figure 5: Optimal solution found for GCUT02 ($W = 250$, $z_{opt} = 1118$)

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

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

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

[5] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4:238–252, 1962.

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

[7] E. K. Burke, G. Kendall, and G. Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 4:655–671, 2004.

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

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

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

[11] J.F. Côté, M. Dell'Amico, and M. Iori. Combinatorial Benders' cuts for the strip packing problem. *Operations Research*, 62:643–661, 2014.

[12] J. De Armas, C. Leon, G. Miranda, and C. Segura. Optimisation of a multi-objective two-dimensional strip packing problem based on evolutionary algorithms. *International Journal of Production Research*, 48(7):2011–2028, 2010.

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

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

[15] M. Iori, S. Martello, and M. Monaci. Metaheuristic algorithms for the strip packing problem. In P. Pardalos and V. Korotkich, editors, *Optimization and Industry: New Frontiers*, pages 159–179. Kluwer, Boston, 2003.

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

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

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

[19] J.M. Valério de Carvalho. Exact solution of bin packing problems using column generation and branch and bound. *Annals of Operations Research*, 86(0):629–659, 1999.

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

[21] J. Westerlund, L. G. Papageorgiou, and T. Westerlund. A MILP model for n-dimensional allocation. *Computers & Chemical Engineering*, 31(12):1702–1714, 2007.