PackingSolver: a solver for Packing Problems

Preprint · April 2020

Luc Libralesso
Université Clermont Auvergne

Submitted to *Operations Research*
manuscript 1

PackingSolver: a tree search-based solver for two-dimensional two- and three-staged guillotine packing problems

Florian Fontan
dev@florian-fontan.fr

Luc Libralesso
Univ. Grenoble Alpes, CNRS, Grenoble INP*, G-SCOP, 38000 Grenoble, France, luc.libralesso@grenoble-inp.fr

In this article, we introduce PackingSolver, a new solver for two-dimensional two- and three-staged guillotine Packing Problems. It relies on a simple yet powerful anytime tree search algorithm called Memory Bounded A* (MBA*). This algorithm was first introduced in libralesso2020 for the 2018 ROADEF/EURO challenge glass cutting problem[†], for which it had been ranked first during the final phase. In this article, we generalize it for a large variety of Cutting and Packing problems. The resulting program can tackle two-dimensional Bin Packing, Multiple Knapsack, and Strip Packing Problems, with two- or three-staged exact or non-exact guillotine cuts, the orientation of the first cut being imposed or not, and with or without item rotation. Despite its simplicity and genericity, computational experiments show that this approach is competitive compared to the other dedicated algorithms from the literature. It even returns state-of-the-art solutions on several variants. The combination of efficiency, ability to provide good solutions fast, simplicity and versatility makes it particularly suited for industrial applications, which require quickly developing algorithms implementing several business-specific constraints.

Key words: two-dimensional guillotine packing, bin packing, knapsack, strip packing, anytime algorithm, tree search algorithm

[*] Institute of Engineering Univ. Grenoble Alpes

[†] `https://www.roadef.org/challenge/2018/en/index.php`

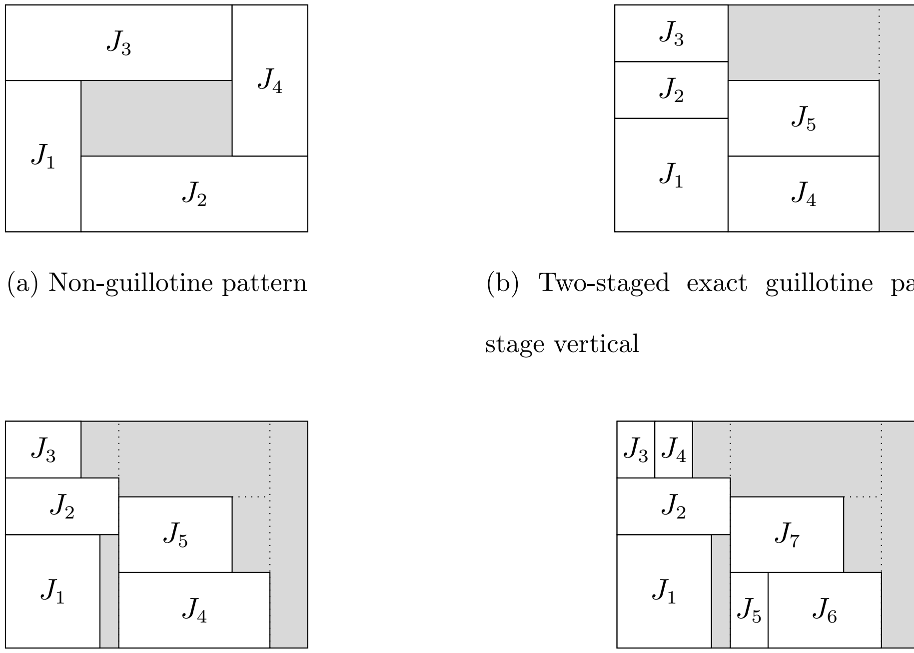
Solving optimization problems in practice often involves short development times and changing objectives and constraints. Hence the need to develop generic tools that require limited adaptations to integrate business-specific constraints. To this extent, Mixed Integer (Non-)Linear Programming solvers, Constraint Programming solvers or other optimization solvers are now of great help for the OR practitioner (Byrd et al. 2006, Bixby and Rothberg 2007, Benoist et al. 2011). However, due to their geometrical constraints, Cutting and Packing Problems remain difficult to solve and quickly developing efficient algorithms for these problems is still challenging. We aim at filling this gap. In this article, we introduce a new solver dedicated to Cutting and Packing Problems. So far, it focuses on two-dimensional two- and three-staged guillotine variants (other Cutting and Packing variants should be added in future releases), and it already returns high-quality solutions on more than 15 variants studied in the literature. Moreover, it provides an implementation for many other variants that have not been studied yet.

1. Introduction

We consider two-dimensional guillotine Packing Problems: one has to pack rectangles of various sizes into larger bins while only edge-to-edge cuts are allowed. In a solution, guillotine cuts can be partitioned into stages, *i.e.* series of parallel cuts, and it is common to limit the number of allowed stages. Here, we restrict to two- or three-staged guillotine patterns. In both cases, we consider both exact and non-exact variants. In the non-exact variant, an additional cut is allowed to separate items from waste. Figure 1 illustrates the different pattern types.

We consider the three main objectives studied in the literature: Bin Packing, Knapsack and Strip Packing. In Bin Packing and Strip Packing Problems, all items need to be produced. In Bin Packing Problems, the number of used bins is minimized, while in Strip Packing Problems, there is only one container with one infinite dimension and the objective is to minimize the length used in this dimension. In Knapsack Problems, the number of containers is limited, every item has a profit and the total profit of the packed items is maximized.

Finally, for each variant, we consider the oriented case where item rotation is not allowed and non-oriented case where it is.



(a) Non-guillotine pattern

(b) Two-staged exact guillotine pattern, first stage vertical

(c) Two-staged non-exact guillotine pattern, first stage vertical

(d) Three-staged exact guillotine pattern, first stage vertical

Figure 1 Pattern type examples

Throughout the article, the different variants are named following our notations illustrated with the following examples:

- BPP-O: (non-guillotine) Bin Packing Problem, Oriented
- G-BPP-R: Guillotine cuts, Bin Packing Problem, Rotation
- 2G-KP-O: 2-staged exact guillotine cuts, first cut horizontal or vertical, Knapsack Problem, Oriented
- 3NEGH-SPP-O: 3-staged non-exact guillotine cuts, first cut horizontal, Strip Packing Problem, Oriented

We also use the following vocabulary: a k-cut is a cut performed in the k-th stage. Cuts separate bins into k-th level sub-plates. For example, 1-cuts separate the bin in several first level sub-plates. S denotes a solution or a node in the search tree (a partial solution).

The following definitions are given for the case where the first cut in the last bin is vertical, but naturally, adapt to the case where it is horizontal. We call the last first level sub-plate, the rightmost

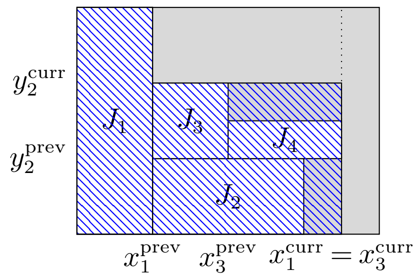


Figure 2 Last bin of a solution which does not contain all items. The area is the whole hatched part and the waste in the grey hatched part.

one containing an item; the last second level sub-plate, the topmost one containing an item in the last first level sub-plate; and the last third level sub-plate the rightmost one containing an item in the last second level sub-plate. $x_1^{\text{prev}}(S)$ and $x_1^{\text{curr}}(S)$ are the left and right coordinates of the last first level sub-plate; $y_2^{\text{prev}}(S)$ and $y_2^{\text{curr}}(S)$ are the bottom and top coordinates of the last second level sub-plate; and $x_3^{\text{prev}}(S)$ and $x_3^{\text{curr}}(S)$ are the left and right coordinates of the last third level sub-plate. Figure 2 presents a usage example of these definitions. We define the area and the waste of a solution S as follows:

$$\text{area}(S) = \begin{cases} A + x_1^{\text{curr}}(S)h & \text{if } S \text{ contains all items} \\ A + x_1^{\text{prev}}(S)h & \\ \quad + (x_1^{\text{curr}}(S) - x_1^{\text{prev}}(S))y_2^{\text{prev}}(S) & \\ \quad + (x_3^{\text{curr}}(S) - x_1^{\text{prev}}(S))(y_2^{\text{curr}}(S) - y_2^{\text{prev}}(S)) & \text{otherwise} \end{cases}$$

$$\text{waste}(S) = \text{area}(S) - \text{item\_area}(S)$$

with A the sum of the areas of all but the last bin, h the height of the last bin and item_area(S) the sum of the area of the items of S. Area and waste are illustrated in Figure 2.

2. Literature review

Two-dimensional guillotine Packing Problems have been introduced by Gilmore and Gomory (1965) and have received a lot of attention since. Researchers usually focus on one specific variant or only on a few ones.

Algorithms are sometimes adapted for both the oriented and the non-oriented cases. Velasco and Uchoa (2019) developed a heuristic for G-KP-O and G-KP-R, Wei et al. (2014) for G-SPP-O and G-SPP-R, Charalambous and Fleszar (2011), Fleszar (2013) and Cui et al. (2018) for G-BPP-O and G-BPP-R, Lodi and Monaci (2003) an exact algorithm for 2NEGH-KP-O and 2NEGH-KP-R.

Some methods have been designed to work on more variants. do Nascimento et al. (2019) developed an exact algorithm for G-KP-O, 3NEGH-KP-O, 2NEGH-KP-O and the three-dimensional variants, Bortfeldt and Winter (2009) developed a genetic algorithm for G-KP-O, G-KP-R, and the non-guillotine variants. Alvelos et al. (2009) and Silva et al. (2010) respectively developed a heuristic and an exact algorithm for 3NEGH-BPP-O, 3GH-BPP-O, 2NEGH-BPP-O and 2GH-BPP-O, and the non-oriented cases. Furini et al. (2016) introduced a model for G-KP-O and G-SPP-O. Lodi et al. (2004) proposed a unified tabu search for two- and three-dimensional Packing Problems. They provide computational experiments for BPP-O and the three-dimensional variant. They also describe how to adapt the algorithm for several variants such as Strip Packing or Multiple Knapsack. However, adapting the algorithm requires to provide a heuristic procedure, on which the efficiency of the algorithm highly relies. We did not find any use of their tabu search in the subsequent literature. Also, a framework has been proposed by Nepomuceno et al. (2008); unfortunately, it has only been implemented for BPP-O and we did not find any use of their framework in the subsequent literature either.

Regarding our methodology, even though tree search algorithms have been widely used to solve Packing Problems, the search algorithm that we implemented does not seem to have been proposed before. However, we may notice that many packing algorithms rely on Beam Search which is relatively close, as discussed in Section 5. Akeb et al. (2009), Hifi and M'Hallah (2009), Akeb et al. (2010) and Akeb et al. (2011) implemented it for Circular Packing Problems; Bennell and Song (2010) and Bennell et al. (2018) for Irregular Packing Problems; Wang et al. (2013), Araya and Riff (2014) and Araya et al. (2020) for three-dimensional Packing Problems; and Hifi et al. (2012) for 2NEGH-KP-O.

3. Algorithm description

We propose an anytime tree search algorithm.

Anytime is a terminology usually found in automated planning and scheduling (AI planning) communities. It means that the algorithm can be stopped at any time and still provides good solutions. In other words, it produces feasible solutions quickly and improves them over time (as classical meta-heuristics do).

Tree search algorithms represent the solution space as an implicit tree called "branching scheme" and explore it completely in the case of exact methods or partially in the case of heuristic methods. The branching scheme is described in Section 3.1 and the tree search algorithm in Section 3.2.

3.1. Branching scheme

We describe the branching scheme for the 3-staged cases with vertical cuts in the first stage. For the 2-staged cases, we merely impose the position of the first cut to be at the end of the bin and adjust the computation of parameters accordingly; and when the cuts in the first stage are horizontal, we simply adapt the computation of coordinates.

The branching scheme is rather straightforward. The root node is the empty solution without any items, and at each stage, a new item is added. All items that do not belong to the current node are considered. However, items in a solution are inserted according to the following order: rightmost first level sub-plates first; within a first level sub-plate, bottommost second level sub-plates first; and within a second level sub-plate, rightmost items first. Thus, a new item can be inserted in a new bin; in a new first level sub-plate to the right of the current one; in a new second-level sub-plate above the current one; in a new third-level sub-plate, to the right of the last added item. If the cuts of the first stage can be vertical or horizontal, then two different insertions in a new bin are considered: an insertion in a new bin with vertical cuts in the first stage, and an insertion in a new bin with horizontal cuts in the first stage.

To handle exact guillotine cuts, we simply fix the position of the 2-cut above an item inserted in a new bin, first or second level sub-plate, *i.e.* the next items inserted in the same second level sub-plate will only be those of the same height.

(a) (b)

Figure 3 Solution (a) dominates solution (b) because the hatched area will not be used

Item rotation or not is naturally handled in the branching scheme.

To reduce the size of the tree, we apply some simple dominance rules.

First, if an item can be inserted in the current bin, we do not consider insertions in a new bin; and if an item can be inserted in the current first (resp. second) level sub-plate without increasing the position of its left 1-cut (resp. top 2-cut), we do not consider insertions in a new first (resp. second) level sub-plate.

Then, if item rotation is allowed, some insertions can be discarded as illustrated in Figure 3.

We also impose an order on identical items.

Finally, we add the following symmetry breaking strategy: a k-level sub-plate is forbidden to contain an item with a smaller index than the previous k level sub-plate of the same $(k-1)$-level sub-plate. The symmetry breaking strategy is controlled with a parameter s, $1 \leq s \leq 4$. If $s = k$, then the symmetry breaking strategy is only used with k' level sub-plates, $k' \geq k$. For example, if $s = 4$, no symmetry breaking strategy is used. The choice of the value of s is discussed in Section 5.

3.2. Tree search algorithm

The tree described in the previous section is too large to be entirely explored. Therefore, we use a tree search algorithm that we called Memory Bounded A* (MBA*) to explore the most interesting parts in priority. The pseudo-code is given in Algorithm 1. MBA* starts with a queue containing only the root node. At each iteration, the "best" node is extracted from the queue and its children are added to the queue. If the size of the queue goes over a pre-defined threshold value, the "worst" nodes are discarded. We start with a threshold of 2, and each time the queue becomes empty, we start over with a threshold multiplied by the growth factor f. We choose $f = 1.5$ as discussed in Section 5.

Algorithm 1 Memory Bounded A* (MBA*)

```
queue ← {root}
while |queue| ≠ ∅ and time < timelimit do
    n ← extractBest(queue)
    queue ← queue \ {n}
    for all v ∈ children(n) do
        queue ← queue ∪ {v}
    while |queue| > D do
        n ← extractWorst(queue)
        queue ← queue \ {n}
```

The function used to define “better” and “worse” is called a guide. The lower the value of the guide function is, the better the solution. For Bin Packing and Strip Packing Problems, we designed the following guide functions:

$$c_0(S) = \text{waste\_percentage}(S)$$

$$c_1(S) = \frac{\text{waste\_percentage}(S)}{\text{mean\_item\_area}(S)}$$

$$c_2(S) = \frac{0.1 + \text{waste\_percentage}(S)}{\text{mean\_item\_area}(S)}$$

$$c_3(S) = \frac{0.1 + \text{waste\_percentage}(S)}{\text{mean\_squared\_item\_area}(S)}$$

with

- waste_percentage(S) = waste(S)/area(S);
- mean_item_area(S) the mean area of the items of S;
- mean_squared_item_area(S) the mean squared area of the items of S.

For Knapsack Problems, we use the following guide function:

$$c_4(S) = \frac{\text{area}(S)}{\text{profit}(S)}$$

with profit(S) the sum of profit of the items of S.

The importance and design of these guide functions are discussed in Section 5.

4. Computational experiments

PackingSolver is implemented in C++. The code is available online[1]. The repository also contains all the scripts used to conduct the experiments so that results can be reproduced. The results presented above have been obtained with PackingSolver 0.2[2] running on a personal computer with an Intel Core i5-8500 CPU @ 3.00GHz × 6. We allow running up to 3 threads with different settings in parallel. The settings have been chosen following the observations given in Section 5. Better settings may exist, we try to reproduce the results one would obtain in a practical situation where the global characteristics of the instances are known.

We compare the performances of our algorithm with the best algorithms from the literature for each variant. Due to a large number of problems, we only provide a synthesis of the results here. However, detailed results are available online[3] and the interested reader is encouraged to have a look at them.

Results are summarized in Tables 1, 2 and 3. The first column of the tables indicates the article from which the results have been extracted or the parameters we used for our algorithm. c_a^b indicates a thread with guide function c_a and symmetry breaking parameter b. TL stands for "time limit". The time limit has been chosen to yield a good compromise between computation time and the best solution value. We only indicate the frequencies of the processors used to evaluate the other algorithms when they significantly differ from ours, *i.e.* below 2GHz.

[1] `https://github.com/fontanf/packingsolver`

[2] `https://github.com/fontanf/packingsolver/releases/tag/0.2`

[3] `https://github.com/fontanf/packingsolver/blob/0.2/results_rectangleguillotine.ods`

For Bin Packing Problems, the second column contains the total number of bins used in Table 1a and the average of the average percentage of waste of each sub-dataset in Table 1b. For Knapsack and Strip Packing Problems, it contains the average gap to the best-known solutions. The third one indicates the average time to best when available, or the average computation time.

Dataset "hifi" is a dataset composed of instances from Christofides and Whitlock (1977), Wang (1983), Oliveira and Ferreira (1990), Tschöke and Holthöfer (1995), Fekete and Schepers (1997), Fayard et al. (1998), Hifi (1997) and Cung et al. (2000). Researchers usually test their algorithms on a subset of these instances, but often not the same. Dataset "bwmv" refers to datasets from Berkey and Wang (1987) and Martello and Vigo (1998) which are usually used together.

Other datasets are

- "beasley1985" from Beasley (1985)
- "fayard1998" from Fayard et al. (1998)
- "kroger1995" from Kröger (1995)
- "hopper2000" from Hopper (2000)
- "hopper2001" from Hopper and Turton (2001)
- "alvarez2002" from Alvarez-Valdés et al. (2002)
- "morabito2010" from Morabito and Pureza (2010)
- "hifi2012" from Hifi et al. (2012)
- "velasco2019" from Velasco and Uchoa (2019)

4.1. Bin Packing Problems

Results for Bin Packing Problems are summarized in Table 1. On 2NEGH-BPP-O and 2NEGH-BPP-R, PackingSolver respectively needs fewer bins than the algorithms from Cui and Zhao (2013) and Cui et al. (2016) for the considered datasets. Furthermore, the average time to best is of the order of a second, which is significantly smaller than the average time reported for the other algorithms. On 3NEGH-BPP-O, 3GH-BPP-O, and 2NEGH-BPP-O, the average of the average percentage of waste of PackingSolver is smaller than the one of the algorithms from Alvelos et al. (2009). However, on

2GH-BPP-O, it is greater. Finally, compared to the algorithms from Puchinger and Raidl (2007) and Alvelos et al. (2014), it needs more bins, but the average time to best is two orders of magnitude smaller than the average time reported for those algorithms. We also note that PackingSolver respectively needs significantly fewer bins on 3NEGH-BPP-O and 3GH-BPP-O compared to the algorithms from Puchinger and Raidl (2007) and Alvelos et al. (2014) for 3GH-BPP-O and 2NEGH-BPP-O,

4.2. Knapsack Problems

Results for Knapsack Problems are summarized in Table 2. We include comparisons with algorithms designed for the non-staged variants. In these cases, PackingSolver usually fails to find the best solutions. It seems likely that they often cannot be reached with only 3 stages. However, its average gap to best is generally less than 1% and on datasets “velasco2019” it is even better than the recent algorithm from Velasco and Uchoa (2019). The same happens on dataset “fayard1998” for G-KP-R, but the algorithm developed by Bortfeldt and Winter (2009) seems to perform significantly worse than more recent algorithms and none of them has been tested on this dataset.

On 3NEGV-KP-O, the average gap to best of PackingSolver is better than Cui et al. (2015), but at the expense of longer computation times. For 2NEGH-KP-O, as Alvarez-Valdes et al. (2007), it finds all the best solutions, but faster. Compared to the algorithm from Hifi et al. (2008), it performs slightly worse on dataset “alvarez2002” (even if the average gap is 0.0, it fails to find the best solution on two instances) but better on dataset “hifi2012”.

On variants 2NEG-KP-R, 2G-KP-O, 2GH-KP-O, and 2GV-KP-O for which Lodi and Monaci (2003) and Hifi and Roucairol (2001) developed exact algorithms, PackingSolver finds all optimal solutions in reasonable computation times.

Note that, to the best of our knowledge, only Cui et al. (2008) proposed an algorithm for a variant of a Multiple Knapsack Problem. However, they consider homogenous T-shaped patterns which we do not consider in this article.

Article / Parameters	Total	Time (s)
3NEGH-BPP-O, "bwmv"		
PS, $c_0^2c_2^2c_3^3$, TL 60s	7278	0.790
3GH-BPP-O, "bwmv"		
Puchinger and Raidl (2007)	7325	160.68
PS, $c_0^2c_2^2c_3^3$, TL 60s	7344	0.808
2NEGH-BPP-O, "bwmv"		
Alvelos et al. (2014)	7372	29.42
Alvelos et al. (2014)	7364	84.04
PS, $c_0^2c_2^2c_3^3$, TL 60s	7391	0.814
2NEGH-BPP-O, "hifi"		
Cui and Zhao (2013)	260	0.19
PS, $c_2^3c_3^3c_3^4$, TL 10s	255	0.106
2NEGH-BPP-O, "alvarez2002"		
Cui and Zhao (2013)	219	9.5
PS, $c_2^3c_3^3c_3^4$, TL 10s	218	0.346
2NEGH-BPP-R, "bwmv"		
Cui et al. (2016)	7034	20.72
PS, $c_0^2c_2^2c_3^3$, TL 60s	7029	0.590

(a)

Article / Parameters	Waste	Time (s)
3NEGH-BPP-O, "bwmv"		
Alvelos et al. (2009)	26.52	
PS, $c_0^2c_2^2c_3^3$, TL 60s	20.93	0.790
3GH-BPP-O, "bwmv"		
Alvelos et al. (2009)	26.29	
PS, $c_0^2c_2^2c_3^3$, TL 60s	22.34	0.808
2NEGH-BPP-O, "bwmv"		
Alvelos et al. (2009)	26.12	
PS, $c_0^2c_2^2c_3^3$, TL 60s	23.21	0.807
2GH-BPP-O, "bwmv"		
Alvelos et al. (2009)	49.06	
PS, $c_0^3c_2^3c_3^4$, TL 60s	49.45	0.181

(b)

Table 1 Results on Bin Packing Problems

4.3. Strip Packing Problems

Not many variants of guillotine Strip Packing Problems have been studied in the literature; only G-SPP-O, G-SPP-R, and 2NEGH-SPP-O. This makes comparisons with PackingSolver difficult since it is limited to three-staged patterns, and 2NEGH-SPP-O has several specific structural properties that dedicated algorithms can exploit, but not a more generic one. We still provide computational

Article / Parameters	Gap	Time (s)
G-KP-O, "fayard1998"		
Velasco and Uchoa (2019)	0.00	0.06
PS, 3NEG-KP-O, $c_4^2c_4^3$, TL 10s	0.16	0.182
G-KP-O, "alvarez2002"		
Wei and Lim (2015)	0.02	21.987
Velasco and Uchoa (2019)	0.00	93.681
PS, 3NEG-KP-O, $c_4^2c_4^3$, TL 60s	0.48	13.264
G-KP-O, "hopper2001"		
Wei and Lim (2015)	0.31	22.214
PS, 3NEG-KP-O, $c_4^2c_4^3$, TL 10s	4.69	1.283
G-KP-O, "morabito2010"		
Velasco and Uchoa (2019)	0.01	19.57
PS, 3NEG-KP-O, $c_4^2c_4^3$, TL 10s	0.17	0.332
G-KP-O, "beasley1985"		
Dolatabadi et al. (2012)	0.00	1397.738
Wei and Lim (2015)	0.44	20.923
PS, 3NEG-KP-O, $c_4^2c_4^3$, TL 10s	0.56	0.204
G-KP-O, "velasco2019"		
Velasco and Uchoa (2019)	1.42	165.618
PS, 3NEG-KP-O, $c_4^2c_4^3$, TL 120s	0.47	34.682
G-KP-R, "hopper2001"		
Wei and Lim (2015)	0.00	5.04
PS, 3NEG-KP-R, $c_4^2c_4^3$, TL 30s	1.71	8.049
G-KP-R, "fayard1998"		
Bortfeldt and Winter (2009)	1.57	
PS, 3NEG-KP-R, $c_4^2c_4^3$, TL 30s	0.00	2.578
G-KP-R, "velasco2019"		
Velasco and Uchoa (2019)	1.05	170.20
PS, 3NEG-KP-R, $c_4^2c_4^3$, TL 120s	0.51	38.590
3NEGV-KP-O, "alvarez2002"		
Cui et al. (2015)	0.09	2.06
PS, $c_4^1c_4^2c_4^3$, TL 60s	0.01	11.879

Article / Parameters	Gap	Time (s)
2NEGH-KP-O, "hifi"		
Alvarez-Valdes et al. (2007)	0.00	0.5
PS, $c_4^2c_4^3$, TL 3s	0.00	0.032
2NEGH-KP-O, "alvarez2002"		
Hifi et al. (2008)	0.00	0.2
PS, $c_4^2c_4^3$, TL 10s	0.00	0.410
2NEGV-KP-O, "alvarez2002"		
Hifi et al. (2008)	0.00	0.2
PS, $c_4^2c_4^3$, TL 10s	0.00	0.382
2NEGH-KP-O, "hifi2012"		
Hifi et al. (2008)	0.26	368.365
PS, $c_4^2c_4^3$, TL 300s	0.12	138.742
2NEGV-KP-O, "hifi2012"		
Hifi et al. (2008)	0.24	310.105
PS, $c_4^2c_4^3$, TL 300s	0.00	121.014
2NEGH-KP-R, "hifi"		
Lodi and Monaci (2003) (533 MHz)	0.00	34.348
PS, $c_4^2c_4^3$, TL 3s	0.00	0.161
2G-KP-O, "hifi"		
Hifi and Roucairol (2001) (250 Mhz)	0.00	1.253
PS, $c_4^2c_4^3$, TL 1s	0.00	0.003
2GH-KP-O, "hifi"		
Hifi and Roucairol (2001) (250 Mhz)	0.00	1.145
PS, $c_4^2c_4^3$, TL 1s	0.00	0.002
2GV-KP-O, "hifi"		
Hifi and Roucairol (2001) (250 Mhz)	0.00	1.147
PS, $c_4^2c_4^3$, TL 1s	0.00	0.005

Table 2 Results on Knapsack Problems

experiments for these variants in Table 3. As expected, PackingSolver does not perform as well. Still, on dataset "bwmv", it returns strictly better average solutions on 16 out of 50 groups of instances for G-SPP-O and on 14 out of 50 groups of instances for G-SPP-R than the algorithm from Wei et al. (2014). To highlight a bit more the contribution of PackingSolver for Strip Packing Problems, we provide a comparison of the solutions from Lodi et al. (2004) and from Cui et al. (2017) for 2NEGH-SPP-O with the solutions returned by PackingSolver for 2NEGH-SPP-R, *i.e.* when item rotation is allowed. The average solutions returned by PackingSolver are strictly better on each of the 50 groups of instances of dataset "bwmv".

Article / Parameters	Gap	Time (s)
G-SPP-O, "kroger1995"		
Wei et al. (2014)	0.27	22.67
PS, 3NEGH-SPP-O, $c_0^2c_0^3c_0^4$, TL 30s	3.65	10.416
G-SPP-O, "hopper2001"		
Wei et al. (2014)	0.00	6.267
PS, 3NEGH-SPP-O, $c_0^2c_0^3c_0^4$, TL 30s	6.75	4.364
G-SPP-O, "hopper2000"		
Wei et al. (2014)	0.00	20.647
PS, 3NEGH-SPP-O, $c_0^2c_0^3c_0^4$, TL 30s	8.72	5.899
G-SPP-O, "bwmv"		
Wei et al. (2014)	0.15	17.736
PS, 3NEGH-SPP-O, $c_0^2c_5^2c_6^3$, TL 60s	1.10	12.831
G-SPP-R, "kroger1995"		
Cui et al. (2013)	0.00	56
PS, 3NEGH-SPP-R, $c_0^2c_0^3c_0^4$, TL 30s	1.84	9.716
G-SPP-R, "hopper2001"		
Wei et al. (2014)	0.00	13.466
3NEGH-SPP-R, $c_0^2c_0^3c_0^4$, TL 30s	3.00	4.153
G-SPP-R, "hopper2000"		
Wei et al. (2014)	0.00	13.465
PS, 3NEGH-SPP-R, $c_0^2c_0^3c_0^4$, TL 30s	3.30	10.7
G-SPP-R, "bwmv"		
Wei et al. (2014)	0.13	18.253
PS, 3NEGH-SPP-R, $c_0^2c_5^2c_6^3$, TL 30s	0.58	12.592

Article / Parameters	Gap	Time (s)
2NEGH-SPP-O, "alvarez2002"		
Cui et al. (2013)	0.02	4.78
PS, $c_4^1c_4^2c_4^3$, TL 30s	1.13	3.726
2NEGH-SPP-O, "bwmv"		
Lodi et al. (2004)	0.02	66.71
Cui et al. (2017)	0.13	1.77
PS, $c_0^2c_5^2c_6^3$, TL 30s	0.68	0.992
2NEGH-SPP-R, "bwmv"		
Lodi et al. (2004)	7.96	66.71
Cui et al. (2017)	8.08	1.77
PS, $c_0^2c_5^2c_6^3$, TL 30s	0.00	1.773

Table 3 Results on Strip Packing Problems

5. Discussion

In this section, we discuss some items related to the algorithm.

Growth factor of the queue size threshold: In Section 3.1, we indicated that we set the growth factor of the queue size threshold to 1.5. The greater the threshold, the better the solutions will be, but the longer MBA* will take to terminate. Furthermore, for Bin Packing and Strip Packing Problems, full solutions are usually found shortly before it terminates. Therefore, by choosing a too large value for the growth factor, we take the risk to reach the time limit having to spend a lot of time with a given threshold without obtaining any solutions from it. On the other hand, if the growth factor is too small, then only small thresholds value will be explored and no good solutions will be found. In our experiments, 1.5 proved to be a good compromise.

Choice of guide functions: The effectiveness of MBA* highly relies on the definition of its guide function. For MBA*, the guide function should be relevant to compare two solutions at different stages of the tree. Therefore, the waste-percentage c_0 appears much more relevant than the waste alone for Bin Packing and Strip Packing variants. Guide function c_1 is adapted from c_0, but it favours solutions containing larger items. This helps to avoid situations where all small items are packed in the first bins and the last bins get all the large items that lead to high waste. Guide function c_2 is adapted from c_1: indeed, even if c_1 favors large items first, solutions with no waste at all will always be extracted first, even if they contain only small items. The constant in c_2 aims at fixing this behavior and will lead to better solutions on instances in which optimal solutions contain significant waste (more than 10%). c_3 is adapted from c_2 and favours even more large items first. This guide function is useful for some instances containing several very large items. Finally, c_4 is a natural adaption of c_0 for Knapsack variants. An experimental comparison of several guide functions for the 2018 ROADEF/EURO challenge glass cutting problem is presented in libralesso2020.

Depth of the symmetry breaking strategy: In exact tree search algorithms, it is usually worth breaking symmetries. However, this is not the case when the tree is not meant to be explored completely. For example, consider two symmetrical nodes, the first one normally appearing in the queue, but the second one never being added to the queue because one of its ancestors has been removed to reduce the size of the queue. If the first one is not explored because asymmetry has been detected, then this solution will not be found during the search. How to determine the ideal depth of the symmetry breaking strategy for an instance is not clear yet. The relative size of the items compared to the bin might be an influential factor. For the experiments, we chose 2 or 3 as "standard" values. For some instances containing many items (more than 1000), only a value of 4 ensures finding a feasible solution quickly; in contrast, for some knapsack instances with few first-level sub-plates, a value of 1 gives access to better solutions. An experimental evaluation of the influence of the symmetry breaking strategy for the 2018 ROADEF/EURO challenge glass cutting problem is presented in libralesso2020.

MBA vs Beam Search:* Beam Search is another popular tree search algorithm in the packing literature. Beam Search also starts with a queue containing only the root node. However, at each iteration, all nodes of the queue are expanded, and as in MBA*, if the size of the queue goes over a pre-defined threshold, the worst nodes are discarded. Thus, at each iteration, the queue always contains nodes belonging to the same level of the tree. Beam Search seems therefore effective when the guide function is relevant to compare nodes belonging to the same level. This is for example generally not the case in Branch-and-Cut trees where branching consists in fixing a variable to 0 or 1. In Packing Problems, it is easier to compare such solutions, but the guide functions we presented in Section 3 make it even possible to compare nodes at different levels of the search tree. Thus, Beam Search expands many nodes which are not that much interesting, whereas MBA* always expands only the best current node. An experimental comparison of MBA* and Beam Search for the 2018 ROADEF/EURO challenge glass cutting problem is presented in libralesso2020.

Higher staged guillotine cuts: Our branching scheme generates up to three-staged patterns. One could wonder whether it could be possible to adapt it for four-staged or non-staged guillotine patterns. However, if a similar branching scheme seems possible, it may significantly increase symmetry issues. We believe that this would be prohibitive.

Item-based vs block-based: Many researchers highlighted the benefits of using block-based approaches, *i.e.* inserting several items at each stage of the tree (Bortfeldt and Jungmann 2012, Wei et al. 2014, Lodi et al. 2017). It is interesting to note that it is not what we do in PackingSolver, yet our algorithm is competitive.

6. Conclusion and future work

We introduced a new solver for two-dimensional two- and three-staged guillotine Packing Problems, implementing a new tree search algorithm called MBA*. We showed that even on the pure Packing Problems from the literature, it is competitive compared to other algorithms, and is even able to return state-of-the-art solutions on several variants. Its performances seem to rely on two key components: a branching scheme which limits symmetry issues; and a tree search algorithm fully exploiting guide functions which make it possible to compare nodes at different levels of the search tree.

In addition to effectiveness, the choice of a tree search algorithm makes the solver attractive for problems with additional side constraints. Indeed, new constraints are likely to reduce the size of the search tree. The solver already handles some constraints previously considered in the literature such as the presence of defective areas on bins (Afsharian et al. 2014, Martin et al. 2019) or homogenous patterns (Cui 2008, Chen et al. 2016), and some not considered yet such as chain precedence constraints between items.

The current roadmap includes implementing algorithms for Cutting Stock and Variable-sized Bin Packing Problems, and a branching scheme which generates non-guillotine patterns.

References

Afsharian M, Niknejad A, Wäscher G (2014) A heuristic, dynamic programming-based approach for a two-dimensional cutting problem with defects. *OR Spectrum* 36(4):971–999, ISSN 1436-6304, URL `http://dx.doi.org/10.1007/s00291-014-0363-x`.

Akeb H, Hifi M, M'Hallah R (2010) Adaptive beam search lookahead algorithms for the circular packing problem. *International Transactions in Operational Research* 17(5):553–575, ISSN 1475-3995, URL `http://dx.doi.org/10.1111/j.1475-3995.2009.00745.x`.

Akeb H, Hifi M, M'Hallah R (2009) A beam search algorithm for the circular packing problem. *Computers & Operations Research* 36(5):1513–1528, ISSN 0305-0548, URL http://dx.doi.org/10.1016/j.cor.2008.02.003.

Akeb H, Hifi M, Negre S (2011) An augmented beam search-based algorithm for the circular open dimension problem. *Computers & Industrial Engineering* 61(2):373–381, ISSN 0360-8352, URL http://dx.doi.org/10.1016/j.cie.2011.02.009.

Alvarez-Valdes R, Martí R, Tamarit JM, Parajón A (2007) GRASP and Path Relinking for the Two-Dimensional Two-Stage Cutting-Stock Problem. *INFORMS Journal on Computing* 19(2):261–272, ISSN 1091-9856, URL http://dx.doi.org/10.1287/ijoc.1050.0169.

Alvarez-Valdés R, Parajón A, Tamarit JM (2002) A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. *Computers & Operations Research* 29(7):925–947, ISSN 0305-0548, URL http://dx.doi.org/10.1016/S0305-0548(00)00095-2.

Alvelos F, Chan TM, Vilaça P, Gomes T, Silva E, Carvalho JMVd (2009) Sequence based heuristics for two-dimensional bin packing problems. *Engineering Optimization* 41(8):773–791, ISSN 0305-215X, URL http://dx.doi.org/10.1080/03052150902835960.

Alvelos F, Silva E, de Carvalho JMV (2014) A Hybrid Heuristic Based on Column Generation for Two- and Three- Stage Bin Packing Problems. Murgante B, Misra S, Rocha AMAC, Torre C, Rocha JG, Falcão MI, Taniar D, Apduhan BO, Gervasi O, eds., *Computational Science and Its Applications – ICCSA 2014*, 211–226, Lecture Notes in Computer Science (Cham: Springer International Publishing), ISBN 978-3-319-09129-7, URL http://dx.doi.org/10.1007/978-3-319-09129-7_16.

Araya I, Moyano M, Sanchez C (2020) A beam search algorithm for the biobjective container loading problem. *European Journal of Operational Research* ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2020.03.040.

Araya I, Riff MC (2014) A beam search approach to the container loading problem. *Computers & Operations Research* 43:100–107, ISSN 0305-0548, URL http://dx.doi.org/10.1016/j.cor.2013.09.003.

Beasley JE (1985) Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. *Journal of the Operational Research Society* 36(4):297–306, ISSN 1476-9360, URL http://dx.doi.org/10.1057/jors.1985.51.

Bennell JA, Cabo M, Martínez-Sykora A (2018) A beam search approach to solve the convex irregular bin packing problem with guillotine cuts. *European Journal of Operational Research* 270(1):89–102, ISSN 0377-2217, URL `http://dx.doi.org/10.1016/j.ejor.2018.03.029`.

Bennell JA, Song X (2010) A beam search implementation for the irregular shape packing problem. *Journal of Heuristics* 16(2):167–188, ISSN 1572-9397, URL `http://dx.doi.org/10.1007/s10732-008-9095-x`.

Benoist T, Estellon B, Gardi F, Megel R, Nouioua K (2011) LocalSolver 1.x: a black-box local-search solver for 0-1 programming. *4OR* 9(3):299, ISSN 1614-2411, URL `http://dx.doi.org/10.1007/s10288-011-0165-9`.

Berkey JO, Wang PY (1987) Two-Dimensional Finite Bin-Packing Algorithms. *Journal of the Operational Research Society* 38(5):423–429, ISSN 1476-9360, URL `http://dx.doi.org/10.1057/jors.1987.70`.

Bixby R, Rothberg E (2007) Progress in computational mixed integer programming—A look back from the other side of the tipping point. *Annals of Operations Research* 149(1):37–41, ISSN 1572-9338, URL `http://dx.doi.org/10.1007/s10479-006-0091-y`.

Bortfeldt A, Jungmann S (2012) A tree search algorithm for solving the multi-dimensional strip packing problem with guillotine cutting constraint. *Annals of Operations Research* 196(1):53–71, ISSN 1572-9338, URL `http://dx.doi.org/10.1007/s10479-012-1084-7`.

Bortfeldt A, Winter T (2009) A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces. *International Transactions in Operational Research* 16(6):685–713, ISSN 1475-3995, URL `http://dx.doi.org/10.1111/j.1475-3995.2009.00701.x`.

Byrd RH, Nocedal J, Waltz RA (2006) Knitro: An Integrated Package for Nonlinear Optimization. Di Pillo G, Roma M, eds., *Large-Scale Nonlinear Optimization*, 35–59, Nonconvex Optimization and Its Applications (Boston, MA: Springer US), ISBN 978-0-387-30065-8, URL `http://dx.doi.org/10.1007/0-387-30065-1_4`.

Charalambous C, Fleszar K (2011) A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. *Computers & Operations Research* 38(10):1443–1451, ISSN 0305-0548, URL `http://dx.doi.org/10.1016/j.cor.2010.12.013`.

Chen Q, Cui Y, Chen Y (2016) Sequential value correction heuristic for the two-dimensional cutting stock problem with three-staged homogenous patterns. *Optimization Methods and Software* 31(1):68–87, ISSN 1055-6788, URL `http://dx.doi.org/10.1080/10556788.2015.1048860`.

Christofides N, Whitlock C (1977) An Algorithm for Two-Dimensional Cutting Problems. *Operations Research* 25(1):30–44, ISSN 0030-364X, URL `http://dx.doi.org/10.1287/opre.25.1.30`.

Cui Y (2008) Heuristic and exact algorithms for generating homogenous constrained three-staged cutting patterns. *Computers & Operations Research* 35(1):212–225, ISSN 0305-0548, URL `http://dx.doi.org/10.1016/j.cor.2006.02.029`.

Cui Y, Gu T, Hu W (2008) An algorithm for the constrained two-dimensional rectangular multiple identical large object placement problem. *Optimization Methods and Software* 23(3):375–393, ISSN 1055-6788, URL `http://dx.doi.org/10.1080/10556780701617163`.

Cui Y, Yang L, Chen Q (2013) Heuristic for the rectangular strip packing problem with rotation of items. *Computers & Operations Research* 40(4):1094–1099, ISSN 0305-0548, URL `http://dx.doi.org/10.1016/j.cor.2012.11.020`.

Cui Y, Yao Y, Cui YP (2016) Hybrid approach for the two-dimensional bin packing problem with two-staged patterns. *International Transactions in Operational Research* 23(3):539–549, ISSN 1475-3995, URL `http://dx.doi.org/10.1111/itor.12188`.

Cui Y, Zhao Z (2013) Heuristic for the rectangular two-dimensional single stock size cutting stock problem with two-staged patterns. *European Journal of Operational Research* 231(2):288–298, ISSN 0377-2217, URL `http://dx.doi.org/10.1016/j.ejor.2013.05.042`.

Cui YP, Cui Y, Tang T, Hu W (2015) Heuristic for constrained two-dimensional three-staged patterns. *Journal of the Operational Research Society* 66(4):647–656, ISSN 0160-5682, URL `http://dx.doi.org/10.1057/jors.2014.33`.

Cui YP, Yao Y, Zhang D (2018) Applying triple-block patterns in solving the two-dimensional bin packing problem. *Journal of the Operational Research Society* 69(3):402–415, ISSN 0160-5682, URL `http://dx.doi.org/10.1057/s41274-016-0148-5`.

Cui YP, Zhou Y, Cui Y (2017) Triple-solution approach for the strip packing problem with two-staged patterns. *Journal of Combinatorial Optimization* 34(2):588–604, ISSN 1573-2886, URL `http://dx.doi.org/10.1007/s10878-016-0088-7`.

Cung VD, Hifi M, Cun BL (2000) Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm. *International Transactions in Operational Research* 7(3):185–210, ISSN 1475-3995, URL `http://dx.doi.org/10.1111/j.1475-3995.2000.tb00194.x`.

do Nascimento OX, de Queiroz TA, Junqueira L (2019) A MIP-CP based approach for two- and three-dimensional cutting problems with staged guillotine cuts. *Annals of Operations Research* ISSN 1572-9338, URL `http://dx.doi.org/10.1007/s10479-019-03466-x`.

Dolatabadi M, Lodi A, Monaci M (2012) Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research* 39(1):48–53, ISSN 0305-0548, URL `http://dx.doi.org/10.1016/j.cor.2010.12.018`.

Fayard D, Hifi M, Zissimopoulos V (1998) An efficient approach for large-scale two-dimensional guillotine cutting stock problems. *Journal of the Operational Research Society* 49(12):1270–1277, ISSN 0160-5682, URL `http://dx.doi.org/10.1057/palgrave.jors.2600638`.

Fekete SP, Schepers J (1997) A new exact algorithm for general orthogonal d-dimensional knapsack problems. Burkard R, Woeginger G, eds., *Algorithms — ESA '97*, 144–156, Lecture Notes in Computer Science (Berlin, Heidelberg: Springer), ISBN 978-3-540-69536-3, URL `http://dx.doi.org/10.1007/3-540-63397-9_12`.

Fleszar K (2013) Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. *Computers & Operations Research* 40(1):463–474, ISSN 0305-0548, URL `http://dx.doi.org/10.1016/j.cor.2012.07.016`.

Furini F, Malaguti E, Thomopulos D (2016) Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. *INFORMS Journal on Computing* 28(4):736–751, ISSN 1091-9856, URL `http://dx.doi.org/10.1287/ijoc.2016.0710`.

Gilmore PC, Gomory RE (1965) Multistage Cutting Stock Problems of Two and More Dimensions. *Operations Research* 13(1):94–120, ISSN 0030-364X, URL `http://dx.doi.org/10.1287/opre.13.1.94`.

Hifi M (1997) An improvement of viswanathan and bagchi's exact algorithm for constrained two-dimensional cutting stock. *Computers & Operations Research* 24(8):727–736, ISSN 0305-0548, URL http://dx.doi.org/10.1016/S0305-0548(96)00095-0.

Hifi M, M'Hallah R (2009) Beam search and non-linear programming tools for the circular packing problem. *International Journal of Mathematics in Operational Research* 1(4):476–503, ISSN 1757-5850, URL http://dx.doi.org/10.1504/IJMOR.2009.026278.

Hifi M, M'Hallah R, Saadi T (2008) Algorithms for the Constrained Two-Staged Two-Dimensional Cutting Problem. *INFORMS Journal on Computing* 20(2):212–221, ISSN 1091-9856, URL http://dx.doi.org/10.1287/ijoc.1070.0233.

Hifi M, Negre S, Ouafi R, Saadi T (2012) A parallel algorithm for constrained two-staged two-dimensional cutting problems. *Computers & Industrial Engineering* 62(1):177–189, ISSN 0360-8352, URL http://dx.doi.org/10.1016/j.cie.2011.09.005.

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(4):465–494, ISSN 1573-2886, URL http://dx.doi.org/10.1023/A:1011628809603.

Hopper E (2000) *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. PhD Thesis, University of Wales. Cardiff.

Hopper E, Turton BCH (2001) An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research* 128(1):34–57, ISSN 0377-2217, URL http://dx.doi.org/10.1016/S0377-2217(99)00357-4.

Kröger B (1995) Guillotineable bin packing: A genetic approach. *European Journal of Operational Research* 84(3):645–661, ISSN 0377-2217, URL http://dx.doi.org/10.1016/0377-2217(95)00029-P.

Lodi A, Martello S, Vigo D (2004) Models and Bounds for Two-Dimensional Level Packing Problems. *Journal of Combinatorial Optimization* 8(3):363–379, ISSN 1573-2886, URL http://dx.doi.org/10.1023/B:JOCO.0000038915.62826.79.

Lodi A, Monaci M (2003) Integer linear programming models for 2-staged two-dimensional Knapsack problems. *Mathematical Programming* 94(2):257–278, ISSN 1436-4646, URL http://dx.doi.org/10.1007/s10107-002-0319-9.

Lodi A, Monaci M, Pietrobuoni E (2017) Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints. *Discrete Applied Mathematics* 217:40–47, ISSN 0166-218X, URL `http://dx.doi.org/10.1016/j.dam.2015.09.012`.

Martello S, Vigo D (1998) Exact Solution of the Two-Dimensional Finite Bin Packing Problem. *Management Science* 44(3):388–399, ISSN 0025-1909, URL `http://dx.doi.org/10.1287/mnsc.44.3.388`.

Martin M, Hokama PHDB, Morabito R, Munari P (2019) The constrained two-dimensional guillotine cutting problem with defects: an ILP formulation, a Benders decomposition and a CP-based algorithm. *International Journal of Production Research* 0(0):1–18, ISSN 0020-7543, URL `http://dx.doi.org/10.1080/00207543.2019.1630773`.

Morabito R, Pureza V (2010) A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem. *Annals of Operations Research* 179(1):297–315, ISSN 1572-9338, URL `http://dx.doi.org/10.1007/s10479-008-0457-4`.

Nepomuceno N, Pinheiro P, Coelho ALV (2008) A Hybrid Optimization Framework for Cutting and Packing Problems. Cotta C, van Hemert J, eds., *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, 87–99, Studies in Computational Intelligence (Berlin, Heidelberg: Springer), ISBN 978-3-540-70807-0, URL `http://dx.doi.org/10.1007/978-3-540-70807-0_6`.

Oliveira J, Ferreira J (1990) An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research* 44(2):256–266, ISSN 0377-2217, URL `http://dx.doi.org/10.1016/0377-2217(90)90361-E`.

Puchinger J, Raidl GR (2007) Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research* 183(3):1304–1327, ISSN 0377-2217, URL `http://dx.doi.org/10.1016/j.ejor.2005.11.064`.

Silva E, Alvelos F, Valério de Carvalho JM (2010) An integer programming model for two- and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research* 205(3):699–708, ISSN 0377-2217, URL `http://dx.doi.org/10.1016/j.ejor.2010.01.039`.

Tschöke S, Holthöfer N (1995) A new parallel approach to the constrained two-dimensional cutting stock problem. Ferreira A, Rolim J, eds., *Parallel Algorithms for Irregularly Structured Problems*, 285–300, Lecture Notes in Computer Science (Berlin, Heidelberg: Springer), ISBN 978-3-540-44915-7, URL http://dx.doi.org/10.1007/3-540-60321-2_24.

Velasco AS, Uchoa E (2019) Improved state space relaxation for constrained two-dimensional guillotine cutting problems. *European Journal of Operational Research* 272(1):106–120, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2018.06.016.

Wang N, Lim A, Zhu W (2013) A multi-round partial beam search approach for the single container loading problem with shipment priority. *International Journal of Production Economics* 145(2):531–540, ISSN 0925-5273, URL http://dx.doi.org/10.1016/j.ijpe.2013.04.028.

Wang PY (1983) Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research* 31(3):573–586, ISSN 0030-364X, URL http://dx.doi.org/10.1287/opre.31.3.573.

Wei L, Lim A (2015) A bidirectional building approach for the 2D constrained guillotine knapsack packing problem. *European Journal of Operational Research* 242(1):63–71, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2014.10.004.

Wei L, Tian T, Zhu W, Lim A (2014) A block-based layer building approach for the 2D guillotine strip packing problem. *European Journal of Operational Research* 239(1):58–69, ISSN 0377-2217, URL http://dx.doi.org/10.1016/j.ejor.2014.04.020.