CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

The Meet-in-the-Middle Principle for Cutting and Packing Problems

Jean-François Côté
Manuel Iori

June 2016

CIRRELT-2016-28

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2016-007.

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

The Meet-in-the-Middle Principle for Cutting and Packing Problems

Jean-François Côté[1,*], Manuel Iori[2]

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, 2325 de la Terrasse, Université Laval, Québec, Canada G1V 0A6

[2] University of Modena and Reggio Emilia, via Amendola 2, 42122 Reggio Emilia, Italy

Abstract. Cutting and packing (C&P) is a fundamental research area that models a large number of managerial and industrial optimization issues. A solution to a C&P problem basically consists in a set of one- or multi- dimensional items packed in/cut from one or more bins, by satisfying problem constraints and minimizing a given objective function. Normal patterns are a well-known C&P technique used to build solutions where each item is aligned to the bottom of the bin along each dimension. The rationale in their use is that they can reduce the search space while preserving optimality, but the drawback is that their number grows consistently when the number of items and the size of the bin increase. In this paper we propose a new set of patterns, called meet-in-the-middle, that lead to several interesting results. Their computation is achieved with the same time complexity as that of the normal patterns, but their number is never higher, and in practical applications it frequently shows reductions of about 50%. The new patterns are applied to improve some state-of-the-art C&P techniques, including arc-flow formulations, combinatorial branch-and-bound algorithms, and mixed integer linear programs. The efficacy of the improved techniques is assessed by extensive computational tests on a number of relevant applications.

Keywords. Cutting and packing, normal patterns, meet-in-the-middle, cutting stock, orthogonal packing.

Acknowledgement. The second author acknowledges financial support by CAPES/Brazil under grant PVE nº A007/2013.

---

* Corresponding author: Jean-Francois.Cote@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2016

1 Introduction

A solution to a *cutting and packing* (C&P) problem basically consists in a set of one- or multi-dimensional items packed in (or cut from) one or more bins, by satisfying some constraints and minimizing a given objective function. Typical constraints impose that all items should lie entirely within the bin in which they are packed, and that they should not overlap among themselves. Typical objective functions require the maximization of some item profits (knapsack problems) or the minimization of the number of the selected bins (bin packing problems and variants).

C&P problems are a fundamental research area in the field of Operations Research, as they model several real-world issues, arising for example in production industry (see, e.g., Vanderbeck 2001), transportation (see, e.g., Iori et al. 2007), and container-loading (see, e.g., Bortfeldt and Wäscher 2013), just to cite some. We refer to Wäscher et al. (2007) for an extensive typology of the several C&P problems and for further hints on their applications.

Normal patterns are a well-known C&P technique that builds solutions where each item is aligned to the bottom of each dimension. They were independently introduced first by Herz (1972) (who called them *canonical dissections*) and then by Christofides and Whitlock (1977) in the context of two-dimensional cutting, and have been used later on in literally hundreds of algorithms. Consider for example Figure 1: Figure 1-(a) gives a solution to a general two-dimensional problem where eight items are feasibly packed in a single bin; Figure 1-(b) provides an equivalent solution satisfying the principle of normal patterns. Intuitively, solution (a) can be transformed in solution (b) by repeatedly moving each item to its left and/or down until its border touches that of another item or that of the bin. On the basis of this observation, Herz (1972) and Christofides and Whitlock (1977) defined the set of normal patterns as the set of all possible item width combinations, and then proved that the search for an optimum may be limited to solutions where each item is packed in a normal pattern.

The drawback of normal patterns is that their efficacy is noticed almost only at the beginning of the bin, and then decreases consistently towards the end of it. Intuitively, a certain width p is a normal pattern if there exists a combination of item widths whose sum is p. This might be difficult to obtain for small p values, but becomes easier for large values. In practice, when the number of items is high and the bin width is large, the effect of the normal patterns tends to be irrelevant in the second half of the bin.

Several attempts have been developed in the literature to try to overcome this drawback. In this paper we continue this line of research and propose an idea that proved to be very effective in practice. It consists of a new set of patterns, called *meet-in-the-middle* (MIM), obtained by aligning items along each dimension either to the bottom of the bin or to the top of it. In details, we consider the first dimension of the bin (the width), fix a certain threshold value along it (for example, the half bin width), force all items whose left border is at the left of the threshold to be left aligned, and force the remaining items to be right aligned. The process is then repeated for the successive dimensions. Consider again Figure 1 and suppose that the thresholds for the two dimensions are fixed to, respectively, the half bin width and the half bin height (dashed lines). Then, using the

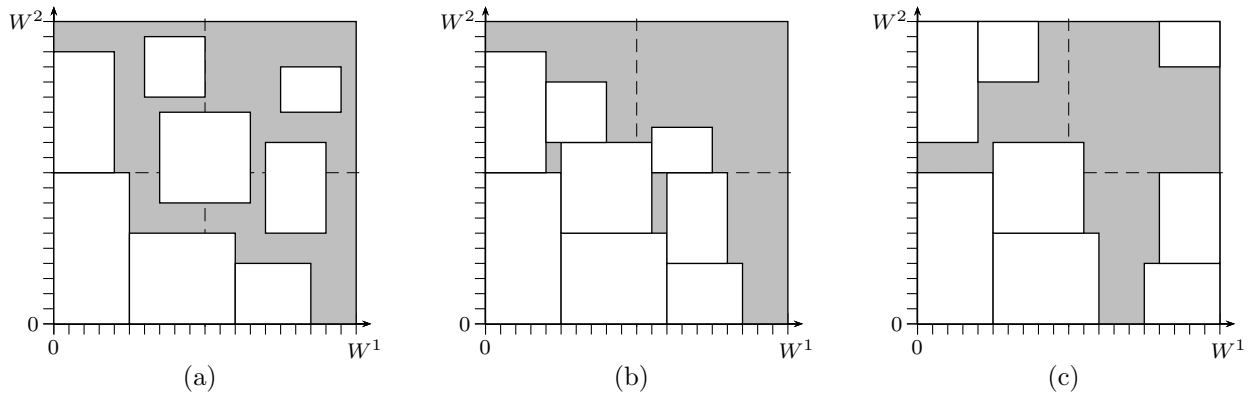

Figure 1: (a) general 2D packing; (b) packing satisfying *normal patterns* (*canonical dissections*); (c) packing satisfying *meet-in-the-middle patterns* (with thresholds $W^1/2$ and $W^2/2$).

MIM idea solution (a) is transformed into solution (c). As it will be shown in the next section, the search for an optimum may be limited to solutions satisfying the MIM patterns.

The idea is simple, yet it provides interesting results. The number of the MIM patterns is never higher that that of the normal ones, and in practical applications it is much smaller. Moreover, further reductions may be obtained by additional preprocessing techniques. In addition, the computational effort required to compute the MIM patterns is the same than that required for the normal ones. We also note that the MIM principle is useful to reduce not only the normal patterns, but also other forms of patterns that have been presented in the literature (and indeed our computational work builds upon the patterns by Boschetti et al. 2002).

To the best of our knowledge the MIM principle has not been investigated in the literature, and in the next sections we describe its application to some classical C&P problems. Indeed, the MIM patterns easily apply to several optimization techniques, such as mathematical formulations, where they allow to reduce the number of variables and constraints, and combinatorial enumeration trees, where they can be used to fathom unnecessary nodes.

The name that we adopted comes from cryptography ("meet in the middle attack"). It has been used previously in the C&P literature by Horowitz and Sahni (1974) to describe their branch-and-bound algorithm for the binary knapsack problem. They divide the input item set, having n items, into two mutually-exclusive subsets having $n/2$ items each. They enumerate the partial solutions on each subset, and then merge the partial solutions to build an optimal one. The complexity of their algorithm is $O(2^{n/2})$, which is the best known for the problem. Their approach is very different from the one that we propose here, as it divides the set of items instead of dividing the space of the bin along each dimension.

The remainder of the paper is organized as follows. In Section 2 the related literature is revised,

the MIM patterns are presented and some properties are discussed. The remaining sections describe some relevant applications to well-known C&P problems. The classical cutting stock and bin packing problems are solved in Section 3, the "old" two-dimensional two-stage cutting stock problem in Section 4, and the fundamental two-dimensional orthogonal packing problem in Section 5.

All sections provide evidence of our assessments by means of extensive computational tests. These have been obtained by implementing all algorithms in C++ and running them on a PC equipped with an Intel 2.667 GHz Westmere EP X5650 processor. We used Cplex 12.6 as *mixed integer linear programming* (MILP) solver, imposing it to run on a single thread.

2 Meet-in-the-Middle Principle

We consider a generic orthogonal C&P problem in k dimensions, in which we are given a set $I = \{1, 2, \ldots, n\}$ of items and a bin. Both the items and the bin are k-dimensional rectangular boxes. Each item $i \in I$ has width w_i^d and the bin has width W^d, for $d = 1, 2, \ldots, k$. We suppose that all widths are positive integer values.

A *feasible* solution is a packing of an item set $I' \subseteq I$ in the bin, such that all items are completely contained in the bin and they do not overlap one with the other. Two feasible solutions are *equivalent* one with the other if they pack the same item set I'. We say that a reduction property *preserves optimality*, if it possibly removes some solutions but guarantees that at least one is kept among all equivalent solutions for any set $I' \subseteq I$.

We make use of a Cartesian coordinate system whose axes are parallel to the edges of the bin (see Figure 1). For a given box, we call *lowest* the box corner that is closest to the origin of the system (in two dimensions, the lowest corner is the bottom-left one). We say that an item i is packed in position p_i if its lowest corner is in p_i.

Our techniques apply to any dimension d, but for the sake of clarity in the remaining of this section we focus on the first dimension, that is, the width. When no confusion arises, we thus write for short W instead of W^d and w_i instead of w_i^d. For descriptive purposes, in the next Sections 2.1-2.3 we resume the following example a number of times.

Example 1 *A simple one-dimensional instance with* $W = 27$*,* $n = 4$*, and* $w = (5, 10, 12, 15)$.

2.1 Normal Patterns and Known Reductions

According to Herz (1972) and Christofides and Whitlock (1977) the set of normal patterns can be formally defined as

$$\mathcal{N}_0 = \left\{ x = \sum\nolimits_{j \in I} w_j \xi_j : 0 \leq x \leq W, \xi_j \in \{0, 1\} \text{ for } j \in I \right\}. \tag{1}$$

For Example 1, we have $\mathcal{N}_0 = \{0, 5, 10, 12, 15, 17, 20, 22, 25, 27\}$. This set was introduced in the context of cutting, and also includes patterns positioned towards the end of the bin that are only

needed to model cuts for the right borders of the items. If one is interested instead in modeling only the positions where the items can be packed (lowest corners), as in our case, then $\mathcal{N}_0$ can be conveniently reduced. Following Beasley (1985a), this results in

$$\mathcal{N} = \{x \in \mathcal{N}_0 : x \leq W - w_{\min}\}, \tag{2}$$

where $w_{\min} = \min_{j \in I}\{w_j\}$. Set $\mathcal{N}$ models the simple fact that no item can be packed with its lowest corner after $W - w_{\min}$. In the remaining of the paper we refer to (2) when we mention normal patterns. For Example 1, we have $\mathcal{N} = \{0, 5, 10, 12, 15, 17, 20, 22\}$.

Terno et al. (1987) (see also Scheithauer and Terno 1996) attempt to reduce $\mathcal{N}_0$ as follows. For a given pattern $p \in \mathcal{N}_0$, let $\bar{w}(W-p) = \max\{x = \sum_{j \in S} w_j : x \leq W-p, S \subseteq I\}$. If $p+\bar{w}(W-p) < W$, then packing an item in p leads to a loss of at least $W - p - \bar{w}(W-p)$ width units. If there is a subsequent pattern, $q > p$, such that $q+\bar{w}(p) \leq W$, then the packing of an item in p can be skipped, as an equivalent or better solution can be obtained by packing it in q (as this would lead to a not greater loss). Moreover, Terno et al. (1987) noticed that for the computation of the $\bar{w}(W-p)$ values one can use directly the entries in $\mathcal{N}_0$. Formally, by setting $\bar{w}(W-p) = \max\{x \in \mathcal{N}_0 : x \leq W-p\}$, the set $\mathcal{T}_0$ of the so-called reduced raster points (*raster points* for short in the following) is then

$$\mathcal{T}_0 = \{\bar{w}(W-p) : p \in \mathcal{N}_0\}. \tag{3}$$

For Example 1, we have $\mathcal{T}_0 = \{0, 5, 10, 12, 15, 17, 22, 27\}$. The only item that can be packed in 20 has width 5, but this option is skipped as an equivalent solution can be obtained by packing the item in 22. Similarly to what done for $\mathcal{N}_0$, here we reduce the raster points by computing $\mathcal{T} = \{x \in \mathcal{T}_0 : x \leq W - w_{\min}\}$.

Boschetti et al. (2002) followed a different strategy, and conveniently reduced the set of normal patterns *for a given item* i, by computing its possible patterns as combinations of items rather than i. Formally, let

$$\mathcal{B}_i = \left\{x = \sum\nolimits_{j \in I \setminus \{i\}} w_j \xi_j : 0 \leq x \leq W - w_i, \xi_j \in \{0,1\} \text{ for } j \in I \setminus \{i\}\right\} \tag{4}$$

be the set of patterns for any $i \in I$, and then compute the overall set as $\mathcal{B} = \cup_{i \in I} \mathcal{B}_i$. To avoid confusion with the standard normal patterns, in the following we call $\mathcal{B}$ the set of *regular normal patterns*. For Example 1, $\mathcal{B}_1 = \{0, 10, 12, 15, 22\}$, $\mathcal{B}_2 = \{0, 5, 12, 15, 17\}$, $\mathcal{B}_3 = \{0, 5, 10, 15\}$, $\mathcal{B}_4 = \{0, 5, 10, 12\}$, and hence $\mathcal{B} = \{0, 5, 10, 12, 15, 17, 22\}$. For this example $\mathcal{B}$ is coincident with $\mathcal{T}$. In general, it is easy to see that $\mathcal{T} \subseteq \mathcal{N}$ and $\mathcal{B} \subseteq \mathcal{N}$, but no fixed relation exists between $\mathcal{T}$ and $\mathcal{B}$.

As shown in Christofides and Whitlock (1977), the computation of $\mathcal{N}_0$ may be obtained by a standard dynamic programming procedure, that we report in Algorithm 1. Procedure `NormalPatterns` works in $O(nW)$: It first computes the feasible item width combinations using a support array T and then stores the resulting values in $\mathcal{N}_0$. The computation of $\mathcal{B}_i$, for any $i \in I$, may be obtained by invoking `NormalPatterns`$(I \setminus \{i\}; W - w_i)$, and thus the computation of $\mathcal{B}$ requires $O(n^2 W)$.

Algorithm 1 `NormalPatterns`$(I;W)$

Require: I: set of items, W: bin width
$T \leftarrow [0 \text{ to } W]$: an array with all entries initialized to 0
$T[0] \leftarrow 1$
for $i \in I$ **do**
    for $p = W - w_i$ to 0 **do**
        if $T[p] = 1$ **then** $T[p + w_i] \leftarrow 1$
$\mathcal{N}_0 \leftarrow \emptyset$
for $p = W$ to 0 **do**
    if $T[p] = 1$ **then** $\mathcal{N}_0 \leftarrow \mathcal{N}_0 \cup \{p\}$
return $\mathcal{N}_0$

In the following we pursue the idea of reducing the number of patterns for each item, thus building upon the regular normal patterns in (4). This is done because these patterns led to good computational results in several applications (see, e.g., Boschetti and Montaletti 2010 and Côté et al. 2014a), and because they model in a natural way choices made in standard C&P techniques such as arc-flow formulations and combinatorial branch-and-bound algorithms. However, our results can be easily adapted to the simpler case of a direct computation of the standard normal patterns starting from (2).

We note that other attempts have been made in the literature to reduce the normal patterns. Among these we mention: Côté et al. (2014a), that use a quick preprocessing technique in the idea of the raster points; Côté et al. (2014b), that divide a set of two-dimensional items into two subsets according to an input packing ordering, and then pack the first half on the bottom of the bin and the second half on the top; and Alvarez-Valdes et al. (2005), who propose reduction rules for the specific case where just two item widths are given (pallet loading).

2.2 Meet-in-the-Middle Patterns

The *meet-in-the-middle* (MIM) patterns are defined for each item $i \in I$ and for a threshold $t \in \{1, 2, \ldots, W\}$ as the combination of two types of patterns. First the *left* patterns are computed as

$$\mathcal{L}_{it} = \left\{ x = \sum\nolimits_{j \in I \setminus \{i\}} w_j \xi_j : 0 \le x \le \min\{t-1, W - w_i\}, \xi_j \in \{0,1\} \text{ for } j \in I \setminus \{i\} \right\}, \tag{5}$$

and then the *right* patterns as

$$\mathcal{R}_{it} = \left\{ x' = W - w_i - x : x = \sum\nolimits_{j \in I \setminus \{i\}} w_j \xi_j, 0 \le x \le W - w_i - t, \xi_j \in \{0,1\} \text{ for } j \in I \setminus \{i\} \right\}. \tag{6}$$

In practice, an item is packed in a left pattern when the coordinate x of its lowest corner is at the left of t ($x \le t - 1$), and in a right pattern otherwise ($x \ge t$). Refer again to Figure 1-(c) for a

graphical example. The "min" function in (5) is used to impose that an item i is not packed after $W - w_i$. Note also that large items having width $w_i > W - t$ are always left aligned, because $\mathcal{R}_{it} = \emptyset$ when $W - w_i - t < 0$. The set of MIM patterns for item i is then simply assessed by

$$\mathcal{M}_{it} = \mathcal{L}_{it} \cup \mathcal{R}_{it}, \tag{7}$$

and the overall set as

$$\mathcal{M}_t = \cup_{i \in I} \mathcal{M}_{it}. \tag{8}$$

The computation of each $\mathcal{M}_{it}$ set may be obtained in $O(nW)$ by running Algorithm 2. The left patterns are computed by determining all item combinations, rather than the selected item i, whose total width does not exceed $t - 1$ and the residual space left by w_i in the bin. For the right patterns, we first compute the standard set of (left-aligned) normal patterns whose total width does not exceed the residual space, if any, obtained by subtracting from the bin width both w_i and t (consider that `NormalPatterns` returns the empty set when $W - w_i - t < 0$). Then we obtain $\mathcal{R}_{it}$ by mapping each left-aligned pattern p into a right-aligned pattern $W - w_i - p$.

Algorithm 2 `MIMPatterns`$(I;i;W;t)$

Require: I: set of items, i: selected item, W: bin width, t: threshold value
$\mathcal{L}_{it} \leftarrow$ `NormalPatterns`$(I \setminus \{i\}; \min\{t - 1, W - w_i\})$
$\mathcal{R}'_{it} \leftarrow$ `NormalPatterns`$(I \setminus \{i\}; W - w_i - t)$
$\mathcal{R}_{it} \leftarrow \emptyset$
for $p \in \mathcal{R}'_{it}$ **do**
    $\mathcal{R}_{it} \leftarrow \mathcal{R}_{it} \cup \{W - w_i - p\}$
$\mathcal{M}_{it} \leftarrow \mathcal{L}_{it} \cup \mathcal{R}_{it}$
return $\mathcal{M}_{it}$

For Example 1, when $t = 10$ we have $\mathcal{L}_{1,10} = \{0\}$, $\mathcal{R}_{1,10} = \{10, 12, 22\}$, and so $\mathcal{M}_{1,10} = \{0, 10, 12, 22\}$. For the next items we get $\mathcal{M}_{2,10} = \{0, 5, 12, 17\}$, $\mathcal{M}_{3,10} = \{0, 5, 10, 15\}$, and $\mathcal{M}_{4,10} = \{0, 5, 12\}$. In this case the resulting set $\mathcal{M}_{10}$ is coincident with $\mathcal{B}$, but overall we have $\sum_i |\mathcal{M}_{i,10}| = 11 < \sum_i |\mathcal{B}_i| = 18$, thus resulting in fewer packing options for the items.

Some interesting properties may be noticed for the MIM patterns.

Proposition 1 *Optimality is preserved by considering only solutions where all items are packed in MIM patterns.*

Proof The proof follows the footsteps of the simple one in Herz (1972) and Christofides and Whitlock (1977). Suppose a feasible packing for a generic item set $I' \subseteq I$ is provided. Then select an item $i \in I'$ not packed in a MIM pattern, if any, and repeat the following procedure for each dimension: if the lowest corner of i is at the left of t, then move i as much as possible to the left, otherwise move it as much as possible to the right. Reiterate until all items are packed in a MIM pattern. The thesis follows because the procedure holds for any $I' \subseteq I$. □

Proposition 2 *The set of MIM patterns computed for $t = W$ is equivalent to the set $\mathcal{B}$.*

Proof By replacing t with W in (5) and (6) we obtain $\min\{t-1, W-w_i\} = W-w_i$, so $\mathcal{L}_{iW} = \mathcal{B}_i$, and $W - w_i - t < 0$, so $\mathcal{R}_{iW} = \emptyset$. Consequently $\mathcal{M}_{iW} = \mathcal{B}_i$ for any $i \in I$, and thus $\mathcal{M}_W = \mathcal{B}$. □

As it will be shown in Section 2.4 below, the value taken by t may have a strong impact on the cardinality of the resulting set $\mathcal{M}_t$. We thus define the *minimal* set of MIM patterns as

$$\mathcal{M} = \left\{ \mathcal{M}_t : t = \text{argmin}_{s\in\{1,2,\ldots,W\}} \sum\nolimits_{i\in I} |\mathcal{M}_{is}| \right\}. \tag{9}$$

A direct consequence of Property 2 and of the minimality of $\mathcal{M}$ is the following.

Proposition 3 $|\mathcal{M}| \leq |\mathcal{B}|$ *(and consequently $|\mathcal{M}| \leq |\mathcal{N}|$).*

The computation of the minimal set $\mathcal{M}$ may be trivially obtained by invoking Algorithm 2 for each value of t and storing the best result according to (9), thus using a time complexity of $O(n^2W^2)$. A better implementation reduces the computational effort as follows.

Proposition 4 *The minimal set $\mathcal{M}$ of MIM patterns can be computed in $O(n^2W)$.*

Proof The proof is based on Algorithm 3, which we describe in details. We first compute all sets $\mathcal{B}_i$ and we sum them together to determine the number of left patterns having value p and the number of right patterns having value $W - w_i - p$ (steps 2-9). The arrays T_{left} and T_{right} store the resulting values. The same arrays are then used to compute the cumulative number of patterns in an incremental way, on the basis of the following observation.

Let us consider two threshold values t_1 and t_2, with $t_2 \geq t_1 + 1$. We rewrite twice equation (5), by first replacing t with t_1 and then with t_2. Then, by computing the difference between the two resulting equations, we obtain

$$\mathcal{L}_{i,t_2} = \mathcal{L}_{i,t_1} \cup \{x = \sum\nolimits_{j\in I\setminus\{i\}} w_j\xi_j : t_1 \leq x \leq \min\{t_2 - 1, W - w_i\}, \xi_j \in \{0,1\} \text{ for } j \in I \setminus \{i\}\}.$$

In other words, the left patterns up to $t_2 - 1$ may be computed by summing those up to $t_1 - 1$ and those in the interval $[t_1, t_2 - 1]$. The same reasoning applies to the right patterns, by using an incremental process from right to left.

Coming back to Algorithm 3, the incremental computation of the left and right patterns is performed at steps 10-13. Then, steps 14-20 determine the threshold value $t_{\min}$ for which the overall number of patterns is a minimum, and steps 21-29 build the resulting MIM patterns. □

An important remark is thus that the minimal set of MIM patterns may reduce the set of regular normal patterns defined in (4), and may be computed with the same algorithmic complexity. Note that the same remark applies when considering the original normal patterns in (2): the resulting number of MIM patterns would not exceed that of the normal patterns because of Property 3; their computation would require $O(nW)$ (the same complexity required for $\mathcal{N}$) by a simplified version

Algorithm 3 `MinimalMIMSet`$(I;W)$

1: **Require:** I: set of items, W: bin width
2: $T_{left}, T_{right} \leftarrow [0 \text{ to } W]$: two arrays with all entries initialized at zero
3: **for** $i \in I$ **do**
4: $\quad \mathcal{B}_i \leftarrow$ `NormalPatterns`$(I \setminus \{i\}, W - w_i)$
5: $\quad$ **for** $p \in \mathcal{B}_i$ **do**
6: $\quad\quad T_{left}[p] \leftarrow T_{left}[p] + 1$
7: $\quad\quad T_{right}[W - w_i - p] \leftarrow T_{right}[W - w_i - p] + 1$
8: $\quad$ **end for**
9: **end for**
10: **for** $p = 1$ **to** W **do**
11: $\quad T_{left}[p] \leftarrow T_{left}[p] + T_{left}[p-1]$
12: $\quad T_{right}[W - p] \leftarrow T_{right}[W - p] + T_{right}[W - (p-1)]$
13: **end for**
14: $t_{\min} \leftarrow 1$
15: $\min \leftarrow T_{left}[0] + T_{right}[1]$
16: **for** $p = 2$ **to** W **do**
17: $\quad$ **if** $T_{left}[p-1] + T_{right}[p] < \min$ **then**
18: $\quad\quad \min \leftarrow T_{left}[p-1] + T_{right}[p]$
19: $\quad\quad t_{\min} \leftarrow p$
20: $\quad$ **end if**
21: $\mathcal{M} \leftarrow \emptyset$
22: **for** $i \in I$ **do**
23: $\quad \mathcal{M}_i \leftarrow \emptyset$
24: $\quad$ **for** $p \in \mathcal{B}_i$ **do**
25: $\quad\quad$ **if** $p < t_{\min}$ **then** $\mathcal{M}_i \leftarrow \mathcal{M}_i \cup \{p\}$
26: $\quad\quad$ **if** $W - w_i - p \geq t_{\min}$ **then** $\mathcal{M}_i \leftarrow \mathcal{M}_i \cup \{W - p - w_i\}$
27: $\quad$ **end for**
28: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_i$
29: **end for**
30: **return** $\mathcal{M}$

of Algorithm 3 in which the computation of all $\mathcal{B}_i$ is replaced by that of $\mathcal{N}$, and the double loops in i and p are replaced by single loops in p. Note also that one could be interested in using a different criterion for the minimality of $\mathcal{M}$, for example replacing $\sum_{i \in I} |\mathcal{M}_{is}|$ with $|\mathcal{M}_s|$ in (9). In this case too Property 4 holds via a trivial modification of Algorithm 3, in which at steps 6 and 7 $T_{left}[p] \leftarrow T_{left}[p] + 1$ and $T_{right}[W - w_i - p] \leftarrow T_{right}[W - w_i - p] + 1$ are replaced by, respectively, $T_{left}[p] \leftarrow 1$ and $T_{right}[W - w_i - p] \leftarrow 1$.

2.3 Further Reduction by Preprocessing Criteria

The MIM patterns can be reduced, while preserving optimality, by applying two criteria.

Proposition 5 *Consider a feasible solution packing a generic item set $I' \subseteq I$ in the bin. Consider then an item $k \in I'$ and a threshold value t. Among the equivalent solutions that pack I', there exists one satisfying both the MIM patterns and the following condition: if $t \leq \left\lceil \frac{W-w_k}{2} \right\rceil$, then k is packed in a right pattern, else it is packed in a left pattern.*

Proof Let us first prove the "if" part. Given a solution where k is packed in a left pattern, we transform it into an equivalent solution satisfying the MIM patterns and where k is packed in a right pattern. We use a two-step procedure (refer to Figure 2 below). In the first step we perform a reflection of the original solution using axle $x = W/2$, and obtain a *mirror solution*. An item i originally packed in a pattern p_i is packed in a pattern $p'_i = W - p_i - w_i$ in the mirror solution. The mirror solution is feasible but does not necessarily satisfy the MIM patterns, thus the second step of our procedure simply moves as much as possible to the left all items i packed in a pattern $p'_i \leq t-1$, and to the right all items i packed in $p'_i \geq t$. The new solution is feasible and accomplishes the MIM principle. To guarantee that in the new solution item k is in the right patterns, we need to show that $p'_k \geq t$. As k was packed in a left pattern, we have $p_k \leq t-1$, so $p'_k = W - p_k - w_k \geq W - w_k - t + 1$. Moreover, from the hypothesis we know that $W - w_k \geq 2\lceil (W - w_k)/2 \rceil - 1 \geq 2t - 1$, so we can conclude that $p'_k \geq 2t - 1 - t + 1 = t$. A similar reasoning applies to "else" part by using the fact that $t \geq \lceil (W - w_k)/2 \rceil + 1$, and this concludes the proof. □

Observe for example Figure 2, and consider $k = 2$ and $t = 4 \leq \lceil (W - w_k)/2 \rceil = 9$. Figure 2-(a) gives a solution in which k is packed in a left pattern; Figure 2-(b) provides the corresponding mirror solution; and Figure 2-(c) presents the solution obtained at the end of the second step of our procedure, satisfying the MIM principle and having k packed in a right pattern.

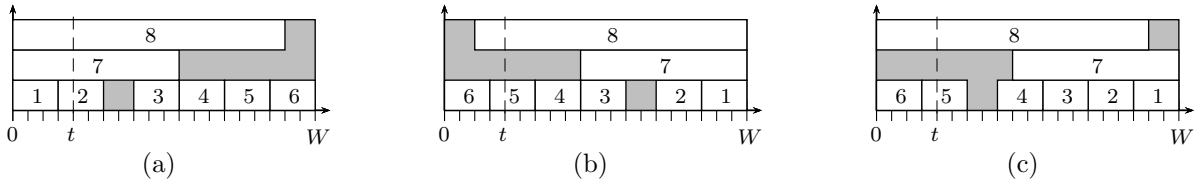

Figure 2: (a) solution satisfying MIM and having item $k = 2$ packed in a left pattern; (b) mirror solution; (c) solution satisfying MIM and having k packed in a right pattern.

We can thus apply Proposition 5 to reduce the search space while preserving optimality. We found computationally convenient to apply it in the following way.

Preprocessing 1 *Select an item k of minimum width, remove it from the computation of the left patterns when $t \leq \lceil (W - w_k)/2 \rceil$ and of the right patterns when $t \geq \lceil (W - w_k)/2 \rceil + 1$.*

Classical C&P preprocessing techniques attempt to increase the widths of the items as much as possible while preserving optimality (see, e.g., Boschetti et al. 2002). This usually results in more constrained packings that can be easier to solve in practice (because bounding techniques can have an improved performance). Here we pursue this line of research, but try to increase the width of an item when it is packed in a particular MIM pattern, and then show how this can also be used to reduce the number of patterns. To this aim let us first define

$$\widetilde{w}_{ip} = \text{width of item } i \in I \text{ when packed in pattern } p \in \mathcal{M}_i. \tag{10}$$

Proposition 6 *Consider an item $k \in I$ and a pattern $p \in \mathcal{M}_{kt}$ for an arbitrary threshold t.*

A) If $p \in \mathcal{L}_{kt}$, then let $q = \min\{W; \min\{s \in \mathcal{M}_{it} : i \in I \setminus \{k\}, s \geq p + w_k\}\}$. Then, optimality is preserved by setting $\widetilde{w}_{kp} = q - p$.

B) If $p \in \mathcal{R}_{kt}$, then let $q = \max\{0; \max\{s \in \mathcal{M}_{it} : i \in I \setminus \{k\}, s + w_i \leq p\}\}$. Then, optimality is preserved by setting $\mathcal{R}_{kt} = \mathcal{R}_{kt} \cup \{q\} \setminus \{p\}$, and setting $\widetilde{w}_{kq} = p + w_k - q$.

Proof Let us first concentrate on part A). Parameter q gives the value of the leftmost pattern that can be used for packing an item at the right of item k, and takes the value W if no such pattern exists. If $q > p$, then, because of Proposition 1, we know that among the optimal solutions there exists one in which no item is packed in a position belonging to the width interval from $p + w_k$ to $q - 1$ (where there are no MIM patterns). We can thus increase the item width to $\widetilde{w}_{ip} = q - p$, because this does not cause an overlapping with other items in any solution that does not violate the MIM principle.

The proof of part B) is somehow specular. In this case we consider the set of all item packings at the left of k, and in this set we compute q as the rightmost position where an item can end (0 if no such position exists). Then, in a solution satisfying the MIM principle no item can have its lowest corner in the interval between q and p. Thus, we can move to the left the current right pattern, from p to q, and then enlarge the item width to $\widetilde{w}_{kq} = p + w_k - q$ (thus preserving the same value for the right border of item k when packed in this pattern), knowing that the packing of k in q will not cause any overlapping with other items. □

Proposition 7 *Consider an item $k \in I$ and two patterns $p, s \in \mathcal{M}_{kt}$ with $p < s$. By using Proposition 6, enlarge the width of the item when it is packed in the two patterns to, respectively, $\widetilde{w}_{kp}$ and $\widetilde{w}_{ks}$. If $s + \widetilde{w}_{ks} \leq p + \widetilde{w}_{kp}$, then the removal of pattern p from $\mathcal{M}_{kt}$ preserves optimality.*

Proof There are two cases of interest, if item k if packed in pattern s, then it occupies the width interval $[s, s + \widetilde{w}_{ks}]$. If instead it is packed in p, then it occupies the interval $[p, p + \widetilde{w}_{kp}]$. If $p < s$ and $s + \widetilde{w}_{ks} \leq p + \widetilde{w}_{kp}$, then $[s, s + \widetilde{w}_{ks}] \subseteq [p, p + \widetilde{w}_{kp}]$. Consequently any solution where k is packed in p can be replaced by an equivalent one in which k is packed in s, without affecting optimality. Note that the opposite does not hold, as there could be items whose right border is in the interval between p and s, that can be packed side by side with k when it is packed in s, but not when it is packed in p. □

The above results lead to our second preprocessing criterion.

Preprocessing 2 *Enlarge the widths of all items in all patterns using Proposition 6, first for all the left patterns and then for all the right ones. Then remove redundant patterns, first left and then right, following Proposition 7.*

2.4 Evaluation

We conclude this section by presenting a numerical evaluation of the size of the different sets of patterns that we discussed. We concentrate on the widths of three well-known sets of two-dimensional instances, namely, the cgcut by Christofides and Whitlock (1977), and the gcut and ngcut by Beasley (1985a,b).

The results that we obtained are summarized in Table 1. For the literature, the table gives the cardinality of the sets (namely, $|\mathcal{N}|$, $|\mathcal{B}|$, and $|\mathcal{T}|$) and the total number of patterns obtained by

			literature						MIM				MIM + Preproc. 1 and 2			
			$\mathcal{N}$		$\mathcal{B}$		$\mathcal{T}$		$\min \sum \lvert\mathcal{M}_i\rvert$		$\min \lvert\mathcal{M}\rvert$		$\min \sum \lvert\mathcal{M}_i\rvert$		$\min \lvert\mathcal{M}\rvert$	
inst.	n	W	$\sum \lvert\mathcal{N}_i\rvert$	$\lvert\mathcal{N}\rvert$	$\sum \lvert\mathcal{B}_i\rvert$	$\lvert\mathcal{B}\rvert$	$\sum \lvert\mathcal{T}_i\rvert$	$\lvert\mathcal{T}\rvert$	$\sum \lvert\mathcal{M}_i\rvert$	$\lvert\mathcal{M}\rvert$	$\sum \lvert\mathcal{M}_i\rvert$	$\lvert\mathcal{M}\rvert$	$\sum \lvert\mathcal{M}_i\rvert$	$\lvert\mathcal{M}\rvert$	$\sum \lvert\mathcal{M}_i\rvert$	$\lvert\mathcal{M}\rvert$
ngcut01	10	10	36	6	36	6	33	5	33	5	33	5	31	5	31	5
ngcut02	17	10	108	10	106	10	108	10	105	10	105	10	98	10	98	10
ngcut03	21	10	162	10	162	10	162	10	162	10	162	10	157	10	157	10
ngcut04	7	10	62	10	61	10	62	10	60	10	60	10	50	9	50	9
ngcut05	14	10	114	10	113	10	114	10	112	10	112	10	107	10	107	10
ngcut06	15	10	99	10	98	10	99	10	97	10	97	10	78	9	78	9
ngcut07	8	20	131	20	119	20	131	20	119	20	119	20	106	20	106	20
ngcut08	13	20	214	20	211	20	214	20	208	20	208	20	188	19	188	19
ngcut09	18	20	245	18	242	18	245	18	221	18	221	18	197	17	197	17
ngcut10	13	30	169	20	146	19	139	16	123	17	123	17	102	13	102	13
ngcut11	15	30	290	27	282	27	287	26	250	26	250	26	224	25	224	25
ngcut12	22	30	329	30	315	30	329	30	305	30	305	30	284	30	284	30
cgcut01	16	10	124	10	123	10	124	10	122	10	122	10	102	9	102	9
cgcut02	23	70	963	50	961	50	851	42	665	42	665	42	637	42	637	42
cgcut03	62	70	1730	45	1715	45	1671	40	932	40	932	40	885	40	885	40
gcut01	10	250	24	8	22	8	20	4	17	8	18	4	11	3	11	3
gcut02	20	250	431	47	391	47	265	23	131	47	210	25	79	21	79	21
gcut03	30	250	505	40	480	40	408	25	111	40	304	25	85	22	85	22
gcut04	50	250	1797	80	1757	80	1400	48	431	80	1051	50	295	46	295	46
gcut05	10	500	145	22	115	22	72	10	43	22	61	16	28	11	28	11
gcut06	20	500	310	37	281	37	197	17	79	37	139	21	59	16	61	14
gcut07	30	500	338	32	322	27	254	18	69	32	79	23	51	17	52	16
gcut08	50	500	2973	132	2921	132	1710	58	641	132	1236	58	386	58	397	57
gcut09	10	1000	67	10	60	10	38	6	19	10	19	10	17	5	17	5
gcut10	20	1000	342	48	304	46	182	17	99	48	146	21	64	16	64	16
gcut11	30	1000	1312	103	1222	103	610	35	257	103	439	39	143	32	143	32
gcut12	50	1000	1731	114	1643	114	1049	41	288	114	820	41	200	40	200	40
gcut13	32	3000	50286	2227	49580	2227	44338	1684	30120	1691	30682	1684	25129	1566	25129	1566
avg			2322.8	114.1	2278.1	113.9	1968.3	80.8	1279.3	94.4	1382.8	82.0	1064.0	75.8	1064.5	75.6
% reductions					-5%	-1%	-17%	-25%	-38%	-3%	-31%	-21%	-47%	-29%	-47%	-30%

Table 1: Computational evaluation (% reductions evaluated with respect to $\mathcal{N}$)

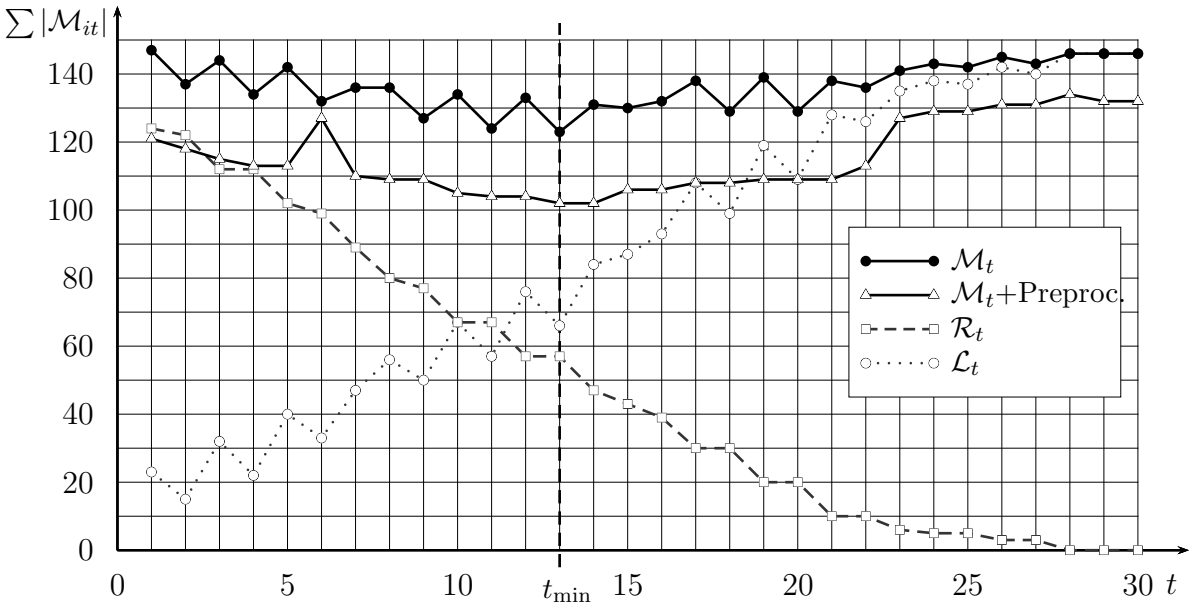

Figure 3: MIM patterns for different values of t on instance ngcut10.

summing the patterns of all items (namely, $\sum |\mathcal{N}_i|$, $\sum |\mathcal{B}_i|$, and $\sum |\mathcal{T}_i|$). In details, the set $\mathcal{N}_i$ is the subset of $\mathcal{N}$ that can be used to pack item i, and is computed as $\mathcal{N}_i = \{p \in \mathcal{N} : p \leq W - w_i\}$. Similarly, $\mathcal{T}_i = \{p \in \mathcal{T} : p \leq W - w_i\}$. For the MIM patterns we tested both versions in which either $\sum_{i \in I} |\mathcal{M}_{is}|$ or $|\mathcal{M}_s|$ is minimized in (9). In the last line we show the percentage reductions of a set, say $\mathcal{X}$, with respect to the normal patterns, computed as $100(\sum |\mathcal{X}_i| - \sum |\mathcal{N}_i|)/\sum |\mathcal{N}_i|$ and $100(|\mathcal{X}| - |\mathcal{N}|)/|\mathcal{N}|$.

Among the methods in the literature, the raster points provide on average the best values and are particularly effective on the gcut instances. The MIM patterns always provide equivalent or larger reductions than those by the literature, with a single exception on instance gcut05 where $|\mathcal{M}| > |\mathcal{T}|$. They are particularly effective for large values of bin width: the reduction that they achieve is quite limited for the ngcut instances, becomes larger for the cgcut ones, and is very relevant for the gcut ones. In particular, for 10 out of 13 gcut instances the reduction in terms of total number of patterns is higher than 70%.

In Figure 3 we show the evolution of the MIM patterns for instance ngcut10 under different threshold values t. For the rightmost value, we have $\sum |\mathcal{B}_i| = \sum |\mathcal{M}_{i,30}|$=146 (recall Proposition 2). The number of left patterns increases when t increases, the opposite happens for the number of right patterns, and in $t_{\min} = 13$ their sum achieves the minimum value 123. The preprocessing techniques always decrease the number of MIM patterns, and also lead to a minimum in $t_{\min} = 13$ (of value 102). This is a typical behavior noticed for many instances.

3 Application I: Bin Packing and Cutting Stock Problem

The *bin packing problem* (BPP) requires to cut/pack a set of n one-dimensional items, each having width w_i, from/into the minimum number of identical bins of capacity W. The *cutting stock problem* (CSP) is the BPP version in which all items having the same width are grouped together. The CSP input consists of m item types, where each type i has width w_i and number copies (demand) equal to d_i (and $n = \sum_{i=1}^{m} d_i$). Branch-and-Price algorithms are the most powerful technique to solve the BPP and the CSP, but in recent years, thanks also to the progress of commercial MILP solvers, pseudo-polynomial formulations became a valid alternative, see Delorme et al. (2016, forthcoming).

To the best of our knowledge, the most famous among these formulations is the *arc-flow* by Valério de Carvalho (1999), which explicitly refers to the CSP. Let $G = (V, A)$ be a digraph where $V = \{0, 1, \ldots, W\}$ is the set of vertices representing all partial bin fillings, and A is the set of arcs (p, q) representing either (i) the packing of an item of width $q - p$ starting from the partial bin filling p ("item arc"), or (ii) an empty portion of the bin between fillings p and q ("loss arc"). By introducing x_{pq} as an integer variable giving the number of times arc $(p, q) \in A$ is selected, and defining $\delta^-(q)$ (resp. $\delta^+(q)$) the set of arcs entering (resp. leaving) a vertex q, the CSP can be modeled as

$$\min \quad z \tag{11}$$

$$\text{s.t.} \quad \sum_{(q,r)\in\delta^+(q)} x_{qr} - \sum_{(p,q)\in\delta^-(q)} x_{pq} = \begin{cases} z & \text{if } q = 0, \\ -z & \text{if } q = W, \\ 0 & \text{if } q = 1, 2, \ldots, W-1, \end{cases} \tag{12}$$

$$\sum_{(q,q+w_i)\in A} x_{q,q+w_i} \geq d_i \qquad i = 1, 2, \ldots, m, \tag{13}$$

$$x_{pq} \geq 0, \text{ integer} \qquad (p, q) \in A. \tag{14}$$

Constraints (12) impose flow conservation, whereas constraints (13) state that all item demands must be fulfilled. Each possible packing of a bin is thus represented by a path from 0 to W in the digraph, and the aim is to minimize the number of selected paths.

The computational behavior of model (11)–(14) strictly depends on the set of arcs, that should guarantee optimality but at the same time be as small as possible. To this aim, Valério de Carvalho (1999) preliminary sorted items according to non-increasing width, and then created only those item arcs that fulfilled this sorting. In this way, the largest item can only start in zero, the second largest item can start in zero or right after the largest one, and so on (clearly, this would not preserve optimality for problems where items have two or more dimensions). He then imposed that loss arcs cannot be used before item arcs, and he created only unit-width loss arcs in the interval $[w_{\min}, w_{\min} + 1, \ldots, W]$. In the following we call this the *normal* arc-flow formulation.

Here we show how further improvements can be obtained with the MIM principle. Suppose again that items are sorted by non-increasing width. Moreover, for any $i = 1, 2, \ldots, m$, let $\bar{d}^i_j = d_j$ for all $j = 1, 2, \ldots, i-1$ and $\bar{d}^i_i = d_i - 1$.

Proposition 8 *In the normal arc-flow formulation the set of patterns where an item i can be packed (i.e., partial bin fillings where an item arc can start) is given by*

$$\mathcal{B}'_i = \left\{ x = \sum\nolimits_{j=1}^{i} w_j \xi_j : 0 \leq x \leq W - w_i, \xi_j \in \{0, 1, \ldots, \bar{d}_j^i\} \text{ for } j = 1, 2, \ldots, i \right\}. \tag{15}$$

In practice, $\mathcal{B}'_i$ is the subset of $\mathcal{B}_i$ (recall that items having the same width are grouped together in item types in the CSP notation) which is induced by the adopted ordering: a copy of item type i can have its lowest corner in a pattern created by combinations of the previous items and of the first $d_i - 1$ copies of i.

In our implementation, we take advantage of the item ordering to obtain a simple computation of $\mathcal{B}' = \cup_i \mathcal{B}'_i$. Indeed, it is enough to run a modified version of Algorithm 1 that provides all $\mathcal{B}'_i$ sets in just one call. Details are provided in the electronic companion to this paper. We also introduce two small reductions with respect to Valério de Carvalho (1999): in all our formulations we remove the original unit-width loss arcs and introduce only loss arcs that connect pairs of consecutive vertices in $\mathcal{B}'$; we remove a loss arc connecting two vertices if there is an item arc connecting the same two vertices (this is possible because of the "$\geq$" in (13)).

Similarly to what seen for the regular normal patterns in (15), also the MIM patterns may be restated by considering the item sorting. We can formally define the left and right patterns for the CSP as

$$\mathcal{L}'_{it} = \left\{ x = \sum\nolimits_{j=1}^{i} w_j \xi_j : 0 \leq x \leq \min\{t - 1, W - w_i\}, \xi_j \in \{0, 1, \ldots, \bar{d}_j^i\} \text{ for } j = 1, 2, \ldots, i \right\}, \tag{16}$$

$$\mathcal{R}'_{it} = \left\{ W - w_i - x : x = \sum\nolimits_{j=1}^{i} w_j \xi_j, 0 \leq x \leq W - w_i - t, \xi_j \in \{0, 1, \ldots, \bar{d}_j^i\} \text{ for } j = 1, 2, \ldots, i \right\}, \tag{17}$$

for $i \in I$, and then obtain the minimal set $\mathcal{M}'$ of MIM patterns by following (7), (8), and (9). In our implementation, we compute $\mathcal{M}'$ by using a modified version of Algorithm 3 that takes advantage of the item sorting. Also this algorithm is provided in the electronic companion. Once the minimal set has been obtained, we use it to build a reduced set A of arcs by considering, for each item i, only those item arcs that start in a MIM pattern. We obtain a small reduction by imposing each item of width larger than $W/2$ to have its lowest corner in 0. A further reduction is obtained by applying Preprocessing 2. We disregard instead Preprocessing 1 because it is incompatible with the adopted non-increasing width sorting.

An example of the graphs associated to the three arc-flow formulations (normal, MIM-based, and MIM-based plus Preprocessing 2) is given in Figure 4. It refers to a CSP instance with $w = (6, 5, 3, 2)$, $d = (1, 1, 2, 2)$, and $W = 8$. Figure 4-(a) presents the normal arc-flow formulation, which contains 10 item arcs (depicted in straight lines) and 6 loss arcs (dotted lines). Figure 4-(b) gives the MIM-based formulation computed for $t = 2$, which contains 4 left arcs (straight lines), 5 right arcs (dashed lines), and 4 loss arcs (dotted lines). Figure 4-(c) shows the further reduction obtained by Preprocessing 2: item arc (4,6) is enlarged to (3,6) and then removed because dominated by (3,5); consequently, the two loss arcs (3,4) and (4,5) are also removed.

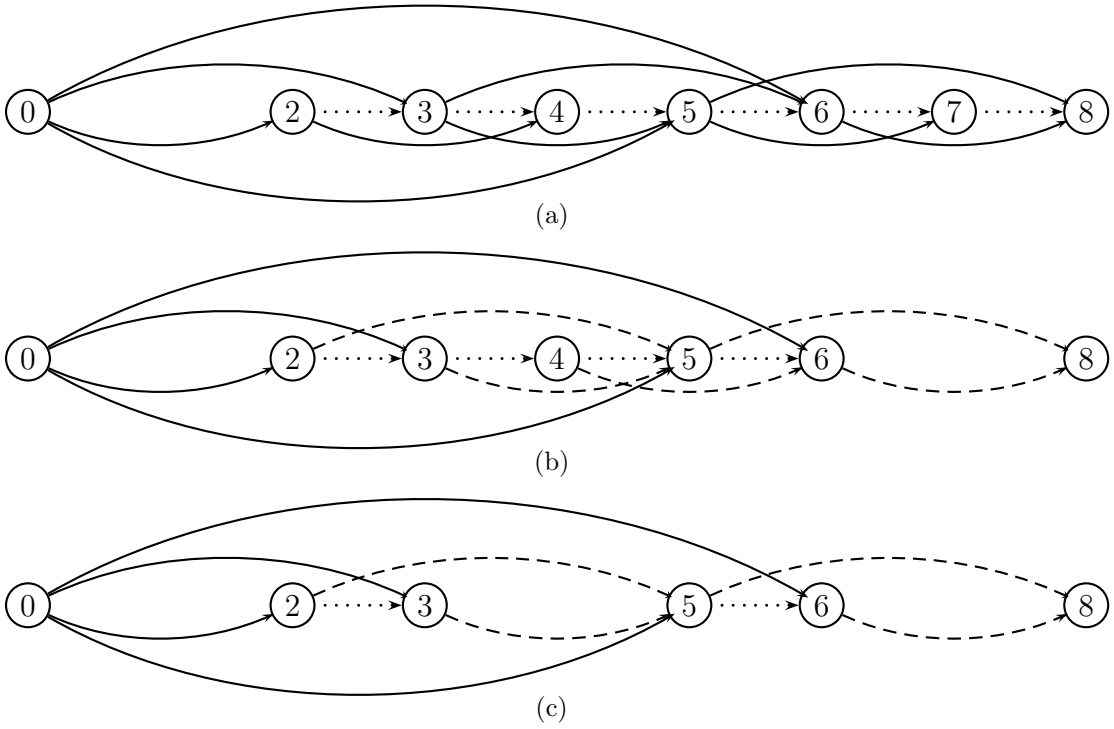

Figure 4: (a) normal arc-flow; (b) MIM-based arc-flow ($t = 2$); (c) MIM-based + Preprocessing 2.

The three arc-flow formulations have been tested on the classical CSP and BPP benchmark sets, with a time limit of 1200 seconds per instance (refer to Delorme et al. 2016, forthcoming for details on the benchmark sets). Table 2 shows the results that we obtained. The table first reports the name of the set, the number of instances (#inst.), the average number of items ($\overline{n}$), and the average bin width ($\overline{W}$). Then, for each formulation, it reports the average number of arcs ($\overline{|A|}$), the total number of instances unsolved to proven optimality (#fails), and the average number of elapsed seconds (sec). The minimum values of #fails for each group of instances are highlighted in bold. The last line reports overall total numbers of instances and fails, and overall average numbers of arcs and seconds. The arc-flow using the MIM patterns needs roughly 50% less arcs than the normal one, and it is more efficient. This behavior is particularly evident for the set Scholl 3: the average number of arcs decreases from about 1.5 million arcs to about 50.000; the MIM-based formulation solves all instances to proven optimality while the normal one could not solve any. Further reduction on the number of arcs is obtained by the preprocessing. Overall the MIM-based formulation solves to proven optimality 48 more instances than the normal one and requires a smaller average computational time. The formulation using the preprocessing technique solves 4 more instances to proven optimality with a similar computational effort.

				normal arc-flow			arc-flow + MIM			arc-flow + MIM and Preprocessing 2		
set	#inst.	$\overline{n}$	$\overline{W}$	$\overline{\lvert A\rvert}$	#fails	sec	$\overline{\lvert A\rvert}$	#fails	sec	$\overline{\lvert A\rvert}$	#fails	sec
AI 201	50	171	2220	121388	8	449.5	85464	10	411.6	85445	**7**	390.9
ANI 201	50	170	2220	119635	35	963.4	83905	**22**	733.5	83883	23	746.2
Falkenauer T	80	118	1000	16380	**0**	4.4	4836	**0**	1.6	3279	**0**	2.1
Falkenauer U	80	76	150	3024	**0**	0.2	1488	**0**	0.1	1487	**0**	0.1
Hard28	28	162	1000	36837	**0**	96.0	21633	**0**	61.9	21599	**0**	34.7
Scholl 1	720	64	123	1740	**0**	0.1	941	**0**	0.1	928	**0**	0.1
Scholl 2	480	98	1000	39373	50	225.7	26123	24	157.5	25955	**23**	162.0
Scholl 3	10	199	100000	1560847	10	1208.2	52094	**0**	236.3	33790	**0**	166.6
Schwerin 1	100	44	1000	11798	**0**	3.3	5353	**0**	1.0	4621	**0**	0.9
Schwerin 2	100	46	1000	12601	**0**	3.4	5760	**0**	1.1	4979	**0**	0.9
Waescher	17	50	10000	175462	12	932.9	104014	11	831.8	103480	**10**	820.2
tot/avg	1715	83	1330	32547	115	122.9	15276	67	88.3	14948	**63**	88.4

Table 2: Impact of the MIM patterns on standard CSP instances.

4 Application II: Non-Exact Two-Stage Cutting Stock Problem

In this section we solve the *non-exact two-stage guillotine cutting stock problem* (2S-CSP), which is the generalization of the CSP of Section 3 in which: (i) items and bins are two-dimensional rectangles; (ii) items can be produced from the bins by a series of successive *guillotine* cuts, that is, cuts that traverse entirely the bin (or the residual bin portion under processing) from one edge to the other; (iii) just two series of cuts can be used; and (iv) a final trimming stage is possibly adopted to remove waste. In practice each bin is first cut down along its height into horizontal strips (1st stage), these slices are then cut vertically across their widths (2nd stage), and then, if the obtained items are higher than the required ones, a final horizontal cutting stage is adopted for removing the waste.

The 2S-CSP has been introduced in the 1960s by Gilmore and Gomory (1965), and has attracted the interest of many researchers and practitioners in the following decades because it can naturally model cutting problems that arise in many production industries such as steel, wood, glass, etc. The problem has been solved with several optimization techniques, including MILP models by Lodi et al. (2004), arc-flow formulations by Macedo et al. (2010) and Silva et al. (2010), and branch-and-price algorithms by Alves et al. (2009) and Mrad et al. (2013).

Here we apply the MIM patterns to the formulation by Macedo et al. (2010), which we briefly recall. To ease notation, we denote the height of the items by h_i instead of w_i^2 and that of the bins by H instead of W^2. Let m^* be the number of different item heights and $\{h_1^*, h_2^*, \ldots, h_{m^*}^*\}$ the corresponding set. We create m^*+1 digraphs $G^0, G^1, \ldots, G^{m^*}$ as follows. $G^0 = (V^0, A^0)$ is a standard digraph associated to the 1st stage cut, where $V^0 = \{0, 1, \ldots, H\}$ and A^0 is the set of arcs (a, b) representing either the cutting of a strip of height $b-a$ starting at height a, or an empty portion of a bin between heights a and b. $G^s = (V^s, A^s)$ is instead a multi-digraph associated to

a 2nd stage cut on a strip s of height h^*_s, for $s \in \{1, 2, \dots, m^*\}$, where $V^s = \{0, 1, \dots, W\}$ and A^s contains arcs (d, e, i) of two types: for $i \in I$, arc $(d, d + w_i, i)$ refers to the cut of an item i starting at width d; for $i = 0$, arc $(d, e, 0)$ is a loss arc representing an empty portion in the strip between widths d and e. Let also $A^s(i) \subseteq A^s$ define the subset of arcs referring to a given item $i \in I$.

Following Macedo et al. (2010), the set A_0 is constructed by creating all patterns where an item can start, as in (15), but replacing widths with heights and preliminary sorting items according to non-increasing height. Sets A_s are built using the same principle, but imposing that only arcs referring to items i of height $h_i = h^*_s$ can start from vertex 0. In terms of decision variables, let z_0 be the number of used bins, z_s the number of adopted strips of height h^*_s, y_{ab} the number of times in which arc (a, b) is used for a 1st stage cut, and x^s_{dei} the number of times in which arc (d, e, i) is used for a 2nd stage cut on a strip s. The 2S-CSP is thus:

$$\min \quad z^0 \tag{18}$$

$$\text{s.t.} \; -\sum_{(a,b)\in\delta^-(b)} y_{ab} + \sum_{(b,c)\in\delta^+(b)} y_{bc} = \begin{cases} z^0 & \text{if } b = 0, \\ -z^0 & \text{if } b = H, \\ 0 & \text{if } b = 1, 2, \dots, H-1 \end{cases} \tag{19}$$

$$\sum_{(a,a+h^*_s)\in A^0} y_{a,a+h^*_s} = z^s \qquad s = 1, 2, \dots, m^*, \tag{20}$$

$$-\sum_{(d,e,i)\in\delta^-(e)} x^s_{dei} + \sum_{(e,f,i)\in\delta^+(e)} x^s_{efi} = \begin{cases} z^s & \text{if } e = 0, \\ -z^s & \text{if } e = W, \\ 0 & \text{if } e = 1, 2, \dots, W-1 \end{cases} \qquad s = 1, 2, \dots, m^*, \tag{21}$$

$$\sum_{s=1,2,\dots,m^*} \sum_{(d,d+w_i,i)\in A^s(i)} x^s_{d,d+w_i,i} \geq d_i \quad i \in I, \tag{22}$$

$$z_s \geq 0, \text{ integer} \qquad s = 0, 1, \dots, m^*, \tag{23}$$

$$y_{ab} \geq 0, \text{ integer} \qquad (a, b) \in A^0, \tag{24}$$

$$x^s_{dei} \geq 0, \text{ integer} \qquad s = 1, 2, \dots, m^*, (d, e, i) \in A^s. \tag{25}$$

Flow conservation is imposed for the first stage by constraints (19) and for the 2nd stage by constraints (21). Constraints (20) link together the y and z variables, and (22) impose that all demands are fulfilled.

We call *normal* this arc-flow formulation. Similarly to what was done for the CSP, also for the 2S-CSP we improve the normal formulation by replacing the regular normal patterns with the MIM, and then with the MIM plus Preprocessing 2. The way in which we impose these modifications follow the footsteps of what done in Section 3 for the CSP, but uses the item orderings suggested by Macedo et al. (2010).

The results of our implementations are shown in Table 3. The columns have the same meanings as those in Table 2. The formulations have been tested on the two publicly available benchmark sets

for the 2S-CSP, A and ATP, attempting the case in which the 1st stage cut is along the height (A(H) and ATP(H)) or along the width (A(W) and ATP(W)). Following previous works in the literature, each model was allowed a time limit of 7200 seconds. The MIM-based formulation improves the normal one by reducing the average number of arcs (to almost one third) and the average computational effort, finding 5 additional proven optimal solutions. The use of Preprocessing 2 reduces the number of arcs by an additional 5%, but does not help improving the number of proven optimal solutions, as the number of fails increases from 16 to 17. We believe this fact may be imputed to a worsening of the behavior of the automatic Cplex heuristics, that fail in finding a good feasible solutions for two instances (for which a quick solution was instead found when Preprocessing 2 was not used).

In Table 4 we compare our best algorithm with those in the literature, that we call for short MMH for Mrad et al. (2013), MAV for Macedo et al. (2010), LMV for Lodi et al. (2004), AMM for Alves et al. (2009), and SAV for Silva et al. (2010). The MIM-based arc-flow formulation has a smaller number of fails than the other algorithms. Algorithm MAV is also based on the normal arc-flow formulation, but it includes a number of improvements and is very competitive as it solves all instances of the set A(H) to proven optimality. Still the MIM-based arc-flow formulation is quite faster than it.

				normal arc-flow			arc-flow + MIM			arc-flow + MIM and Preprocessing 2		
set	#inst.	$\overline{n}$	$\overline{W}$	$\overline{\lvert A\rvert}$	#fails	sec	$\overline{\lvert A\rvert}$	#fails	sec	$\overline{\lvert A\rvert}$	#fails	sec
A(H)	43	29	2599	23756	**0**	59.1	6659	**0**	4.8	5436	**0**	10.2
A(W)	43	29	1874	138816	6	1177.6	39410	5	947.3	36889	**4**	936.3
ATP(H)	20	40	565	77994	5	2681.1	38411	**4**	1637.0	37839	5	1999.8
ATP(W)	20	40	494	76594	10	3666.6	35894	**7**	3057.3	35273	8	3132.4
tot/Avg	126	32	1695	80019	21	1429.6	27516	**16**	1070.1	26049	17	1137.6

Table 3: Impact of the MIM patterns on the 2S-CSP.

		MMH$^{(1)}$		MAV$^{(2)}$		LMV$^{(2)}$		AMM$^{(2)}$		SAV$^{(2)}$		arc-flow+MIM	
set	#inst.	#fails	sec	#fails	sec	#fails	sec	#fails	sec	#fails	sec	#fails	sec
A(H)	43	1	277.2	**0**	377.9	21	3519.4	10	1854.3	2	735.0	**0**	4.8
A(W)	43	6	1014.2									**5**	947.3
ATP(H)	20									8	3642.6	**4**	1637.0
ATP(W)	20											**7**	3057.3

(1) = Pentium IV 2.2 GHz with 4 GB RAM; (2) =1.87 GHz Intel Core Duo with 2 GB RAM.

Table 4: Comparison with the 2S-CSP literature.

5 Application III: Two-Dimensional Orthogonal Packing Problem

The *two-dimensional orthogonal packing problem* (2OPP) is the basic feasibility test of determining whether a set I of rectangular items fits or not into a rectangular bin. Rotation of the items is not allowed. The 2OPP arises as a subproblem in many two-dimensional C&P problems, such as knapsack, bin packing, and strip packing. It has been tackled with several algorithms, including, e.g., constraint programming techniques by Clautiaux et al. (2008), and mixed methods by Mesyagutov et al. (2012) and Belov and Rohling (2013). Here we solve it first by means of branch-and-bound algorithms and then by primal decomposition techniques. Once again we ease notation by writing h_i instead of w_i^2 and H instead of W^2 (recall the general problem definition in Section 2).

5.1 Combinatorial Branch-and-Bound Algorithms

Combinatorial *branch-and-bound* (B&B) algorithms for C&P attempt to build solutions by packing one item at a time in the bin, according to a specific enumeration rule. Here we use a basic B&B that builds upon the rule by Boschetti and Montaletti (2010). We start from the empty bin and pack items one at a time from the bottom to the top. At a given partial packing, let the *skyline* represent the set of the top borders of the items (or bottom of the bin where no item has been packed yet), that is, a set of consecutive horizontal segments positioned at different y-coordinates. Let the *niche* be the segment of the skyline positioned at the smallest y-coordinate (breaking ties by smallest x-coordinate). The borders of the niche are the vertical segments at its left and right. For example, in Figure 1-(b) the niche is the horizontal segment [17, 20] at the y-coordinate 0, its left border is the vertical segment [0, 4], and its right border is the vertical segment [0, 20].

Let p be the x-coordinate of the leftmost position of the current niche. Note that $p \in \mathcal{B}$ because it is a feasible combination of item widths. We attempt packing in p any item i that can fit and for which the condition $p \in \mathcal{B}_i$ is satisfied. We select items by non-increasing order of width, and create a new node in the enumeration tree for each resulting packing. We also create a last additional node, that we call *loss node*, in which no item is packed in p. Let $\hat{h}$ be the minimum height among the heights of left and right borders of the current niche and the heights of the residual items to be packed. When a loss node is explored, the portion of the niche going from p to the successive pattern in $\mathcal{B}$ and having height $\hat{h}$ is closed and considered unavailable for packing. To this aim the set $\mathcal{B}$ is re-evaluated at every node by taking in consideration only the items that still have to be packed. The next pattern in $\mathcal{B}$ inside the niche is selected, if any, and the process is iterated, from left to right. When no such pattern exists, the current niche is closed and the next niche is computed. The tree is explored in depth-first fashion. Nodes are fathomed by the use of a simple estimation of the area of the residual items to be packed and or the residual bin area (continuous bound), and by the so-called DP-cuts of Kenmochi et al. (2009).

Many improvements can be done to this simple scheme (preprocessing techniques, improved

computations of $\hat{h}$, techniques to fathom nodes, ...), but this is out of the scope of this paper. Here, similarly to what was done in the previous sections, we call *normal* this basic B&B technique and attempt to improve it by the use of the MIM patterns. The first improved version is the one in which $\mathcal{B}$ is replaced by $\mathcal{M}$. This reduces the number of nodes because: (i) only items i for which $p \in \mathcal{M}_i$ are selected for packing in p, and (ii) the distance between two consecutive patterns in $\mathcal{M}$ is typically larger than in $\mathcal{B}$, and hence the area made unavailable when a loss node is selected is larger. The second version makes use also of Preprocessings 1 and 2. The last improved version also changes the way in which the tree is explored, by including a new branching scheme (*MIM-branch*). Let $t_{\min}$ be the threshold value used to build $\mathcal{M}$ (see Section 2). If $p < t_{\min}$ then we proceed from left to right in the selection of the positions in the niche as done in the previous branching schemes, otherwise we proceed from right to the left. In other words, if $p \geq t_{\min}$ then we select for packing the first position q from the right, and pack there all items i for which $q \in \mathcal{M}_i$.

A computational evaluation of these four techniques is proposed in Table 5. Each algorithm was run with a time limit of 900 seconds. We selected the well-known 2OPP benchmark instances, namely sets E, C, N, and T, plus the two sets of instances created by Mesyagutov et al. (2012), that we call MSB-450 and MSB-630. We refer to Mesyagutov et al. (2012) for details on all benchmark sets. The column headings are the same as in the previous tables, with the exception of "nodes", that report the number, in millions, of explored nodes.

				normal B&B			B&B +MIM			B&B + MIM and Preproc. 1-2			B&B+MIM, Preproc. 1-2, MIM-branch		
set	#inst	$\overline{n}$	$\overline{W}$	#fails	sec	nodes	#fails	sec	nodes	#fails	sec	nodes	#fails	sec	nodes
MSB-450	450	20	100	58	130.8	139.2	38	91.183	96.1	37	91.7	97.5	**35**	78.2	64.2
MSB-630	630	20	1000	208	301.8	59.4	191	281.18	60.1	191	279.6	61.1	**183**	264.2	43.8
E	42	15.9	20	**22**	491.0	68.7	**22**	490.99	68.8	**22**	493.0	68.1	**22**	508.9	64.5
C	21	69.3	68.6	**13**	621.3	37.3	**13**	618.14	37.1	**13**	598.7	30.7	**13**	611.1	23.8
N	35	69.6	200	**21**	566.8	9.1	**21**	565.7	8.9	22	598.3	9.4	22	601.9	7.1
T	35	69.9	200	**23**	600.3	12.1	**23**	600.07	12.2	**23**	616.9	11.4	**23**	625.4	8.4
tot/Avg	1213	23.6	570	345	266.7	86.1	308	241.2	70.5	308	241.7	71.4	**298**	229.8	49.7

Table 5: Evaluation of different B&B algorithms on 2OPP instances (nodes = 10^6 explored nodes).

The normal B&B fails in providing a certified proof of feasibility or infeasibility for 345 out of 1213 instances. The effect of the MIM is not relevant for sets E, C, N, and T, but it is quite remarkable for the MSB sets, where #fails decrease by 37 units. The use of the new branching scheme is effective, because it allows the algorithm to solve 10 more instances and obtain a strong decrease in the number of explored nodes.

5.2 Primal Decomposition Methods

Côté et al. (2014a) solved the strip packing problem by iteratively calling an inner model to test the feasibility of 2OPP instances. Here we describe their model for the 2OPP, which is based on a

primal decomposition with combinatorial Benders cuts. The decomposition first take cares of the horizontal positions of the items, and makes use of a binary variable x_{ip} taking the value 1 if item i is packed in pattern p along the x-axis, 0 otherwise. Let $\mathcal{B}_{i,q}$ denote the subset of patterns for which item i occupies position q, formally $\mathcal{B}_{i,q} = \{p \in \mathcal{B}_i : q - w_i + 1 \leq p \leq q\}$. Then the 2OPP can be modeled as the following integer linear feasibility test:

$$\sum_{p \in \mathcal{B}_i} x_{ip} = 1 \qquad i \in I, \tag{26}$$

$$\sum_{i \in I} \sum_{p \in \mathcal{B}_{i,q}} h_i x_{ip} \leq H \qquad q \in \mathcal{B}, \tag{27}$$

$$\sum_{i \in I} x_{i,p_i^s} \leq n - 1 \qquad \forall\ s \text{ infeasible for the SP}, \tag{28}$$

$$x_{ip} \in \{0, 1\} \qquad i \in I, p \in \mathcal{B}_i. \tag{29}$$

Constraint (26) impose that each item is packed once. Constraints (27) state that the sum of the item heights on a certain pattern q does not exceed the bin height. Before discussing constraints (28), let us focus on the sub-model induced by (26)–(27) and (29). Solving this sub-model requires to find an allocation for unit-width slices of the items into the bin, in such a way that all slices are contiguous one with the other. This problem (known in the literature as the *bar relaxation* or as the *bin packing problem with contiguity constraints*) corresponds to the first *master problem* (MP) of the proposed decomposition. Suppose a solution s for the MP is found, in which any item $i \in I$ is packed in a pattern p_i^s. Then the *slave problem* (SP) is to determine the set of vertical positions for all the items that lead a packing without overlapping, if any. If such a set is found, then the model returns a feasible 2OPP solution, otherwise a feasibility cut (28) is added to the MP to disregard solution s. The approach works well when the simple feasibility cuts are improved into the much stronger *lifted combinatorial Benders cuts*, as discussed in Côté et al. (2014a).

Here we solve the problem by using the same algorithm in Côté et al. (2014a): the MP is solved with Cplex, the SP with a dedicated branch-and-bound (Section 3 of their article), and the feasibility cuts are improved with greedy procedures and a lifting based on linear programming (Section 4 of their article). The only difference is that we replace the regular patterns $\mathcal{B}$ with the MIM patterns $\mathcal{M}$ in model (26)-(29).

The results that we obtained are presented in Table 6. The column headings are the same used for the previous tables, with the addition of "var", that denotes the average number of x_{ip} variables in the different mathematical models. Each algorithm was run with a time limit of 900 seconds. The normal decomposition fails for 181 instances. The MIM patterns are effective in reducing the number of variables and thus allow the algorithm to close 25 more instances. The two preprocessing techniques are not effective on sets N and T. This probably happens because these sets are composed entirely by 2OPP feasible instances with zero waste ("perfect packing" instances), that have been created mostly with the aim of testing heuristic algorithms, and, as noticed in Section 4, preprocessing may have a slight negative influence on the automatic Cplex

				normal decomposition			decomposition + MIM			decomposition + MIM and Preproc. 1-2		
set	#inst	$\overline{n}$	$\overline{W}$	#fails	sec	var	#fails	sec	var	#fails	sec	var
MSB-450	450	20	100	48	125.7	7308	40	94.7	4770	**38**	95.4	4054
MSB-630	630	20	1000	61	117.2	7191	47	88.8	4919	**44**	83.6	4163
E	42	16	20	**0**	0.1	202	**0**	0.1	184	**0**	0.1	173
C	21	69	69	**14**	600.9	6258	**14**	602.4	6235	**14**	601.1	6203
N	35	70	200	30	772.8	12136	**28**	729.8	11885	29	749.8	11726
T	35	70	200	28	724.7	12197	**27**	699.9	11959	29	746.9	11802
tot/avg	1213	24	570	181	161.2	7263	156	133.0	5127	**154**	132.4	4458

Table 6: Evaluation of different decomposition algorithms on 2OPP instances.

heuristics. The preprocessing techniques provide instead positive improvements on the two MSB sets, where they allow the decomposition method to find 5 more proven optimal solutions.

6 Conclusions

In this paper we proposed a principle to reduce the number of patterns in multi-dimensional *cutting and packing* (C&P) problems. It consists of a new set of patterns, called *meet-in-the-middle* (MIM), obtained by aligning items along each dimension either to the bottom of the bin or to the top of it. The computation of the MIM patterns does not require additional effort with respect to previous methods in the literature and usually leads to a smaller number of patterns. Further reduction criteria can also be applied. Extensive computational tests showed the efficiency of the proposed techniques on a number of relevant C&P problems.

The MIM principle can be used in several optimization algorithms, because it usually reduces the number of variables required by mathematical models and the number of nodes explored by combinatorial branch-and-bound algorithms. The principle can be adapted to a large number of applications, not only in C&P but also in other combinatorial optimization fields, such as vehicle routing and scheduling. There is thus a large number of possible future research applications.

Acknowledgment

The second author acknowledges financial support by CAPES/Brazil under grant PVE n° A007/2013.

References

Alvarez-Valdes, R., F. Parreño, J.M. Tamarit. 2005. A branch-and-cut algorithm for the pallet loading problem. *Computers & Operations Research* **32** 3007–3029.

Alves, C., R. Macedo, M. Mrad, J.M. Valério de Carvalho, F. Alvelos, T.M. Chan, E. Silva. 2009. An exact branch-and-price algorithm for the two-dimensional cutting stock problem. Working paper.

Beasley, J. E. 1985a. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society* **36** 297–306.

Beasley, J. E. 1985b. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research* **33** 49–64.

Belov, G., H. Rohling. 2013. LP bounds in an interval-graph algorithm for orthogonal-packing feasibility. *Operations Research* **61** 483 – 497.

Bortfeldt, A., G. Wäscher. 2013. Constraints in container loading. A state of the art review. *European Journal of Operational Research* **229**(1) 1–20.

Boschetti, M.A., E. Hadjiconstantinou, A. Mingozzi. 2002. New upper bounds for the two-dimensional othogonal non guillotine cutting stock problem. *IMA Journal of Management Mathematics* **13** 95–119.

Boschetti, M.A., L. Montaletti. 2010. An exact algorithm for the two-dimensional strip-packing problem. *Operations Research* **58** 1774–1791.

Christofides, N., C. Whitlock. 1977. An algorithm for two-dimensional cutting problems. *Operations Research* **25** 30–44.

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

Côté, J.-F., M. Dell'Amico, M. Iori. 2014a. Combinatorial benders' cuts for the strip packing problem. *Operations Research* **62** 643 – 661.

Côté, J.-F., M. Gendreau, J.-Y. Potvin. 2014b. An exact algorithm for the two-dimensional orthogonal packing problem with unloading constraints. *Operations Research* **62**(5) 1126–1141.

Delorme, M., M. Iori, S. Martello. 2016, forthcoming. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research* .

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

Herz, J.C. 1972. Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development* **16** 462–469.

Horowitz, W., S. Sahni. 1974. Computing partitions with applications to the knapsack problem. *Journal of the ACM* **21** 277–292.

Iori, M., J.J. Salazar González, D. Vigo. 2007. An exact approach for the vehicle routing problem with two-dimensional loading constraints. *Transportation Science* **41** 253–264.

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

Lodi, A., S. Martello, D. Vigo. 2004. Models and bounds for two-dimensional level packing problems. *Journal of Combinatorial Optimization* **8**(1) 363–379.

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

Mesyagutov, M., G. Scheithauer, G. Belov. 2012. LP bounds in various constraint programming approaches for orthogonal packing. *Computers & Operations Research* **39** 2425 – 2438.

Mrad, M., I. Meftahi, M. Haouari. 2013. Branch-and-price algorithm for the two-stage guillotine cutting stock problem. *Journal of the Operational Research Society* **64**(5) 629–637.

Scheithauer, G., J. Terno. 1996. The g4-heuristic for the pallet loading problem. *Journal of the Operational Research Society* **47** 511–522.

Silva, E., F. Alvelos, J.M. Valério de Carvalho. 2010. An integer programming model for two- and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research* **205** 699–708.

Terno, J., R. Lindemann, G. Scheithauer. 1987. Zuschnittprobleme and ihre praktische lösung. Tech. rep., Verlag Harry Deutsch, Thun und FrankfurtMain.

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

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

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

Supplementary Algorithms

As discussed in Section 2.1, for a general cutting and packing packing problem the computation of the regular normal patterns $\mathcal{B}_i$, for any item i in the item set I, may be obtained by invoking Algorithm 1 as `NormalPatterns`$(I \setminus \{i\}; W - w_i)$. Then, the computation of the entire set is simply obtained by letting $\mathcal{B} = \cup_i \mathcal{B}_i$. As described in Section 3, the input item set I of the one-dimensional *cutting stock problem* (CSP) consists of m item types, where each type i comprises d_i items having the same width w_i. For the CSP, cutting items according to a given order (the one usually adopted is by non-increasing width) preserves optimality, and consequently the set $\mathcal{B}$ may be replaced by a smaller, or at least equivalent, set $\mathcal{B}'$ that considers such order, see Equation (15).

The computation of $\mathcal{B}'$ may be obtained by invoking Algorithm 4 below. A support array T is used to store all feasible item width combinations. T is first initialized to consider only the empty bin filling. Then, in the main loop starting at step 5, T is updated by considering all item combinations and the set of patterns is built accordingly. Note that, for each item type $i > 1$, the set $\mathcal{B}'_i$ is computed incrementally starting from the support array T obtained during the computation of the previous set $\mathcal{B}'_{i-1}$.

Algorithm 4 `BPatternsCSP`$(I; W)$

```
1: Require: I: set of sorted items, W: bin width
2: B' ← ∅
3: T ← [0 to W]: an array with all entries initialized at 0
4: T[0] ← 1
5: for i = 1 to m do
6:     B'_i ← ∅
7:     for p = W − w_i to 0 do
8:         if T[p] = 1 then
9:             for k = 1 to d_i do
10:                if p + w_i * k > W then break
11:                T[p + w_i * k] ← 1
12:                B'_i ← B'_i ∪ {p + w_i * (k − 1)}
13:            end for
14:        end if
15:    end for
16:    B' ← B' ∪ B'_i
17: end for
18: return B'
```

In a similar way, also the computation of the minimal set of MIM patterns for the CSP may benefit from the adopted item sorting. This can be done by using Algorithm 5 below, which is an adaption to the CSP of Algorithm 3 of Section 2.2. Algorithm 5 starts by computing the regular

normal patterns at step 3. Then, it fills the support arrays for the left and right patterns, T_{left} and T_{right}, respectively, at steps 4-13. Note that the check on $w_i \leq W/2$ at step 7 is used to impose that items having width larger than $W/2$ are only packed with their lowest corner in 0. The computation of the threshold value $t_{\min}$ for which $\sum_{i \in I} |\mathcal{M}'_{is}|$ is a minimum is obtained by steps 14-21. Finally, set $\mathcal{M}'$ is built at steps 22-30.

Algorithm 5 `MinimalMIMSetCSP`$(I;\ W)$

1: **Require:** I: set of *sorted* items, W: bin width
2: $T_{left},\ T_{right} \leftarrow [0 \text{ to } W]$: two arrays with all entries initialized at zero
3: $\mathcal{B}' \leftarrow$ `BPatternsCSP`$(I;\ W)$
4: **for** $i = 1$ to m **do**
5: **for** $p \in \mathcal{B}'_i$ **do**
6: $T_{left}[p] \leftarrow T_{left}[p] + 1$
7: **if** $w_i \leq W/2$ **then** $T_{right}[W - w_i - p] \leftarrow T_{right}[W - w_i - p] + 1$
8: **end for**
9: **end for**
10: **for** $p = 1$ **to** W **do**
11: $T_{left}[p] \leftarrow T_{left}[p] + T_{left}[p-1]$
12: $T_{right}[W - p] \leftarrow T_{right}[W - p] + T_{right}[W - (p-1)]$
13: **end for**
14: $t_{\min} \leftarrow 1$
15: $\min \leftarrow T_{left}[0] + T_{right}[1]$
16: **for** $p = 2$ **to** W **do**
17: **if** $T_{left}[p-1] + T_{right}[p] < \min$ **then**
18: $\min \leftarrow T_{left}[p-1] + T_{right}[p]$
19: $t_{\min} \leftarrow p$
20: **end if**
21: **end for**
22: $\mathcal{M}' \leftarrow \emptyset$
23: **for** $i = 1$ to m **do**
24: $\mathcal{M}'_i \leftarrow \emptyset$
25: **for** $p \in \mathcal{B}'_i$ **do**
26: **if** $p < t_{\min}$ **then** $\mathcal{M}'_i \leftarrow \mathcal{M}'_i \cup \{p\}$
27: **if** $(W - w_i - p \geq t_{\min}$ **and** $w_i \leq W/2)$ **then** $\mathcal{M}'_i \leftarrow \mathcal{M}'_i \cup \{W - p - w_i\}$
28: **end for**
29: $\mathcal{M}' \leftarrow \mathcal{M}' \cup \mathcal{M}'_i$
30: **end for**
31: **return** $\mathcal{M}'$