Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)

A Constraint Programming Solution to the Guillotine Rectangular Cutting Problem

Sergey Polyakovskiy[1], **Peter J. Stuckey** [2,3]

[1] School of Information Technology, Deakin University, Australia
[2] Faculty of Information Technology, Monash University, Australia
[3] OPTIMA ARC Industrial Training and Transformation Center, Melbourne Connect, Australia
sergey.polyakovskiy@deakin.edu.au, peter.stuckey@monash.edu

Abstract

The guillotine rectangular cutting problem deals with a single rectangular plate of raw material and a collection of rectangular items to be cut from the plate. Each item is associated with a profit and a demand. The problem searches for a feasible layout of a subset of items on the plate so as to maximize the total profit of selected items. The guillotine constraint restricts feasible layouts to those that can be obtained via guillotine edge-to-edge cuts that run parallel to an edge of the plate. We propose a novel constraint programming model that is suitable for guillotine cutting with an arbitrary number of stages of alternating horizontal and vertical guillotine cuts. This is an assignment-based model that models guillotine cuts using a constant number of rectangular regions, with some regions allocated to items. It treats the entire plate as a primary region and decides on the guillotine cuts required to split the regions recursively till they produce space for the items. To speed the search, the model explores the strength of cumulative scheduling relaxations of the cutting problem. Our model is a successful alternative to more traditional mixed-integer linear programming (MILP) models. It outperforms a number of state-of-the-art MILPs on a set of small and moderate size benchmark instances and proves optimality for several instances that remain challenging for these MILPs.

Introduction

The (two-dimensional) rectangular cutting problem (RCP) is essential in several real-life industrial applications that require cutting large rectangular stock objects into small rectangular items to produce parts of end products. Relevant examples include glass, plastic, or metal industries, where rectangular components have to be cut from a single sheet of raw material as to minimize waste. Similar applications emerge in the shipping and transportation industries, where rectangular packages have to be positioned on a two-dimensional surface in a way that minimizes the empty space. Variants of the RCP arise when additional constraints are imposed. Usually, such constraints reflect conditions of the arrangement of items within the object and/or technological constraints, e.g. requiring guillotine cuts, limiting the number of cuts, fixing items' orientation, and etc. (Lodi, Martello, and Vigo 1999).

This $\mathcal{NP}$-hard optimization problem is academically challenging. It has a huge solution space with many alternative solutions whose packing configurations are symmetric. Often the success of exact approaches, such as based on mixed-integer linear programming (MILP) and constraint programming (CP), is limited to only small instances unless they employ strong relaxation and symmetry breaking techniques. With their use, exact approaches may find optima for instances with up to 50 items in a reasonable time. However, they still may fail to prove optimality or even guarantee a sufficiently good convergence to a global optimum for instances with as few as 20 items. Efficient exact approaches to this problem are of vital importance and have broad applications. In fact, the RCP is a main ingredient of more complex packing and cutting problems and approaches to solve them. For example, the RCP is a (pricing) sub-problem in the column generation procedure to the two-dimensional bin packing problem; hence, the efficiency of the entire approach directly depends on the solution technique selected for the RCP (Pisinger and Sigurd 2005).

It is well understood that two-dimensional packing problems give rise to two scheduling problems as relaxations (Clautiaux et al. 2008). In application to the RCP, this paper proposes a new problem-tailored CP model and shows that taking advantage of these scheduling relaxations is invaluable in reducing the search. The rest of the paper defines the exact problem and reviews the related work, then it explains the CP model to solve the problem, analyzes the results of computational experiments, and draws a conclusion.

Problem Formulation and Related Work

The rectangular cutting problem consists in cutting a set $I = \{1, \dots, n\}$ of n distinct small rectangular items from a single large rectangular stock plate of width $\overline{W} \in \mathbb{Z}^+$ and height $\overline{H} \in \mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of strictly positive integers. An item $i \in I$ of size (w_i, h_i) is characterized by its width $w_i \in \mathbb{Z}^+$, its height $h_i \in \mathbb{Z}^+$, a profit $\pi_i \in \mathbb{Z}^+$, and a demand $d_i \in \mathbb{Z}^+$. A multiset of selected items positioned on the plate forms a cutting pattern. For a cutting pattern to be considered feasible, the layout of its items must fulfill a number of requirements: (i) the items must lie entirely inside the plate; (ii) they must be cut with their edges parallel to those of the plate; (iii) no two items may overlap. The objective of the problem is to find a feasible pattern that maximizes the total profit gained from the selected items.

When $d_i = 1$ for each item $i \in I$, the problem is classi-

fied as the two-dimensional (2D) single knapsack problem (2DSKP) (Wäscher, Haußner, and Schumann 2007). The 2DSKP is a special case of the two-dimensional single large object placement problem (2DSLOPP) that allows $d_i > 1$, $d_i \neq \infty$, for at least one item $i \in I$. The problem is said to be *unweighted* if π_i is equal or proportional to the area $w_i \times h_i$ for each item $i \in I$; otherwise, the problem is *weighted*. Thus, the unweighted problem searches for a pattern of maximal packing density defined as the ratio of the total area of selected items to the area of the plate. It is common that the weighted version is computationally more challenging than its unweighted counterpart (Silva et al. 2023).

The focus of this research is on the oriented guillotine stage-unrestricted 2DSKP and 2DSLOPP, where items have a fixed orientation and can be obtained only via guillotine edge-to-edge cuts that run parallel to an edge of the plate. Furthermore, there is no limit applied to the number of sequential stages of alternating horizontal and vertical guillotine cuts required to cut the items, with each stage being a set of cuts made on the sub-plate resulting from the cut made in the previous stage.

Both problems have been extensively studied and the recent comprehensive survey by Iori et al. (2021) outlines the existing solution techniques. Therefore, here we only review the state of the art that is most relevant to our research. The first tree search exact method for the 2DSLOPP was proposed by Christofides and Whitlock (1977) and recently revised by Velasco and Uchoa (2019), who applied dynamic programming procedures to obtain sharp upper bounds. Dyckhoff (1981) was the first to observe that cutting an item $i \in I$ of width w_i from a plate of width $\overline{W}$ produces a sub-plate of width $\overline{W} - w_i$, which then can be further (recursively) decomposed. This observation (also known as the 'cut-and-plate' principle) is commonly used by guillotine cutting approaches (including our CP solution) as it is a natural way to obtain all items via vertical and horizontal edge-to-edge cuts. For example, it is employed by the first MILP model by Furini, Malaguti, and Thomopulos (2016), which is built on a pseudo-polynomial number of variables and constraints. Their model relies on a dynamic programming procedure to determine a subset of variables used in an optimal solution. Martin, Morabito, and Munari (2021) proposed a top-down cutting approach. Their MILP model considers the cutting pattern as a binary tree, in which the root node is the object, and branches correspond to guillotine cuts. Their model appears efficient for instances with a moderate number of items in an optimal solution. Recently, Silva et al. (2023) proposed a generic Floating-Cuts MILP model for both non-guillotine and guillotine problems. Their model also explores the idea of the tree search where branching results from making successive cuts. Its advantage is that the cuts' positions are not fixed in advance, but remain floating until items are assigned to child nodes. The model is competitive on benchmark instances to the existing approaches from the literature. To the best of our knowledge, CP approaches to this problem are only presented by the pricing problem proposed by Pisinger and Sigurd (2007) as part of their column generation procedure to the two-dimensional packing problem. The guillotine constraints in their model are not implemented explicitly; the model relies on a special subroutine to check whether the candidate solution fulfills the constraints. The authors did not provide any analysis of their model in application to the 2DSKP and the 2DSLOPP.

As observed by Iori et al. (2021), CP recently led to consistent improvements in results for some 2D problems (e.g., see the work of Clautiaux et al. (2008), Mesyagutov, Scheithauer, and Belov (2012), Delorme, Iori, and Martello (2017), Luo and Beck (2022), and Polyakovskiy and M'Hallah (2021)). Being inspired by the success of this type of methodology, we introduce a novel CP model to the 2DSLOPP by modeling the tree of guillotine cuts, with cuts applied first to the original plate in the root node, and then recursively in each internal node to the resulting sub-plates. Eventually, each leaf node of such a tree yields one of the items selected in an optimal solution. Our model takes advantage of cumulative scheduling relaxations intrinsic to 2D packing/cutting problems. This cumulative scheduling component of the model is vital as it includes not only items but also all waste regions that arise in the cutting. This makes the relaxation tight and the model efficient.

Constraint Programming Model

Our CP model for the 2DSLOPP explores the specificity of the cutting problem and does not require guillotine cuts to be set in advance. Indeed, it decides on the necessary cuts, their types, and how they should split the plate to allocate space for the items during the search process. The modeling is centered around the fact that in guillotine cutting a single item or a rectangular region with a set of packed items can be obtained from the original stock plate $(\overline{W}, \overline{H})$ or from its sub-plate (a parent region) via a sequence of exactly two cuts: One cut must be vertical and the other horizontal.

Consider the approach to extract a region r of size (W_r, H_r) from a parent region r^* of size (W_{r^*}, H_{r^*}). Let r be referred to as the '*core*' region of a pattern. As Figure 1 depicts, there are only two options to cut r^*.

- When region r^* is first cut horizontally, as depicted in Figure 1.a, it is split into two horizontal strips: a strip $r' = (W_{r^*}, H_{r^*} - H_r)$ above region r, and a strip composed of r and a region $r'' = (W_{r^*} - W_r, H_r)$, which represents the unused area to the right of r. Then, a subsequent vertical cut is necessary to separate r from r''. This sequence of cuts results in a cutting pattern that is hereinafter referred to as an α pattern.
- When r^* is first cut vertically, as shown in Figure 1.b, it is divided into two vertical strips: a strip $r'' = (W_{r^*} - W_r, H_{r^*})$ to the right of region r, and a strip consisting of r and the empty region $r' = (W_r, H_{r^*} - H_r)$ on top of r. This needs the next cut to be made horizontally to split r and r'. This sequence of cuts is associated with a pattern called a β pattern.

Regardless of the cut type applied first, extraction of the core region r produces two regions: r' to the top and r'' to the right of r. Hereinafter, r' and r'' are referred to as the '*top*' and the '*right*' regions of r, respectively. Obviously, when $W_{r^*} = W_r$ (resp. $H_{r^*} = H_r$), r'' (resp. r') is of zero size. Each pattern contains exactly one waste region (r'' or r').

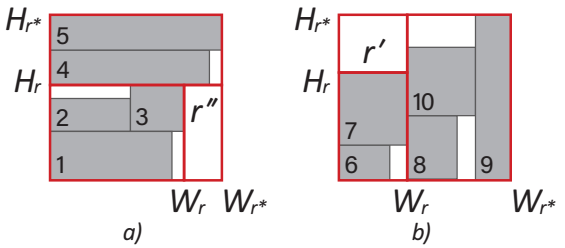

Figure 1: Illustrating the two cutting patterns resulting from a different sequence of cuts: (a) region $r^* \sim (W_{r^*}, H_{r^*})$ is cut via an α pattern to obtain the core region r (with items 1, 2, 3), the top region r' (with items 4 and 5), and the (waste) right region r''; (b) region r^* is cut via a β pattern to extract the core region r (with items 6 and 7), the right region r'' (with items 8, 9, 10), and the (waste) top region r'.

When applying an α (resp. β) pattern, the model assumes that region r'' (resp. r') is a waste (i.e., it must remain empty), while both regions r and r' (resp. r and r'') must accommodate either a single item $i \in I$ or a multiset of items grouped into a rectangular region. Hence, both α and β patterns are applied to construct a complex pattern either by placing two single items side by side, or by attaching a single item to a rectangular region which is a pattern itself (i.e., it must be composed by a group of items and consist of r, r', and r'' regions), or by merging two other patterns. Because this process replicates the binary tree structure of guillotine cutting, obtaining n items requires building exactly $m = n-1$ patterns, with each internal node of the tree being either α or β pattern and leafs representing the items. Let P be a set of these m patterns. Then, for the regions of each pattern $p \in P$, the following constraints must hold: $W_{r'} = W_r + W_{r''}$ if p is an α pattern, and $H_{r''} = H_r + H_{r'}$ if p is a β pattern.

Note that the modeling approach of guillotine gutting presented above is generic in the sense that it can be translated to other solution techniques like MILP.

Decision Variables

Our CP model is assignment-based. Let $R^o = \{r_1, \ldots, r_m\}$ form a set of core regions of the m patterns required to solve the 2DSLOPP. And let $R' = \{r'_1, \ldots, r'_m\}$ and $R'' = \{r''_1, \ldots, r''_m\}$ be sets of their top and right regions, respectively. Then $R = R^o \cup R' \cup R''$ is the set of all $3m$ regions that build the patterns. Hence, for any pattern $p \in P$, $r_p \in R^o$ is the core region of p, while $r'_p \in R'$ and $r''_p \in R''$ are the regions above and to the right of r_p, respectively. To model the recursive nature of guillotine cutting, each pattern but the first must be assigned to a region of another pattern (i.e., the parent region it is to be cut from). Here, regions r_1, r'_1, and r''_1 of pattern $p = 1$ emanate from cutting the plate; so, they are treated as parent regions for any other patterns in $P \setminus \{1\}$. Thus, pattern 1 cannot be assigned to any region. Obviously, because each region $r \in R$ is unique, it can host either a single item or a single pattern (other than the pattern that r belongs to) implementing either α or β layout.

The **first** group of decision variables enables item-to-region assignments. Let $e \in \{0,1\}^{n \times m}$ be a set of binary variables, where $e_{ir} = 1$ (or *true*) if item $i \in I$ is assigned to region $r \in R$, and 0 (or *false*) otherwise. Because an optimal solution to the 2DSLOPP may use up to d_i copies of item $i \in I$, let $u \in \mathbb{N}_0^n$ be a set of integer variables counting the number of copies of each cut item. Hence, the domain $\mathrm{D}(u_i) = \{0, \ldots, d_i\}$, with $u_i = 0$ when item i is not used in the solution. In addition, there are two sets $f' \in \{0,1\}^m$ and $f'' \in \{0,1\}^m$ of auxiliary binary flag variables associated with the regions of sets R' and R'', respectively. Specifically, given a pattern $p \in P$, $f'_p = 1$ (resp. $f''_p = 1$) if there is an item $i \in I$ hosted by region $r'_p \in R'$ (resp. $r''_p \in R''$).

The **second** group of decision variables implements the recursive structure intrinsic to guillotine cutting by assigning patterns to parent regions. Let $a \in \{0,1\}^{m \times 3m}$ be a set of binary variables, where $a_{pr} = 1$ if pattern $p \in P$ is assigned to region $r \in R_p \subset R$, and 0 otherwise. Here, $R_p = \{r_{p-1}, r'_0, \ldots, r'_{p-1}, r''_0, \ldots, r''_{p-1}\}$, which means that pattern p can only occupy the core region of pattern $p-1$ (i.e., that of its immediate predecessor in set P) or be allocated to one of the top or the right regions of patterns that p follows in the order of set P. Pattern m is the last in set P; so, there is no pattern that can occupy any of its regions. Figure 2 illustrates possible assignments.

Indeed, the set I of items may not entirely fit into the plate $(\overline{W}, \overline{H})$. Therefore, some patterns in P may remain unused. Hence, a set $z \in \{0,1\}^m$ of auxiliary binary variables is employed to signal when this happens, with $z_p = 1$ if pattern p is assigned to one of the regions in set R, and 0 otherwise. Furthermore, the model employs two sets $g' \in \{0,1\}^{m-1}$ and $g'' \in \{0,1\}^{m-1}$ of auxiliary binary flag variables. For each pattern $p \in P \setminus \{m\}$, variable $g'_p = 1$ (resp. $g''_p = 1$) if there is a pattern $k \in P$, $p < k$, which is assigned to the p's top region $r'_p \in R'$ (resp. right region $r''_p \in R''$).

The **third** set $v \in \{0,1\}^m$ of binary decision variables selects the layout (α or β) for every cutting pattern $p \in P$: $v_p = 0$ if p is an α pattern, and $v_p = 1$ if it is a β pattern.

Finally, the **fourth** group of decision variables determines the sizes of the $3m$ regions of set R. Let $W \in \mathbb{N}_0^{3m}$ and $H \in \mathbb{N}_0^{3m}$ be two sets of integer variables representing the widths and the heights of these regions, respectively. Then, for each region $r \in R$, a pair (W_r, H_r) specifies its size. Christofides and Whitlock (1977) showed that any feasible cutting pattern can be transformed into an equivalent '*normal*' pattern, where all items are moved towards the bottom-left corner of the plate as much as possible. Thus, in a normal pattern the left and/or the bottom edge of each item $i \in I$ touches the edge of another item or that of the plate. This idea suggests computing in advance the positions $\mathcal{N}_x$ (resp. $\mathcal{N}_y$) of all possible vertical (horizontal) cuts that can be made along the x (resp. y) coordinate. Both sets $\mathcal{N}_x$ and $\mathcal{N}_y$ can be obtained via a standard dynamic programming procedure in pseudo-polynomial time of $\mathcal{O}(n \times \max\{\overline{W}, \overline{H}\})$ by solving Problems (1) and (2), respectively.

$$\mathcal{N}_x = \{x = \textstyle\sum_{i \in I} w_i \xi_i : 0 \le x \le \overline{W},\ \xi_i \in \{0,1\},\ i \in I\} \quad (1)$$

$$\mathcal{N}_y = \{y = \textstyle\sum_{i \in I} h_i \xi_i : 0 \le y \le \overline{H},\ \xi_i \in \{0,1\},\ i \in I\} \quad (2)$$

In our CP model, the domain of width W_r (resp. height H_r) of each region $r \in R^o \cup R'$ (resp. $r \in R^o \cup R''$) can be set to $\mathcal{N}_x$ (resp. $\mathcal{N}_y$). At the same time, if $r \in R'$ (resp. $r \in R''$) the domain of its height H_r (resp. width W_r) remains $\mathrm{D}(H_r) = \{0, \ldots, \overline{H}\}$ (resp. $\mathrm{D}(W_r) = \{0, \ldots, \overline{W}\}$).

Core Constraints

This section formulates a set of important constraints required to ensure guillotine structure of solutions to the 2DSLOPP. The variables' domains are restricted as proposed in the previous section.

The objective function (3) maximizes the total profit produced by the subset of selected items positioned on the plate $(\overline{W}, \overline{H})$, where u_i stores the number of used copies of item $i \in I$. Because each copy of item $i \in I$ can only be allocated to one of the regions in set R, Constraint (4) computes u_i as the sum of e_{ir} assignment variables, with this sum being bound by d_i imposed by the u_i's domain. Constraint (5) ensures that each pattern $p \in P \setminus \{1\}$ is not assigned to more than one region of set R_p. Implicitly, it marks p as used (i.e., $z_p = 1$) when there is a region in R_p which p occupies. This implies that $a_{pr} = 0$ for any $r \in R_p$ if z_p is set to 0.

$$\max \sum_{i \in I} \pi_i u_i \tag{3}$$

$$u_i = \sum_{r \in R} e_{ir} \qquad i \in I \tag{4}$$

$$z_p = \sum_{r \in R_p} a_{pr} \qquad p \in P \setminus \{1\} \tag{5}$$

Constraint (6) ensures that the core region of each pattern $p \in P \setminus \{m\}$ hosts at most one element such as item $i \in I$ or pattern $p+1$, which immediately succeeds p in set P. Since pattern m is the last in set P, there is no pattern that can be allocated to its core region. Hence, Constraint (7) uses z_m to limit the number of items assigned to this region to 1. They both imply that pattern $p \in P$ is in use (i.e., $z_p = 1$) if its core region hosts either an item or a pattern $p+1$.

$$z_p = a_{p+1 r_p} + \sum_{i \in I} e_{i r_p} \qquad p \in P \setminus \{m\} \tag{6}$$

$$z_m = \sum_{i \in P} e_{i r_m} \tag{7}$$

For each pattern $p \in P$, Constraint (8) (resp. Constraint (9)) restricts the number of items assigned to the p's top region r'_p (resp. right region r''_p) to at most one and tags the region as used by an item by setting the corresponding flag variable f'_p (resp. f''_p) to 1. Similarly, for each pattern $p \in P \setminus \{1\}$, Constraint (10) (resp. Constraint (11)) allows only a single pattern-to-region assignment for the p's top region r'_p (resp. right region r''_p) and sets the respective flag variable g'_p (resp. g''_p) to 1 to indicate that the region is occupied by a pattern. Regardless of the layout chosen for p, the use of r'_p excludes that of r''_p, and vice versa. Similarly, the flag variables in pairs (f'_p, g'_p) and (f''_p, g''_p) are mutually exclusive. Therefore, Constraint (12) ensures that at most one of the four variables can take the value of 1.

$$f'_p = \sum_{i \in I} e_{i r'_p} \qquad p \in P \tag{8}$$

$$f''_p = \sum_{i \in I} e_{i r''_p} \qquad p \in P \tag{9}$$

$$g'_p = \sum_{k=p+1}^{m} a_{k r'_p} \qquad p \in P \setminus \{1\} \tag{10}$$

$$g''_p = \sum_{k=p+1}^{m} a_{k r''_p} \qquad p \in P \setminus \{1\} \tag{11}$$

$$z_p = f'_p + f''_p + g'_p + g''_p \tag{12}$$

Constraint (13) implies the application of α layout to each pattern $p \in P$ (i.e., $v_p = 0$) whose top region r'_p holds an item or another pattern. Similarly, Constraint (14) enforces β layout (i.e., $v_p = 1$) for each pattern $p \in P$ whose right region r''_p contains an item or a pattern. Constraint (15) tags p as used when its top r'_p or its right r''_p region is occupied. The implication in Constraints (13-15) is reified.

$$\chi \rightarrow \neg v_p \qquad \chi \in \{f'_p, g'_p\},\ p \in P \tag{13}$$

$$\chi \rightarrow v_p \qquad \chi \in \{f''_p, g''_p\},\ p \in P \tag{14}$$

$$\chi \rightarrow z_p \qquad \chi \in \{f'_p, f''_p, g'_p, g''_p\},\ p \in P \tag{15}$$

Constraints (16)-(21) guarantee the geometric feasibility of the guillotine cutting solution. First, Constraint (16) ensures that pattern $p = 1$ fits into the stock plate $(\overline{W}, \overline{H})$; i.e., the total width of its core region r_p and its right region r''_p is less or equal to $\overline{W}$, while the height of r_p plus the height of its top region r'_p is less or equal to $\overline{H}$.

$$(W_{r_1} + W_{r''_1} \leq \overline{W}) \wedge (H_{r_1} + H_{r'_1} \leq \overline{H}) \tag{16}$$

Then, if item $i \in I$ is assigned to region $r \in R$, Constraint (17) sets the size (W_r, H_r) of r to the size (w_i, h_i) of i. Similarly, Constraints (18) and (19) ensure that the size (W_r, H_r) of region $r \in R_p$ is large enough to fit pattern $p \in P \setminus \{1\}$ if that is allocated to r. The former (resp. latter) constraint sets width W_r (resp. height H_r) to the total width (height) of the p's regions r_p and r''_p (resp. r_p and r'_p).

$$e_{ir} \rightarrow (W_r = w_i) \wedge (H_r = h_i) \qquad i \in I,\ r \in R \tag{17}$$

$$a_{pr} \rightarrow W_r = W_{r_p} + W_{r''_p} \qquad p \in P \setminus \{1\},\ r \in R_p \tag{18}$$

$$a_{pr} \rightarrow H_r = H_{r_p} + H_{r'_p} \qquad p \in P \setminus \{1\},\ r \in R_p \tag{19}$$

Constraint (20) computes the width $W_{r'_p}$ of the top region r'_p of pattern $p \in P$. When p is an α pattern (i.e., $v_p = 0$), its width is that of its core region r_p plus the width of its right (waste) region r''_p. Otherwise (when $v_p = 1$), r'_p appears to be a waste region that makes its width equal to the width of r_p. Similarly, Constraint (21) determines the height $H_{r''_p}$ of the right region r''_p. When p is an β pattern, $H_{r''_p}$ is the total height of r_p and its top region r'_p. Otherwise r''_p is a waste region of height of the core region r_p.

$$W_{r'_p} = W_r + (\neg v_p) \times W_{r''_p} \qquad p \in P \tag{20}$$

$$H_{r''_p} = H_r + v_p \times H_{r'_p} \qquad p \in P \tag{21}$$

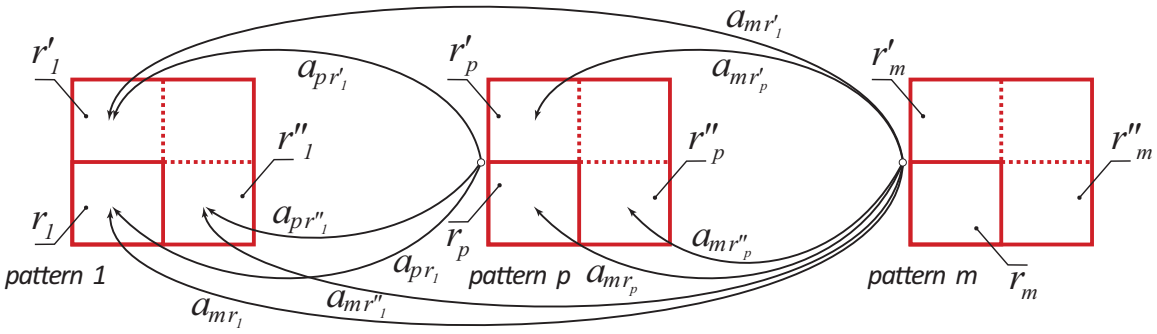

Figure 2: The possible pattern-to-region assignments, where each pattern can only be allocated to the top/right regions of the preceding patterns or to the core region of its immediate predecessor. The dot lines indicate the potential cuts (α or β).

The CP model defined by Constraints (3)-(21) is sufficient to solve the 2DSLOPP. It does not require variables to find items' (x, y) coordinates as the exact position of each item $i \in I$ can be recursively retrieved via the set of (W_r, H_r) pairs, assuming that the core region r_1 of the first pattern is placed into the bottom-left position of the plate with coordinates $(x_{r_1}, y_{r_1}) = (0, 0)$.

The rest of this section first explains how cumulative scheduling can strengthen the existing model, and then proposes a collection of redundant and symmetry breaking constraints to avoid exploration of equivalent solutions.

Cumulative Scheduling Relaxation

Our CP model is strengthened by constraints derived from two non-preemptive cumulative scheduling problems. In practice, an exact solution to each of these problems gives a relaxation to the two-dimensional orthogonal packing problem, which decides whether a whole set of rectangular items is packable into a single large rectangular bin subject to additional geometric constraints (Wäscher, Haußner, and Schumann 2007).

Within the scope of our model, the stock plate $(\overline{W}, \overline{H})$ is seen as a schedule associated with two distinct resources λ_w and λ_h of capacity $\overline{W}$ and $\overline{H}$, respectively. Furthermore, the core regions of set R^o are coupled with two sets of jobs: $J^o = \{j_1, \ldots, j_m\}$ and $L^o = \{l_1, \ldots, l_m, \}$. Similarly, the top (resp. the right) regions of R' (resp. R'') correspond to job sets $J' = \{j'_1, \ldots, j'_m\}$ and $L' = \{l'_1, \ldots, l'_m\}$ (resp. $J'' = \{j''_1, \ldots, j''_m\}$ and $L'' = \{l''_1, \ldots, l''_m\}$). Hence, every region in set R relates to exactly two jobs: one job in the set $J = J^o \cup J' \cup J''$ and the other in the set $L = L^o \cup L' \cup L''$. The first (resp. second) scheduling problem considers the widths (heights) of the regions as processing times of jobs in set J (resp. L) and interprets their heights (widths) as the amount of resource λ_h (λ_w) required to complete these jobs. Thus, the first (resp. second) scheduling problem searches for a feasible schedule in which all jobs in set J (resp. L) are performed within their respective time windows, without preemption and without exceeding the availability $\overline{H}$ (resp. $\overline{W}$) of required resource λ_h (resp. λ_w). Note that the jobs of J and L must be performed simultaneously but using two different resources.

In order to incorporate the two cumulative scheduling problems, the existing CP model requires extra variables to implement the jobs' starting times. In fact, such variables coincide with the (x, y) coordinates of the respective regions. Let $X \in \mathbb{N}_0^{3m}$ and $Y \in \mathbb{N}_0^{3m}$ be two sets of integer variables representing the regions' positions on the plate $(\overline{W}, \overline{H})$. For each region $r \in R$, a pair (X_r, Y_r) determines its location. That is, each job $j \in J$ (resp. $l \in L$) related to its region $r \in R$ must start at time X_r (resp. Y_r) and be completed within the time window $[0, \overline{W})$ (resp. $[0, \overline{H})$). Indeed, because the coordinates of any region in set $R' \cup R''$ can be directly computed from the position and size of its respective core region $r \in R^o$ (cf. Constraints (25) and (26)), variables X_r and Y_r are only required for the core regions, with $X_1 = 0$ and $Y_1 = 0$.

When pattern $p \in P \setminus \{1\}$ is assigned to the core region r_{p-1} of its immediate predecessor $p - 1$, Constraints (22) and (23), respectively, set the x and y-coordinates of its core region r_p equal to those of r_{p-1}. When p is allocated to the top region r'_k of pattern k preceding p in set P, Constraint (24) sets $X_{r_p} = X_{r_k}$, while Constraint (25) calculates Y_{r_p} as the sum of y-coordinate Y_{r_k} and the height H_{r_k}. Similarly, when p occupies the right region r''_k of pattern k (k precedes p), Constraint (26) sets $X_{r_p} = X_{r_k} + W_{r_k}$, while Constraint (27) ensures the equality of Y_{r_p} and Y_{r_k}.

$$a_{pr_{p-1}} \rightarrow (X_{r_p} = X_{r_{p-1}}) \qquad p \in P \setminus \{1\} \quad (22)$$

$$a_{pr_{p-1}} \rightarrow (Y_{r_p} = Y_{r_{p-1}}) \qquad p \in P \setminus \{1\} \quad (23)$$

$$a_{pr'_k} \rightarrow (X_{r_p} = X_{r_k}) \qquad p \in P \setminus \{1\},\ k \in P,\ k < p \quad (24)$$

$$a_{pr'_k} \rightarrow (Y_{r_p} = Y_{r_k} + H_{r_k}) \qquad p \in P \setminus \{1\},\ k \in P,\ k < p \quad (25)$$

$$a_{pr''_k} \rightarrow (X_{r_p} = X_{r_k} + W_{r_k}) \qquad p \in P \setminus \{1\},\ k \in P,\ k < p \quad (26)$$

$$a_{pr''_k} \rightarrow (Y_{r_p} = Y_{r_k}) \qquad p \in P \setminus \{1\},\ k \in P,\ k < p \quad (27)$$

The jobs of sets J and L are tuples of the following form.

$$j_p \sim (X_{r_p}, W_{r_p}, \neg a_{p+1r_p}) \qquad p \in P$$

$$l_p \sim (Y_{r_p}, H_{r_p}, \neg a_{p+1r_p}) \qquad p \in P$$

$$j'_p \sim (X_{r_p}, W_{r'_p}, \neg g'_p) \qquad p \in P$$

$$l'_p \sim (Y_{r_p} + H_{r_p}, H_{r'_p}, \neg g'_p) \qquad p \in P$$

$$j''_p \sim (X_{r_p} + W_{r_p}, W_{r''_p}, \neg g''_p) \qquad p \in P$$

$$l''_p \sim (Y_{r_p}, H_{r''_p}, \neg g''_p) \qquad p \in P$$

Here, the first two elements of each tuple define, respectively, the starting and the processing time while the last element indicates the presence of the job in the schedule. As the result, jobs j_p and l_p, $p \in P$, are both present when $a_{p+1r_p} = 0$, i.e. when its associated core region r_p is occupied by an item, but not pattern $p+1$. Similarly, jobs j'_p and l'_p (resp. j''_p and l''_p) are scheduled when $g'_p = 0$ (resp. $g''_p = 0$), i.e. when the related region r'_p (resp. r''_p) either represents a waste or hosts an item. Because the regions r_1, r'_1, and r''_1 of pattern $p = 1$ (i.e., those constituting the stock plate) must be used in any solution, the jobs related to these regions appear compulsory. Furthermore, the core region of pattern $p = m$ may only contain an item that makes its respective jobs j_m and l_m compulsory too.

To integrate the two cumulative scheduling components, the model employs two constraints specific to CP. Constraint (28) (resp. Constraint (29)) forces the jobs of J (resp. L) to be complete within their respective time windows without exceeding the resource's capacity $\overline{H}$ (resp. $\overline{W}$).

$$\texttt{Cumulative}\big(J, H, \overline{H}\big) \tag{28}$$

$$\texttt{Cumulative}\big(L, W, \overline{W}\big) \tag{29}$$

Here, the three arguments of the cumulative function are, respectively, a set of jobs, a set representing the resource usage related to performing each of the jobs, and the resource capacity.

The cumulative scheduling relaxation of our CP model is *extremely* tight as it requires scheduling both types of jobs: the regular jobs emanating from the items to be cut and all jobs produced by the waste regions of designed patterns. This ensures that the surface of the stock plate is filled with no gaps within the boundaries $(W_{r_1} + W_{r''_1}, H_{r_1} + H_{r'_1})$ determined by the size of the first pattern of set P.

Redundant and Symmetry Breaking Constraints

Constraints (30) and (31) are two redundant symmetry breaking constraints. Although they can be drawn, respectively, from Constraints (20) and (21), adding them to the model does speed the search. For each pattern $p \in P$ representing an α pattern, the former constraint ensures that the width of its core region r_p does not exceed the width of its top region r'_p. Similarly, when p is a β pattern, the latter constraint bounds the height of r_p by the height of its right region r''_p.

$$\neg v_p \rightarrow W_{r_p} \leq W_{r'_p} \qquad p \in P \tag{30}$$

$$v_p \rightarrow H_{r_p} \leq H_{r''_p} \qquad p \in P \tag{31}$$

If pattern $p \in P$ is of no need (i.e., $z_p = 0$), Constraint (32) sets the size (W_{r_p}, H_{r_p}) of the core region r_p to zero, while Constraint (32) sets to 0 the width of p's right and the height of p's top region. Through Constraints (20) and (21), this implies that all three regions (i.e., r_p, r'_p, and r''_p) constituting p are of zero size. Furthermore, Constraint (34) positions such redundant regions in the bottom-left corner $(0, 0)$ of the plate. These three constraints expedite the search.

$$\neg z_p \rightarrow (W_{r_p} = 0) \wedge (W_{r_p} = 0) \qquad p \in P \tag{32}$$

$$\neg z_p \rightarrow (W_{r''_p} = 0) \wedge (H_{r'_p} = 0) \qquad p \in P \tag{33}$$

$$\neg z_p \rightarrow (x_p = 0) \wedge (y_p = 0) \qquad p \in P \tag{34}$$

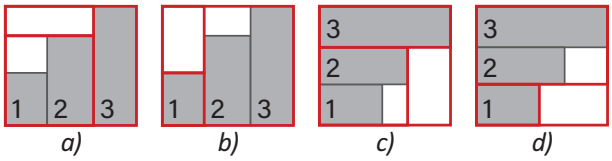

Figure 3: Equivalent solutions resulted from nesting patterns with the same layout.

Constraint (35) breaks the symmetry: If pattern p, $p = 2, \ldots, m-1$, is not used in the solution, then its immediate successor $p+1$ becomes redundant. Generally, this implies that no pattern that follows p in P can be used. Furthermore, for each pattern $p \in P$ but the first, if p is not needed, Constraint (36) prohibits the top and the right regions of its predecessor $p-1$ to hold a pattern. Hence, the entire pattern $p-1$ can only be allocated to items, that makes it the last used pattern in set P.

$$\neg z_p \rightarrow (\neg z_{p+1}) \qquad p = 2, \ldots, m-1 \tag{35}$$

$$\neg z_p \rightarrow (\neg g'_{p-1}) \wedge (\neg g''_{p-1}) \qquad p \in P \setminus \{1\} \tag{36}$$

Constraints (37) and (38) consider two subsequent patterns $p-1$ and p, $p \in P \setminus \{1\}$, and forbid symmetric packing which they can produce. Specifically, these constraints assume that either p is not located in the core region of $p-1$ or the layouts (α or β) of $p-1$ and p are different. For example, there are two patterns that can be obtained without Constraint (37) being imposed, as Figures 3.a and 3.b depict. In Figure 3.a, β pattern p is one composed by items 1 and 2 and assigned to the core region of β pattern $p-1$, whose right region is occupied by item 3. The same layout of items (cf. Figure 3.b) can be achieved by making p consisting of items 2 and 3 and assigning p to the right region of pattern $p-1$, whose core region contains item 1. Therefore, should both $p-1$ and p implement a β pattern, p may only occupy the right region (not the core region) of pattern $p-1$. Therefore, Constraint (37) excludes the pattern in Figure 3.a as it yields a layout equivalent to that in Figure 3.b. Similarly, Constraint (38) excludes pattern in Figure 3.c as equivalent to that in Figure 3.d; that is, if both $p-1$ and p are α patterns, then p may only occupy the top region of pattern $p-1$, not its core region.

$$(\neg v_p) \vee (\neg v_{p-1}) \vee (\neg a_{pp-1}) \qquad p \in P \setminus \{1\} \tag{37}$$

$$(v_p) \vee (v_{p-1}) \vee (\neg a_{pp-1}) \qquad p \in P \setminus \{1\} \tag{38}$$

Consider each pattern $p \in P$ except the last. A group of symmetry breaking Constraints (39-45) controls the allocation of p's top and right regions to its successors in set P. These constraints require a set $t \in \mathbb{N}_0^{m-1}$ of auxiliary integer variables, where t_p, $p \in P \setminus \{m\}$, is a tag value associated with pattern p. For $p = 1$ implementing an α (resp. β) pattern with its top (right) region occupied by an item, Constraint (39) (resp. Constraint (40)) sets the tag variable t_1 to 0. For each pattern p, $p = 2, \ldots, m-1$, Constraints (41) and (42) act similarly and require the tag variable t_p of p be equal to that of its immediate predecessor $p-1$. When pattern p implements an α (resp. β) pattern and its top (right) region

Inst.	n	OPT	Furini et al.		Martin et al.		Silva et al.		CP						
			ρ	t (s)	ρ	t (s)	ρ	t (s)	ρ	t (s)	t^* (s)	$\overline{m}$	u^*	n_{fail}	t^-
gcut1	10	48368	0	8.80	0	0.04	0	642.25	**0**	**0.03**	0.01	2	3	36	0.09
gcut2	20	59307	100	∞	0	4.65	0	∞	**0**	**0.65**	0.19	6	5	4541	0.83
gcut3	30	60241	100	∞	0	13.12	0	∞	**0**	**1.12**	0.55	7	5	7796	2.47
gcut4	50	60942	100	∞	0	509.84	0	∞	**0**	**11.97**	3.56	8	4	38588	18.54
gcut5	10	195582	1.4	∞	0	0.21	0	646.35	**0**	**0.09**	0.04	4	5	481	0.12
gcut6	20	236305	100	∞	0	0.70	0	∞	**0**	**0.37**	0.22	5	4	1431	0.42
gcut7	30	238974	100	∞	0	2.74	0	∞	**0**	**0.47**	0.30	6	4	2267	0.77
gcut8	50	245758	100	∞	0	62.44	0	∞	**0**	**7.68**	4.09	8	4	21978	11.85
gcut9	10	919476	0	2847.90	0	0.28	0	∞	**0**	**0.10**	0.05	5	5	310	0.13
gcut10	20	903435	100	∞	0	0.78	0	∞	**0**	**0.20**	0.14	4	4	608	0.26
gcut11	30	955389	100	∞	0	14.48	0	∞	**0**	**1.18**	0.60	7	6	6292	2.44
gcut12	50	970744	100	∞	0	14.74	0	∞	**0**	**2.08**	0.62	6	4	8176	3.72

Table 1: Stage-unrestricted unweighted 2DSKP: Comparison between MILP_1, MILP_2, MILP_3, and CP.

Inst.	n	$\bar{d}$	OPT	Furini et al.		Martin et al.		Silva et al.		CP						
				ρ	t (s)	ρ	t (s)	ρ	t (s)	ρ	t (s)	t^* (s)	$\overline{m}$	u^*	n_{fail}	t^-
of1	10	23	2737	0	53.90	0	∞	0.88	∞	**0**	**41.06**	2.11	10	10	1284806	151.09
of2	10	24	2690	0	66.00	**0**	**5.94**	0	∞	0	14.49	1.01	9	10	140934	27.76
cu1	25	82	12330	0	1460.80	0.15	∞	0	∞	**0**	**18.22**	2.26	11	6	91969	46.66
cu2	35	90	26100	100	∞	0	∞	0.56	∞	**0**	**27.19**	2.21	10	11	110396	56.64
cu3	45	158	16723	—	—	0.69	∞	1.00	∞	0.26	∞	96.05	16	9	—	∞
cu4	45	113	99495	—	—	0.20	∞	0.23	∞	**0**	**872.31**	186.84	14	10	2389172	∞
cu5	50	120	173364	—	—	0.53	∞	0.20	∞	**0**	**795.27**	51.40	13	9	1539661	∞
cu6	45	124	158572	—	—	0	∞	0.51	∞	**0**	**349.49**	58.96	12	7	539828	395.54
cu7	25	56	247150	—	—	0	∞	0.41	∞	**0**	**11.26**	3.48	10	10	56607	22.84
cu8	35	78	433331	—	—	0.14	∞	0.26	∞	**0**	**153.41**	12.21	13	11	414998	268.43
cu9	25	76	657055	—	—	0	∞	0	∞	**0**	**10.07**	3.08	10	5	63384	19.80
cu10	40	129	773772	—	—	0.85	∞	0.51	∞	0.77	∞	∞	20	14	—	∞
cu11	50	134	924696	—	—	2.02	∞	2.95	∞	0.77	∞	∞	23	12	—	∞
wang20	20	42	2721	0	60.70	0	435.10	0	∞	**0**	**7.22**	5.39	10	6	33582	9.05

Table 2: Stage-unrestricted unweighted 2DSLOPP: Comparison between MILP_1, MILP_2, MILP_3, and CP.

is occupied by a pattern succeeding it in set P, Constraint (43) (resp. Constraint (44)) sets the tag variable t_p to the index of the pattern assigned to p's top region r'_p (resp. right region r''_p). Finally, Constraint (45) breaks symmetry by limiting the tag value of pattern p to the value of its immediate successor $p+1$.

$$(\neg v_1 \wedge f'_1) \rightarrow t_1 = 0 \tag{39}$$

$$(v_1 \wedge f''_1) \rightarrow t_1 = 0 \tag{40}$$

$$(\neg v_p \wedge f'_p) \rightarrow t_p = t_{p-1} \qquad p = 2, \ldots, m-1 \tag{41}$$

$$(v_p \wedge f''_p) \rightarrow t_p = t_{p-1} \qquad p = 2, \ldots, m-1 \tag{42}$$

$$(\neg v_p \wedge g'_p) \rightarrow t_p = \sum_{k=p+1}^{m} k a_{kr'_p} \qquad p \in P \setminus \{m\} \tag{43}$$

$$(v_p \wedge g''_p) \rightarrow t_p = \sum_{k=p+1}^{m} k a_{kr''_p} \qquad p \in P \setminus \{m\} \tag{44}$$

$$t_p \leq t_{p+1} \qquad p \in P \setminus \{m\} \tag{45}$$

Computational Results

The CP model is implemented in C# and solved via Google OR-Tools CP-SAT Solver. It is run on a personal computer with 4GB of RAM and a 3.06GHz Dual Core processor. The solver's settings are default with the limit to one computational core. The model was tested on the benchmark instances classical to the 2DSKP and 2DSLOPP (Iori et al. 2022) and compared to the results of the state-of-the-art exact MILP models proposed by Furini, Malaguti, and Thomopulos (2016) (MILP_1), Martin, Morabito, and Munari (2021) (MILP_2), and Silva et al. (2023) (MILP_3). Tables 1-3 report the results of computational experiments on these test instances. Their first columns present the name, the number of items n, the known optimum (OPT), and, where applicable, the total number of items $\bar{d} = \sum_{i=1}^{n} d_i$ for each instance. For each compared approach, the optimality gap ρ (%) is computed as $100 \times \frac{\text{OPT}-\text{OBJ}}{\text{OPT}}$, where OBJ is the objective function value obtained by the approach. In addition, t (s) provides the computational time taken by each approach. In regard to the CP model, the tables also detail the time t^* (s) taken to reach the optimum, the upper bound $\overline{m}$ on the number of patterns in an optimal solution that CP used by setting $m = \overline{m}$, the number of items in the incumbent solution u^*, and the number n_{fail} of fails the solver made during the search. The last column is the runtime t^- (s) for the CP model run with disabled cumulative scheduling constraints. Though MILP_2 was allocated 1 hour of time limit, our CP model restricts the runtime to 900 seconds as it was done

Inst.	n	$\bar{d}$	OPT	Furini et al.		Martin et al.		Silva et al.		CP						
				ρ	t (s)	ρ	t (s)	ρ	t (s)	ρ	t (s)	t^* (s)	$\overline{m}$	u^*	n_{fail}	t^-
cgcut1	7	16	244	**0**	**0.10**	0	4.92	0	∞	0	169.22	1.55	12	8	2552224	374.46
cgcut2	10	23	2892	**0**	**58.40**	1.24	∞	1.24	∞	0	∞	6.31	17	13	—	∞
cgcut3	19	62	1860	0	58.60	0	8.28	0	∞	**0**	**4.77**	1.43	10	10	36766	9.03
okp1	15	50	27589	0	**490.50**	—	—	0	∞	0	∞	132.27	21	10	—	∞
okp2	30	30	22503	30	∞	—	—	2.47	∞	0.005	∞	∞	14	11	—	∞
okp3	30	30	24019	8.8	∞	—	—	1.15	∞	0	∞	39.15	13	11	—	∞
okp4	33	61	32893	0	**684.40**	—	—	0	∞	0	∞	428.09	15	12	—	∞
okp5	29	97	27923	0	**2118.60**	—	—	0	∞	0	∞	395.23	16	16	—	∞
CW1	25	67	6402	70.7	∞	0	∞	0	∞	**0**	**41.60**	3.84	13	12	188673	81.44
CW2	30	63	5354	100	∞	0	409.66	0	∞	**0**	**19.80**	7.67	10	8	81786	31.86
CW3	40	96	5689	100	∞	0	∞	0	∞	**0**	**20.52**	2.28	10	10	82232	42.80
CW4	39	86	6175	—	—	0	∞	0	∞	**0**	**17.66**	6.64	13	9	83397	47.60
CW5	35	91	11659	—	—	0	∞	0	∞	**0**	**13.21**	1.87	12	12	71204	29.72
CW6	55	149	12923	—	—	0	∞	2.10	∞	**0**	**454.37**	46.85	17	16	792572	800.26
CW7	45	123	9898	—	—	0	∞	0	∞	**0**	**74.91s**	12.64	15	13	140197	138.60
CW8	60	168	4605	—	—	0	∞	2.19	∞	**0**	**352.58**	58.30	11	10	769368	555.03
CW9	50	131	10748	—	—	0	∞	0	∞	**0**	**296.28**	20.49	14	12	549262	492.01
CW10	60	130	6515	—	—	0	583.77	0	∞	**0**	**16.76**	6.85	9	9	59144	31.18
CW11	60	114	6321	—	—	0	214.53	0	∞	**0**	**9.21**	3.25	9	9	36141	19.14

Table 3: Stage-unrestricted weighted 2DSLOPP: Comparison between MILP_1, MILP_2, MILP_3, and CP.

for models MILP_1 and MILP_3. The dash ('—') used in the tables means the instance has no published results for the respective approach, while '∞' indicates a timeout. Thus, $(\rho, t) = (100, \infty)$ means that the approach failed to find any solution, while $(0, \infty)$ indicates that the optimum was reached but not proved.

First, CP was applied to the unweighted 2DSKP and tested on the gcut1-12 instances proposed by Beasley (1985) with results reported in Table 1. Here, CP substantially outperforms all MILPs by solving all instances in a much shorter time. MILP_2 and CP are the only models that solved the entire data set to optimality. However, the average runtime of MILP_2 is 52 sec versus only 2 sec for CP.

Second, the efficiency of CP in application to the unweighted 2DSLOPP was studied on the following instances in Table 2: of1-2, cu1-11, and wang20 proposed by Oliveira and Ferreira (1990), Fayard, Hifi, and Zissimopoulos (1998), and Wang (1983), respectively. In only the of2 instance CP is slightly slower than MILP_2. On other instances, it is much faster and solves instances cu2 and cu4-9 failed by all MILPs, often even in terms of reaching the optima. Although, cu2, cu10, and cu11 appear also hard for CP, the optimality gaps are smaller than those of the MILPs.

Finally, CP tackled the weighted 2DSLOPP. Table 3 addresses test instances cgcut1-3, CW1-11, and okp1-5 proposed by Christofides and Whitlock (1977), Hifi and Roucairol (2001), and Fekete, Schepers, and van der Veen (2007), respectively. The class CW is one where CP outperforms all MILPs offering very short runtimes for most of the instances. Furthermore, CP succeeds with tests CW1 and CW3-9, all failed by MILPs. The only challenging instances for CP are classes cgcut and okp. However, only few of them can be solved by MILP_1, with okp2 and okp3 remaining hard for all models. Although, CP reaches the optimal solution here reasonably fast, proving the optimality remains challenging. We believe that these classes are difficult mainly because of the objective function and the way the profits were generated rather than the rectangular packing itself. In fact, optimising objective functions containing sums is a known weakness of CP as this hinders inference and propagation during the search.

Interestingly, even with the cumulative scheduling constraints being disabled, CP remains a superior model for most of the instances, and often the only model capable of solving difficult instances. On average, these constraints make CP $\sim$2 times faster. This proves that the cumulative scheduling relaxations are a vital ingredient of our model, and potentially that of future methods that may adopt them.

Conclusion

This paper solves the guillotine stage-unrestricted rectangular cutting problem. Its main contribution consists in a novel CP model that appears highly-competitive in application to instances of moderate size. When compared to more traditional MILPs, our model is capable of solving instances that cannot be handled by these alternative models. Embedding cumulative scheduling relaxations is vital for the model as they drive the search process allocating space for the items and all waste regions resulting from cutting. Considering items and waste regions together ensures a very tight relaxation as it leaves no gaps in filling the stock plate.

Our model can be easily extended to more complex packing problems, e.g. the bin packing with identical/heterogeneous bins. Indeed, the model should only deal with multiple bins at the root node of the tree of guillotine cuts and span the two cumulative scheduling relaxations across the bins.

Acknowledgments

The research work of Peter Stuckey was partially supported by the OPTIMA ARC ITTC, Project ID 200100009.

References

Beasley, J. E. 1985. Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. *Journal of the Operational Research Society*, 36(4): 297–306.

Christofides, N.; and Whitlock, C. 1977. An Algorithm for Two-Dimensional Cutting Problems. *Operations Research*, 25(1): 30–44.

Clautiaux, F.; Jouglet, A.; Carlier, J.; and Moukrim, A. 2008. A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, 35(3): 944–959. Part Special Issue: New Trends in Locational Analysis.

Delorme, M.; Iori, M.; and Martello, S. 2017. Logic based Benders' decomposition for orthogonal stock cutting problems. *Computers & Operations Research*, 78: 290–298.

Dyckhoff, H. 1981. A New Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, 29(6): 1092–1104.

Fayard, D.; Hifi, M.; and 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.

Fekete, S. P.; Schepers, J.; and van der Veen, J. C. 2007. An Exact Algorithm for Higher-Dimensional Orthogonal Packing. *Operations Research*, 55(3): 569–587.

Furini, F.; Malaguti, E.; and Thomopulos, D. 2016. Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. *INFORMS Journal on Computing*, 28(4): 736–751.

Hifi, M.; and 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.

Iori, M.; de Lima, V. L.; Martello, S.; Miyazawa, F. K.; and Monaci, M. 2021. Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, 289(2): 399–415.

Iori, M.; de Lima, V. L.; Martello, S.; and Monaci, M. 2022. 2DPackLib: a two-dimensional cutting and packing library. *Optimization Letters*, 16(2): 471 – 480.

Lodi, A.; Martello, S.; and Vigo, D. 1999. Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. *INFORMS Journal on Computing*, 11(4): 345–357.

Luo, Y. L.; and Beck, J. C. 2022. Packing by Scheduling: Using Constraint Programming to Solve a Complex 2D Cutting Stock Problem. In Schaus, P., ed., *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, 249–265. Cham: Springer International Publishing.

Martin, M.; Morabito, R.; and Munari, P. 2021. A top-down cutting approach for modeling the constrained two- and three-dimensional guillotine cutting problems. *Journal of the Operational Research Society*, 72(12): 2755–2769.

Mesyagutov, M.; Scheithauer, G.; and Belov, G. 2012. LP bounds in various constraint programming approaches for orthogonal packing. *Computers & Operations Research*, 39(10): 2425–2438.

Oliveira, J.; and Ferreira, J. 1990. An improved version of Wang's algorithm for two-dimensional cutting problems. *European Journal of Operational Research*, 44(2): 256–266. Cutting and Packing.

Pisinger, D.; and Sigurd, M. 2005. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2): 154–167.

Pisinger, D.; and Sigurd, M. 2007. Using Decomposition Techniques and Constraint Programming for Solving the Two-Dimensional Bin-Packing Problem. *INFORMS Journal on Computing*, 19(1): 36–51.

Polyakovskiy, S.; and M'Hallah, R. 2021. Just-in-time two-dimensional bin packing. *Omega (United Kingdom)*, 102.

Silva, E.; Oliveira, J. F.; Silveira, T.; Mundim, L.; and Carravilla, M. A. 2023. The Floating-Cuts model: a general and flexible mixed-integer programming model for non-guillotine and guillotine rectangular cutting problems. *Omega*, 114: 102738.

Velasco, A. S.; and Uchoa, E. 2019. Improved state space relaxation for constrained two-dimensional guillotine cutting problems. *European Journal of Operational Research*, 272(1): 106–120.

Wang, P. Y. 1983. Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research*, 31(3): 573–586.

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