

A regular perspective on solving the irregular strip packing problem with the dotted-board model: a mosaic approach and a divide-and-conquer strategy

Maxence Delorme⁽¹⁾, José Fernando Oliveira⁽²⁾

(1) Department of Econometrics and Operations Research, Tilburg University, The Netherlands

(2) INESC TEC, Faculdade de Engenharia, Universidade do Porto, Portugal

Corresponding author m.delorme@tilburguniversity.edu

Abstract

Since its introduction in 2013, the dotted-board model has been widely used by the cutting and packing research community to tackle irregular packing problems. However, despite its popularity, little research has aimed at improving its performance when solved via off-the-shelf integer programming solvers, beyond the known strategy of merging pairwise incompatibility constraints into clique constraints during a somewhat lengthy preprocessing. In this work, we bridge this gap by introducing two complementary contributions that bring regular packing concepts to the irregular strip packing problem. In particular, we propose a mosaic approach in which each fragment corresponds to a physical location on the strip and is associated with a clique constraint in the dotted-board model. We also present a divide-and-conquer strategy in which fragments are first computed over a single 1×1 tile of the strip and subsequently extended to the entire strip. Furthermore, we evaluate techniques to reduce the number of constraints and non-zero coefficients in the resulting models, along with alternative formulations for the objective function. Our contributions result in a competitive approach capable of solving benchmark instances previously unsolved in the literature. But more importantly, they offer new perspectives on the dotted-board model, highlighting its structural resemblance to models in regular packing, which, alongside our open-source implementation, will hopefully encourage further research on the topic.

Keywords: combinatorial optimization; irregular strip packing; dotted-board model; integer linear programming; computational experiments.

1 Introduction

Cutting and packing (C&P) problems have been extensively investigated by the research community for various reasons. For instance, one-dimensional C&P problems often serve as a playground to develop a wide range of mathematical models (see, e.g., the reflect-based flow formulations of Delorme and Martinovic (2026) for variable-sized cutting and skiving stock problems), tailored algorithms (see, e.g., the branch-cut-and-price algorithm of da Silva and Schouery (2025) for several variants of cutting stock problems), and theoretical results (see, as one example among many, the improved lower bound on the asymptotic competitive ratio for online bin packing developed by Balogh et al. (2021)). Three-dimensional problems, in turn, often model complex practical applications that are relevant in modern society (see, e.g., the work of Paquay et al. (2016) on air cargo logistics and that of Lamas-Fernandez et al. (2023) linking three-dimensional irregular packing problems to 3D printing).

Two-dimensional problems, meanwhile, seem to effectively combine both worlds, with some contributions aimed at enriching the C&P field as a whole (see, e.g., the scheduling-inspired formulations of Castro and Grossmann (2012) for the strip packing problem) and others at addressing concrete industrial challenges (e.g., in the automotive (Alves et al., 2012), leather (Heistermann and Lengauer, 1995), or glass industries (Han et al., 2013); see the introduction of Lastra-Díaz and Ortuño (2026) for additional examples).

Although two-dimensional C&P problems are highly heterogeneous, their core elements are typically the same: a single container with fixed dimensions into which a set of items must be packed (in packing terminology). Three major families of problems can be distinguished based on which of these elements is relaxed and optimized: (i) in *knapsack* problems, not all items need to be packed; instead, the goal is to maximize the total value of the packed items, (ii) in *strip packing* problems, one of the container’s dimensions is not fixed; instead, the goal is to minimize that dimension, and (iii) in *bin packing* problems, the number of containers is not limited to one; instead, the goal is to minimize the number of containers used. A key challenge in two-dimensional problems (which does not exist in one-dimensional ones) is geometry: how to ensure that items do not overlap and that all items fit inside the container. Since this challenge is shared across all three problem families, it is worth noting that any innovative method for handling geometry in one problem is likely to be applicable to the others.

Within the field of two-dimensional packing problems, another distinction usually exists between *regular* and *irregular* problems. While the former typically assumes that both containers and items are rectangular (though some studies also class circular shapes as regular (Leao et al., 2020)), the latter allows either or both to have arbitrary shapes. (We note that while such arbitrary shapes could, in principle, take any form, we follow the main stream of the literature and restrict our attention to polygons.) Because of the simpler geometric shapes involved, regular packing problems offer several advantages over their irregular counterparts. First and foremost, determining whether two items placed at given positions overlap is trivial in the regular case, whereas it requires advanced techniques (such as the no-fit polygon, the D-function, or the Phi-function) in the irregular case (Bennell and Oliveira, 2009). In addition, because packing positions can always be discretized in the regular case, researchers have developed several approaches over the years to reduce their number, such as normal patterns (Christofides and Whitlock, 1977; Herz, 1972), raster points (Scheithauer and Terno, 1996), and meet-in-the-middle patterns (Côté and Iori, 2018). Such reductions are generally not possible in the irregular case. Furthermore, when item rotation is allowed, it is typically restricted to two orientations in regular packing (0° and 90°), whereas the number of possible rotations is, at least in theory, unlimited in the irregular case (see, e.g., the recent **sparrow** heuristic by Gardeyn et al. (2026), which handles continuous rotation).

These differences have naturally shaped the research directions in each subfield. In particular, the exploitation of the above-mentioned features of regular problems has enabled the development of effective exact approaches capable of solving strip packing instances with up to 100 items, even when 90° item rotations are allowed (Côté et al., 2014; Delorme et al., 2017). Conversely, the much more challenging geometry of irregular problems has so far prevented comparable performance. For example, the integer linear programming (ILP) approach proposed by Cherri et al. (2016) could only solve irregular strip packing instances with up to 14 items when rotation was not allowed, and up to 10 items when 180° rotation was also allowed. This inherent complexity is probably the reason why researchers have, for a long time, primarily focused on heuristics and metaheuristics when dealing with irregular problems (Dowsland and Dowsland, 1995; Hopper and Turton, 2001).

This distinction has blurred in recent years, as several studies have introduced specific simplifications that make irregular packing problems more akin to regular ones, enabling the development of more effective modeling strategies, sometimes inspired by approaches originally proposed for regular packing. One such simplification, referred to as the raster (or pixel) method, divides both containers and items into sets of unit squares (Segenreich and Braga, 1986). While this reduces the number of packing positions (which become a discrete subset of the bottom-left corners of the unit squares) and simplifies overlap detection (since it suffices to check whether a unit square in the container grid is occupied by more than one item), the main challenge lies in choosing the grid resolution. If the resolution is too low, the representation becomes inaccurate, leading to infeasible or subpar solutions; if it is too high, the number of candidate positions and overlap checks grows excessively, resulting in increased computation time. Another simplification is the discretization of packing positions alone, which forms the underlying principle of the well-known *dotted-board model* (Toledo et al., 2013). Although resolution still plays a role here, it is less critical than in the raster method because the shapes of the items are preserved. While discretizing the feasible placement region in this manner does not render overlap detection as simple as in the raster method, it allows item overlaps to be modeled entirely using a conflict graph, where each vertex represents the placement of a given item at a given packing position, and each edge denotes an overlap between two such placements. Any independent set in the resulting graph therefore represents an overlap-free packing, opening a wide range of opportunities to solve irregular packing problems using the vast literature on graph theory.

While the dotted-board model has been extensively used (e.g., Cherri et al. (2026) integrated it into a branch-and-cut algorithm to solve irregular packing problems with guillotine constraints) and even extended (e.g., Cherri et al. (2018) adapted it to account for item rotation), the study of Rodrigues and Toledo (2017) is one of the few and most relevant contributions specifically focused on improving its empirical performance. Their approach merges pairwise incompatibility constraints into clique constraints, though it requires an extensive preprocessing step. In a recent work, Lastra-Díaz and Ortuño (2026) also identified the improved model of Rodrigues and Toledo (2017) as the state of the art for the dotted-board model, while pointing out that its required (and time-consuming) preprocessing step constitutes a drawback of the formulation.

In this work, we aim to improve the empirical performance of the dotted-board model for the irregular strip packing problem beyond what was achieved by Rodrigues and Toledo (2017), while also challenging the claim by Lastra-Díaz and Ortuño (2026) that a clique-based version of the dotted-board model necessarily requires a lengthy preprocessing step. To this end, we make two primary contributions: (i) a mosaic approach in which each fragment corresponds to a physical location in the strip and is associated with a clique constraint in the dotted-board model and (ii) a divide-and-conquer strategy in which fragments are first computed over a single 1×1 tile of the strip and subsequently extended to the entire strip. We also provide an extensive empirical evaluation of alternative structures for modeling the objective function to identify the one that performs best. These contributions are inspired by recent successes in regular packing and, more broadly, combinatorial optimization. We will observe that, when applied appropriately, the proposed techniques result in a much faster preprocessing step and the associated mathematical model solves slightly more instances to optimality than the model of Rodrigues and Toledo (2017) when using a general-purpose ILP solver. In order to encourage further research in this area, we have made our source code publicly available in the online repository <https://github.com/mdelorme2/ISPP-Mosaic>.

The remainder of this paper is structured as follows. In Section 2, we review the relevant litera-

ture. Section 3 provides a formal description of the irregular strip packing problem, the dotted-board model, and its clique-based improvement. Our main contributions, namely the mosaic approach and the divide-and-conquer strategy, are presented in Sections 4 and 5, respectively. Section 6 reports the results of comprehensive computational experiments comparing the proposed approaches with existing dotted-board-based models, along with an evaluation of alternative structures for modeling the objective function. Finally, Section 7 concludes the paper.

2 Literature review

The literature on two-dimensional irregular packing, also known as *nesting*, is vast, and it is not our ambition to provide a comprehensive survey of the field, but rather to focus on the contributions directly related to our work. Nevertheless, we highlight the dedicated survey of Guo et al. (2022), the accessible tutorials of Bennell and Oliveira (2008, 2009), as well as the excellent literature review of Lastra-Díaz and Ortuño (2026), whose study, although primarily aimed at introducing a new state-of-the-art approach for the irregular strip packing problem, also discusses more than a hundred relevant contributions.

Since the seminal work of Art (1966), who presented what is likely the first study on irregular packing problems, numerous contributions dealing with both “exact” and heuristic approaches have been proposed. Because our work focuses on the former, we refer the reader to early surveys by Dowsland and Dowsland (1995) and Hopper and Turton (2001) for details on heuristic methods, as well as to the more recent work of Giovenali et al. (2024), who cataloged and empirically evaluated (meta)heuristic approaches published between 2006 and 2023.

Regarding “exact” approaches, let us first highlight that we also include in this category contributions that propose an exact algorithm for a two-dimensional irregular packing problem incorporating one of the spatial discretizations discussed in the introduction (such as the raster method or the discretization of packing positions alone). Quotation marks are used because the solutions obtained by these approaches are exact only within the context of the underlying spatial discretization and would not necessarily remain optimal if the latter were removed. Interestingly, almost all of these “exact” approaches rely on mathematical formulations solved via general-purpose solvers. For more information on this matter, we refer the reader to the comprehensive survey by Leao et al. (2020), who distinguish three main categories based on the type of variables used to determine item coordinates: (i) models that use a continuous variable for each item coordinate (Cherri et al., 2016), (ii) models that use a continuous variable for one coordinate and a set of binary variables for the other (Leao et al., 2016), and (iii) models that use a set of binary variables for each coordinate (Toledo et al., 2013). Note that the latter two assume spatial discretization.

Recent progress has been made within the continuous-model family. In particular, Lastra-Díaz and Ortuño (2024) proposed mixed integer linear programming formulations based on a vertical-slice representation of the no-fit-polygon feasible regions, together with feasibility cuts and other strengthening techniques, substantially improving the performance of exact approaches in this category. Our model of interest, the dotted-board model, belongs to the third category. It was introduced by Toledo et al. (2013) and has subsequently been extended and strengthened in several directions. Cherri et al. (2018) developed a data structure that also accommodates item rotations, while Rodrigues and Toledo (2017) strengthened the formulation by replacing pairwise incompatibility constraints with clique-covering constraints. More recently, Guo et al. (2025) extended the dotted-board framework

to irregular multi-plate packing, and Cherri et al. (2026) embedded it in a branch-and-cut algorithm for irregular packing problems with guillotine constraints. Lastra-Díaz and Ortuño (2026), in turn, departed from the use of a general-purpose ILP solver and proposed a tailored parallel branch-and-bound-and-check algorithm operating on the complement of the conflict graph. Such enthusiasm for the dotted-board model can be attributed to its high suitability for instances with many identical items, since the number of variables depends on the number of item types rather than on the total number of items. Among the models of the third category, we also mention those of Baldacci et al. (2014), which use the raster method and were proposed for the irregular bin packing and knapsack problems with defects. Because of the high resolution they chose for the raster representation (from 512×512 to 2048×2048), the number of variables and constraints involved in the resulting ILP models was too large to solve with state-of-the-art solvers of that time (and probably with those of today as well), which is why they developed a heuristic approach based on a relaxation of their models.

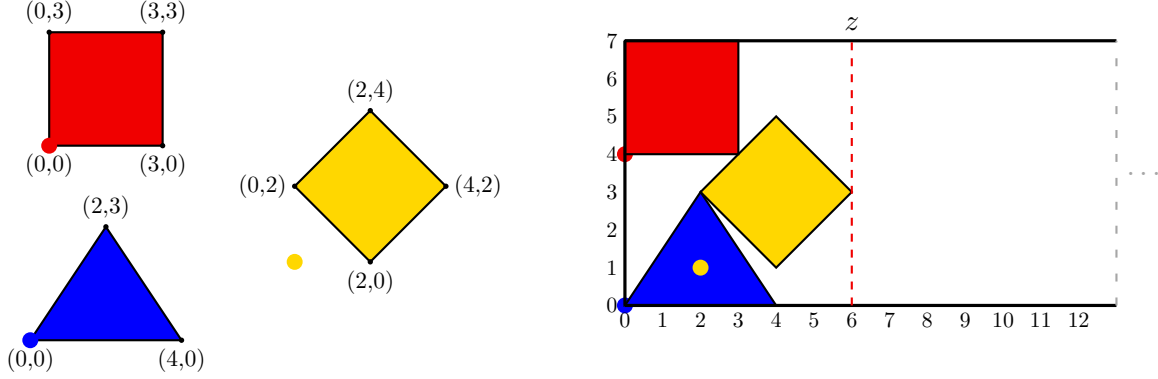
We conclude this section by summarizing the main findings concerning the empirical behavior of the dotted-board model and its improvements for the irregular strip packing problem. The experiments presented by Rodrigues and Toledo (2017) showed that the baseline version of the model could solve 39 out of 81 benchmark instances within 1 hour of computing time, compared with 45 for the clique-based version. These instances were chosen because they are “well-known benchmark instances (...) available at EURO Special Interest Group on Cutting and Packing (ESICUP) website.” In these experiments, the clique-based version of the dotted-board model consistently outperformed the baseline version. In the more recent work of Lastra-Díaz and Ortuño (2026), the reimplementations of the clique-based version could solve 47 instances from the same benchmark, compared with 56 for their tailored parallel branch-and-bound-and-check algorithm. Interestingly, the authors observed that the two approaches did not dominate each other: their tailored enumerative algorithm excelled on instances with few items overall, whereas the clique-based version of the dotted-board model performed better on instances with many items but few distinct item types.

3 Problem description and existing formulations

In the irregular strip packing problem, we are given a strip with a fixed width W and a set of item types $\mathcal{J} = \{1, 2, \dots, J\}$, where each type $j \in \mathcal{J}$ is described by a polygon and a demand d_j . The objective is to pack all required items into the strip to minimize its length z , ensuring that no items *overlap* and that all items *fit* within the boundaries of the strip. Formally, two items overlap if the intersection of their associated polygons has non-zero area. Similarly, an item fits within the boundaries of the strip if the geometric difference between the item and the strip has zero area. In the version of the problem studied here, item rotation is not allowed. We assume that the strip is positioned horizontally in the positive quadrant of the Cartesian coordinate system, with its bottom-left corner located at the origin $(0, 0)$ and its top-left corner at $(0, W)$. Each item is represented by an ordered set of vertices, where each pair of consecutive vertices forms an edge of the polygon. The coordinates of these vertices are given relative to a specific reference point. We assume hereafter that both the minimum vertex abscissa and the minimum vertex ordinate are 0 (if that is not the case, the vertex coordinates can be translated to achieve this), and that the reference point is $(0, 0)$. We define the *length* l_j of an item of type $j \in \mathcal{J}$ as its maximum vertex abscissa, and its *width* w_j as its maximum vertex ordinate. Figure 1 illustrates an example based on the benchmark instance *three* from the literature, featuring three item types depicted in red, blue, and yellow, respectively, each with a unit demand. The left portion of the

figure displays their ordered sets of vertices relative to a reference point at $(0, 0)$, which is depicted as a dot in the item's color. The red item has length $l_1 = 3$ and width $w_1 = 3$, the blue item has $l_2 = 4$ and $w_2 = 3$, and the yellow item has $l_3 = 4$ and $w_3 = 4$. A feasible packing is shown on the right, where the red item is said to be *packed* at position $(0, 4)$, meaning that its reference point is translated to $(0, 4)$, thereby occupying the polygon formed by the vertices $(0, 4)$, $(0, 7)$, $(3, 7)$, and $(3, 4)$ in the strip. Following the same terminology, the blue item is packed at position $(0, 0)$, and the yellow item at $(2, 1)$.

Figure 1: Example based on the benchmark instance *three*



3.1 The dotted-board model

In order to use the dotted-board model, the set of packing positions must be discretized to obtain a finite set. To do so, an integer upper bound L must be given for the strip length, and every item must be packed at a position (x, y) such that $x, y \in \mathcal{N}_0$, with $x \leq L-1$ and $y \leq W-1$. Note that a finer grid resolution can be achieved, for example, by multiplying W , L , and all item vertex coordinates by the same integer coefficient. While the literature typically defines a set of packing positions for each item type, to streamline the presentation of our proposed approaches, we instead introduce a unified set of *packing candidates* $\mathcal{P} = \{1, \dots, P\}$. Each packing candidate corresponds to the placement of a specific item of type $j \in \mathcal{J}$ at a specific position (x, y) . For a given packing candidate $p \in \mathcal{P}$, we denote its item type as $j(p)$, its packing abscissa as $a(p)$, its packing ordinate as $o(p)$, and its packing position $(a(p), o(p))$ as $c(p)$. Note that we restrict ourselves to the set of feasible packing candidates; that is, if $p \in \mathcal{P}$, placing an item of type $j(p)$ at position $c(p)$ fits entirely within the boundaries of the strip. Figure 2 illustrates the complete set of packing candidates for our running example with the length upper bound set to $L = 7$. For clarity, each packing candidate p is plotted at its position $c(p)$ and color-coded according to its item type $j(p)$. In total, there are 61 feasible packing candidates displayed in the figure.

Two packing candidates p and p' are said to be *incompatible* if packing an item of type $j(p)$ at position $c(p)$ overlaps with packing an item of type $j(p')$ at position $c(p')$. We collect all pairs of incompatible packing candidates in the set $\mathcal{I}$. In the running example, for instance, packing an item of type 3 (yellow) at position $(3, 3)$ is incompatible with both packing an item of type 1 (red) at position $(1, 4)$ and packing an item of type 2 (blue) at position $(3, 1)$. However, the latter two packing candidates are mutually compatible (see Figure 3). Hence, if we denote a packing candidate p by the tuple $(j(p), c(p))$, the pairs $\left((3, (3, 3)), (1, (1, 4))\right)$ and $\left((3, (3, 3)), (2, (3, 1))\right)$ both belong to $\mathcal{I}$, whereas $\left((1, (1, 4)), (2, (3, 1))\right)$ does not. In total, there are 1266 pairs of incompatible packing candidates in $\mathcal{I}$ in the running example.

Using a non-negative continuous variable z to keep track of the strip length and binary decision

Figure 2: Feasible packing candidates for instance *three* with $L = 7$

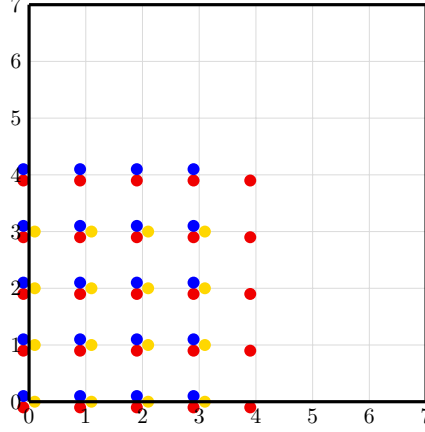
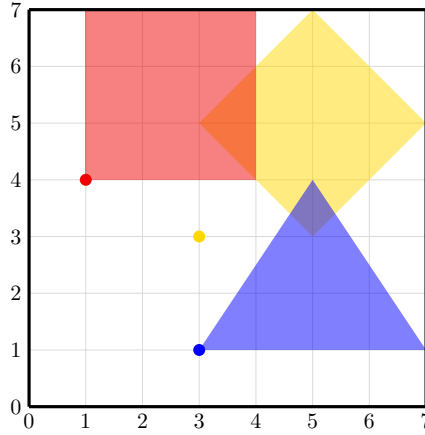


Figure 3: Example of (in)compatible packing candidates for instance *three*



variables x_p taking value 1 if packing candidate $p \in \mathcal{P}$ is used in the solution and value 0 otherwise, the dotted-board model for the irregular strip packing is as follows:

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

$$\text{s.t.} \quad x_p(a(p) + l_{j(p)}) \leq z \quad p \in \mathcal{P}, \tag{2}$$

$$\sum_{\substack{p \in \mathcal{P} \\ j(p)=j}} x_p = d_j \quad j \in \mathcal{J}, \tag{3}$$

$$x_p + x_{p'} \leq 1 \quad (p, p') \in \mathcal{I}, \tag{4}$$

$$x_p \in \{0, 1\} \quad p \in \mathcal{P}, \tag{5}$$

$$z \geq 0. \tag{6}$$

Objective function (1) minimizes the strip length, while constraints (2) ensure the strip is long enough to accommodate any selected packing candidate. Constraints (3) guarantee that the demand for each item type is satisfied, and constraints (4) prevent overlaps. Rodrigues and Toledo (2017) showed that constraints (2) could be replaced by

$$\underline{z} + x_p(a(p) + l_{j(p)} - \underline{z}) \leq z \quad p \in \mathcal{P}_{>\underline{z}}, \tag{7}$$

where $\underline{z}$ is a valid lower bound on the optimal solution and $\mathcal{P}_{>\underline{z}}$ is the subset of packing candidates

that, if used, would require a strip of length strictly greater than $\underline{z}$ (in other words, $p \in \mathcal{P}_{>\underline{z}}$ if $a(p) + l_{j(p)} > \underline{z}$). In the example, if $\underline{z} = 6$ (the lower bound value reported by Rodrigues and Toledo (2017) for their tailored bounding procedure), there are only 14 constraints of type (7), compared to 61 for constraints (2).

3.2 The dotted-board model with clique covering constraints

After highlighting that the pairwise incompatibility constraints (4) were both numerous and weak, Rodrigues and Toledo (2017) suggested replacing them with clique constraints of the form

$$\sum_{p \in q} x_p \leq 1 \quad q \in \mathcal{Q}(\mathcal{E}), \quad (8)$$

where $\mathcal{Q}(\mathcal{E})$ is an *edge clique cover* of the conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (i.e., a set of cliques whose edges cover $\mathcal{E}$), with $\mathcal{V}$ being the set of packing candidates $\mathcal{P}$ and $\mathcal{E}$ being the set of pairs of incompatible packing candidates $\mathcal{I}$. Since the number of constraints (8) depends on the size of the edge clique cover, the objective is to find a cover containing as few cliques as possible. However, finding a minimum edge clique cover is known to be $\mathcal{NP}$ -hard (Kou et al., 1978), and preliminary experiments indicate that, in practice, it is often computationally easier to solve an irregular strip packing instance to optimality than to find a minimum edge clique cover of its corresponding conflict graph. This likely explains why Rodrigues and Toledo (2017) used a heuristic procedure (later formalized by Rodrigues (2021)) to determine $\mathcal{Q}(\mathcal{E})$.

Similarly, Rodrigues and Toledo (2017) argued that an analogous procedure could also be applied to constraints (2), replacing them with clique constraints of the form

$$\sum_{p \in q} x_p (a(p) + l_{j(p)}) \leq z \quad q \in \mathcal{Q}(\mathcal{V}), \quad (9)$$

where $\mathcal{Q}(\mathcal{V})$ is, in this case, a *vertex clique cover* of the same conflict graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (i.e., a set of cliques whose vertices cover $\mathcal{V}$). Since finding a minimum vertex clique cover of $\mathcal{G}$ is equivalent to coloring the complement of $\mathcal{G}$ with a minimum number of colors, this problem is also $\mathcal{NP}$ -hard (Karp, 1972). We conclude this section by noting that, empirically, finding a minimum vertex clique cover is easier than finding a minimum edge clique cover, and that clique constraints can also be used to replace (7).

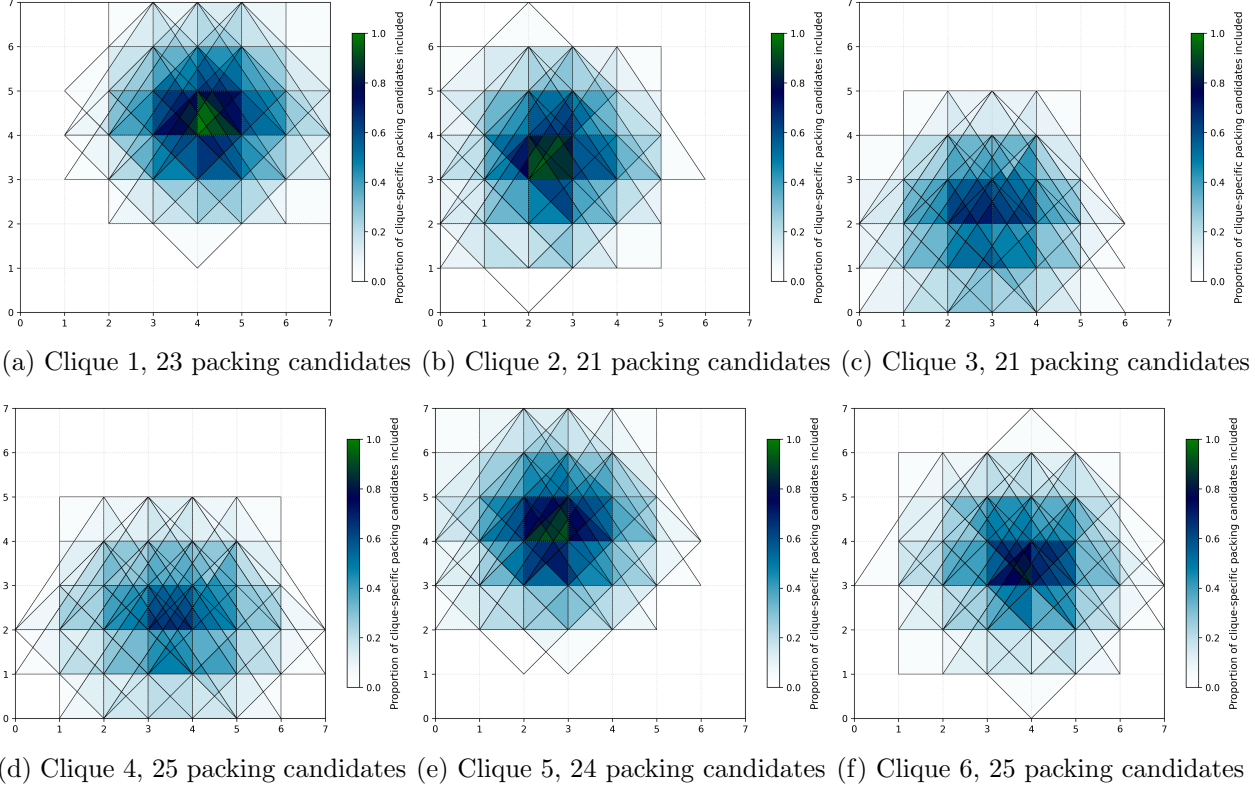
4 A mosaic approach

Although the conflict graph of instance *three* contains only 61 vertices and 1266 edges, we could not compute an optimal solution to the edge clique cover problem. We therefore settled for a heuristic approach, running the ECC8 software library of Conte et al. (2020) one thousand times on this instance, and eventually finding that these incompatible packing candidates could be covered using only 23 clique constraints of type (8).

We now investigate the relationship between a clique in this high-quality edge clique cover and the physical overlap of the polygons associated with the packing candidates contained in the clique. For brevity, a packing candidate $p \in \mathcal{P}$ will henceforth also refer to its associated polygon (i.e., the polygon obtained when packing an item of type $j(p)$ at position $c(p)$). We provide in Figure 4 a visual representation of 6 cliques identified by ECC8 for the running instance, where, for each clique, we depict the packing candidates it involves. Because many of these packing candidates overlap,

we partition the strip into disjoint *fragments*, coloring each fragment according to the proportion of packing candidates in the clique that cover it. Fragments not covered by any packing candidate are shown in white, whereas fragments covered by every packing candidate in the clique are shown in green (a gradient of blue is used for intermediate cases).

Figure 4: Visual representation of 6 out of the 23 cliques identified by ECC8 for instance *three*

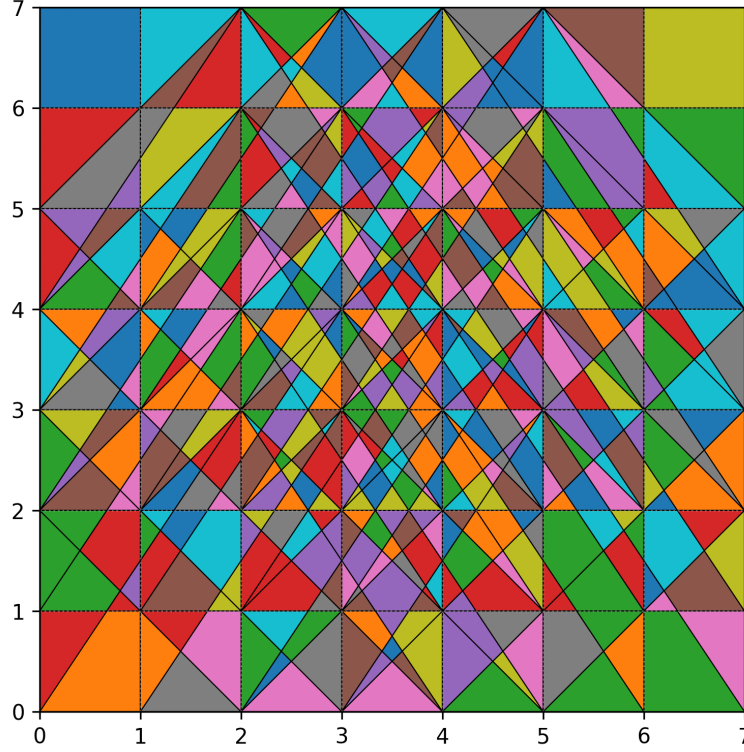


As one might expect, there appears to be a strong relationship between a clique identified in a high-quality edge clique cover and the physical overlap of the packing candidates it involves. For example, all 23 packing candidates involved in the first clique constraint share at least one common fragment, as do all 21 packing candidates of the second clique constraint and all 25 packing candidates of the sixth clique constraint.

Such a relationship has long been used (albeit implicitly) in mathematical models for the regular strip packing problem, where all items are rectangular. In that setting, one of the existing modeling strategies (originally proposed by Beasley (1985)) partitions the strip into unit squares and associates a binary decision variable with each packing candidate, much like in the dotted-board model. The key distinction, however, is that any physical overlap between two (regular) packing candidates necessarily covers at least one unit square, which allows non-overlap constraints to be modeled effectively by enforcing that each of the $W \times L$ unit squares is occupied by at most one of the selected candidates.

In the irregular case, on the other hand, an overlap between two packing candidates covers one or more disjoint fragments, a fragment being typically much smaller than a unit square, as shown in Figure 4. That said, the total number of disjoint fragments for a given instance remains finite. For example, after applying a procedure similar to the one used to create Figure 4—but this time using all 61 packing candidates—we obtain a set of 532 disjoint fragments covering the entire strip for instance *three*. This set, referred to hereafter as a *mosaic*, is displayed in Figure 5. This mosaic features a number of interesting (albeit straightforward) properties.

Figure 5: Mosaic for instance *three* composed of 532 fragments



Proposition 1. *When using the dotted-board model for a given irregular strip packing instance, any subset of packing candidates covering the same disjoint, non-degenerate fragment of the mosaic forms a clique in the conflict graph.*

Proof. Let f be a disjoint, non-degenerate fragment of the instance mosaic $\mathcal{M}$. If we denote the subset of packing candidates covering f by $\mathcal{P}_f$, then, by definition, any two distinct packing candidates $p, p' \in \mathcal{P}_f$ must physically overlap. As a result, they are mutually incompatible, which implies that (p, p') is an edge in the instance conflict graph $\mathcal{G}$. Because this holds for every pair of distinct elements in $\mathcal{P}_f$, this subset forms a clique in $\mathcal{G}$. $\square$

Proposition 2. *When using the dotted-board model for a given irregular strip packing instance, the collection of cliques formed by all disjoint fragments of the mosaic constitutes an edge clique cover of the conflict graph.*

Proof. Let (p, p') be an edge in the instance conflict graph $\mathcal{G}$. By definition, packing candidates p and p' physically overlap. Since this overlap necessarily covers at least one disjoint fragment, say f , of the instance mosaic $\mathcal{M}$, the clique formed by f covers the edge (p, p') . Because this holds for every edge in the conflict graph, the collection of cliques formed by all fragments $f \in \mathcal{M}$ constitutes an edge clique cover of $\mathcal{G}$. $\square$

An alternative interpretation of the mosaic is as a geometric hypergraph representation of the packing problem. Recall that, whereas an edge in an ordinary graph connects two vertices, a hyperedge in a hypergraph may connect any number of vertices. In our setting, the vertices are the packing candidates, and each non-degenerate mosaic fragment $f \in \mathcal{M}$ naturally defines a hyperedge $\mathcal{P}_f \subseteq \mathcal{P}$, containing all packing candidates that cover that fragment. The mosaic can therefore be seen as inducing a hypergraph in which each hyperedge identifies a group of placements that compete for

the same physical region of the strip. Since any two candidates belonging to the same $\mathcal{P}_f$ overlap on f , every hyperedge induces a clique in the conflict graph. Conversely, every conflict edge belongs to at least one such clique, because two incompatible packing candidates necessarily overlap on at least one mosaic fragment. Hence, the usual conflict graph can be obtained from this hypergraph by connecting every pair of vertices that occur together in at least one hyperedge; formally, the conflict graph is the 2-section (or primal graph) of this geometric hypergraph. From this perspective, the mosaic approach does not merely search for cliques in an abstract conflict graph: it preserves and exploits the underlying geometric structure that generates those conflicts.

For the running example, using the edge clique cover derived from the instance mosaic results in 532 clique constraints of type (8), a number that is admittedly lower than the trivial solution of 1266 cliques (one for each edge) but still far from the 23 cliques that are known to be achievable. That said, not all of these 532 clique constraints are needed to form an edge clique cover. Consider, for example, the clique associated with the top-left fragment of the mosaic (the dark blue unit square): this fragment is overlapped by only a single packing candidate (the one that packs the square item at $(0, 4)$) and is therefore unnecessary since it does not cover any edge of the conflict graph.

In practice, the size of a *mosaic-induced edge clique cover* can be significantly reduced by solving a set covering problem. Using binary decision variables y_f taking the value 1 if the clique formed by fragment $f \in \mathcal{M}$ is selected in the solution and 0 otherwise, a minimum mosaic-induced edge clique cover can be determined using the following formulation:

$$\min \sum_{f \in \mathcal{M}} y_f \quad (10)$$

$$\text{s.t.} \quad \sum_{\substack{f \in \mathcal{M} \\ p, p' \in \mathcal{P}_f}} y_f \geq 1 \quad (p, p') \in \mathcal{E}, \quad (11)$$

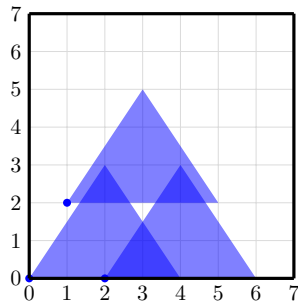
$$y_f \in \{0, 1\} \quad f \in \mathcal{M}. \quad (12)$$

The objective function (10) minimizes the number of selected cliques, whereas constraints (11) ensure that every edge in the conflict graph is covered by at least one selected clique. The left part of Figure 7 shows the 25 fragments associated with a minimum mosaic-induced edge clique cover for the running example. This number is strictly larger than the size of a minimum edge clique cover, for which 23 is a valid upper bound. This difference, albeit small, leads to the following proposition:

Proposition 3. *When using the dotted-board model for a given irregular strip packing instance, there may exist cliques in the conflict graph whose associated packing candidates do not all intersect in a common disjoint fragment of the mosaic.*

Proof. Consider three packing candidates p_1 , p_2 , and p_3 from instance *three*, with associated polygons defined by vertices $(0, 0)$, $(2, 3)$, and $(4, 0)$ for p_1 , $(1, 2)$, $(3, 5)$, and $(5, 2)$ for p_2 , and $(2, 0)$, $(4, 3)$, and $(6, 0)$ for p_3 , as shown in Figure 8. Although p_1 , p_2 , and p_3 form a clique in the conflict graph since $(p_1, p_2), (p_1, p_3), (p_2, p_3) \in \mathcal{E}$, they do not all intersect in a common disjoint fragment of the mosaic.

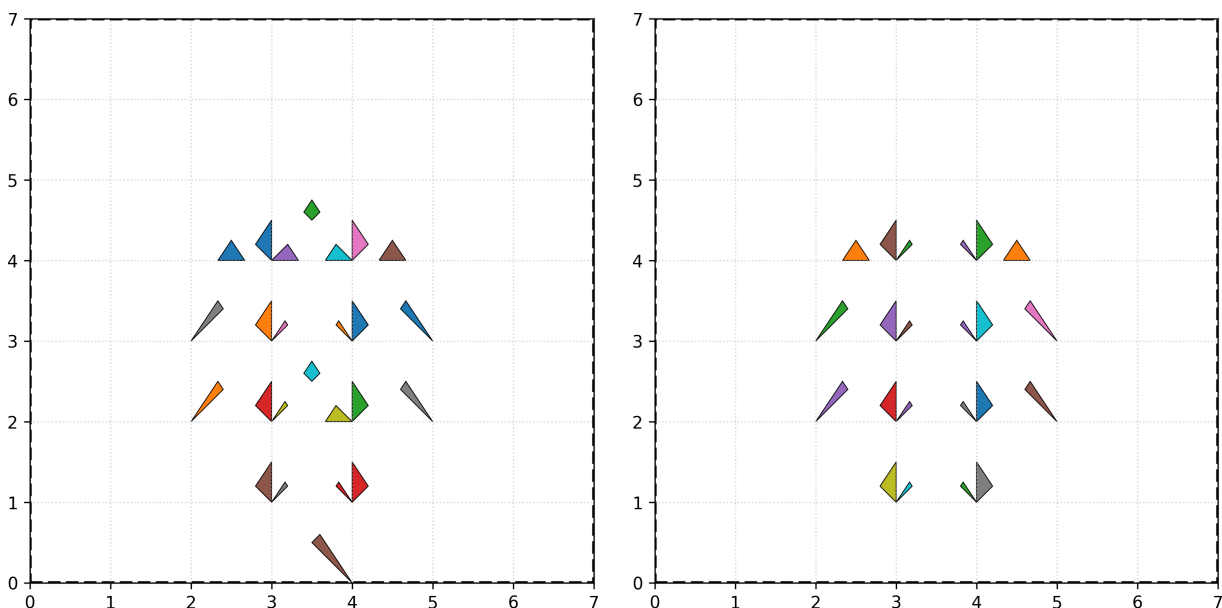
Figure 6: A clique whose packing candidates do not all share a common disjoint fragment



□

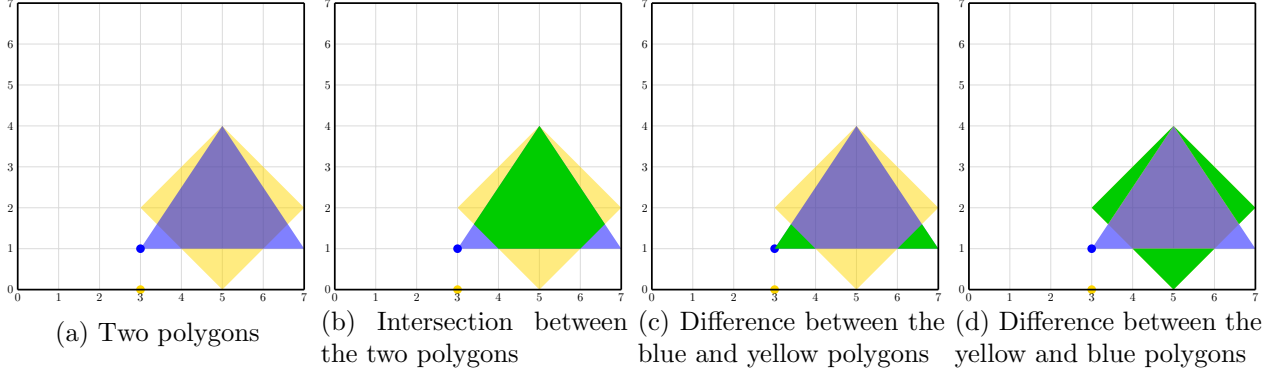
Despite this negative result, we argue throughout the rest of this work that using mosaic-induced edge clique covers offers several advantages. The first of these is the increased flexibility provided by constraints (12) when computing covers that are close (but not identical) to edge clique covers. One such cover that is relevant to the dotted-board model is the *edge clique cover with optional edges*, in which optional edges do not need to be covered, but may still be used to form cliques. Indeed, because constraints (3) of the model already ensure incompatibility between any pair of packing candidates involving the same item type when that type has unit demand, it is not necessary to cover any pair $(p, p') \in \mathcal{E}$ such that (i) $j(p) = j(p')$ and (ii) $d_{j(p)} = 1$, so these edges can be designated as optional. A minimum *mosaic-induced edge clique cover with optional edges* can be found simply by removing all constraints (11) corresponding to optional edges. On the other hand, to the best of our knowledge, existing approaches for the non-mosaic-induced counterpart (such as the ECC3 software library) do not support optional edges, and the trivial fix of removing all optional edges from the conflict graph does not work because it also eliminates cliques that would otherwise be feasible. The right part of Figure 7 shows the 22 fragments associated with a minimum mosaic-induced edge clique cover with optional edges for the running example.

Figure 7: Fragments associated with a minimum mosaic-induced edge clique cover without (left) and with (right) optional edges for instance *three*



We conclude this section by describing in Algorithm 1 the procedure used to construct the mosaics. To streamline the presentation of the strategies described in the next section, we assume that the dimensions of the mosaic, $\tilde{L} \times \tilde{W}$, are given as input to the algorithm and are not necessarily equal to the dimensions of the strip, $L \times W$. Polygon objects are used to represent both the mosaic fragments and the packing candidates. We also assume the availability of two functions: **Intersection**(poly1, poly2), which returns the intersection of two polygons, and **Difference**(poly1, poly2), which returns their geometric difference. The output of these functions may be empty, a single polygon, or a set of polygons (see Figure 8 for an example).

Figure 8: Example outputs (in green) of the intersection and difference functions



Algorithm 1 Decomposing a mosaic into a set of disjoint fragments

Input: $\mathcal{P}, \tilde{L}, \tilde{W}$

```

1:  $f_0 \leftarrow \{(0,0), (0, \tilde{W}), (\tilde{L}, \tilde{W}), (\tilde{L}, 0)\}$ 
2:  $\mathcal{P}(f_0) \leftarrow \emptyset$ 
3:  $\mathcal{M} \leftarrow \{f_0\}$ 
4: for  $p \in \mathcal{P}$  do
5:   if Intersection( $f_0, p$ ) is empty then
6:     continue
7:   if Difference( $f_0, p$ ) is empty then
8:     for  $f \in \mathcal{M}$  do
9:        $\mathcal{P}(f) \leftarrow \mathcal{P}(f) \cup \{p\}$ 
10:    continue
11:   $\mathcal{M}' \leftarrow \mathcal{M}$ 
12:  for  $f \in \mathcal{M}$  do
13:    if Intersection( $f, p$ ) is empty then
14:      continue
15:    if Difference( $f, p$ ) is empty then
16:       $\mathcal{P}(f) \leftarrow \mathcal{P}(f) \cup \{p\}$ 
17:      continue
18:     $\mathcal{M}' \leftarrow \mathcal{M}' \setminus \{f\}$ 
19:    for each fragment  $f_i$  in Intersection( $f, p$ ) do
20:       $\mathcal{P}(f_i) \leftarrow \mathcal{P}(f) \cup \{p\}$ 
21:       $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{f_i\}$ 
22:    for each fragment  $f_d$  in Difference( $f, p$ ) do
23:       $\mathcal{P}(f_d) \leftarrow \mathcal{P}(f)$ 
24:       $\mathcal{M}' \leftarrow \mathcal{M}' \cup \{f_d\}$ 
25:   $\mathcal{M} \leftarrow \mathcal{M}'$ 

```

Output: $\mathcal{M}$

5 A divide-and-conquer strategy

Lastra-Díaz and Ortuño (2026) argued that one of the main drawbacks of the dotted-board model with clique covering constraints proposed by Rodrigues and Toledo (2017) was the need for “a significant preprocessing time to compute edge (...) clique coverings”. So far, our proposed mosaic approach only replaces one preprocessing procedure with another, and although the two approaches are difficult to compare from a theoretical perspective, we observe in practice that constructing the entire mosaic can be time-consuming for medium- and large-sized benchmark instances, since both $|\mathcal{P}|$ and $|\mathcal{M}|$ increase quickly with W , L , and J . Note that, while using the mosaic approach also requires solving a set covering problem, which is $\mathcal{NP}$ -hard (Karp, 1972), optimal solutions can often be achieved in a short time for medium-sized instances, while high-quality solutions can be obtained using heuristics for large ones.

Instead of constructing the mosaic for the entire strip and solving a single set covering problem, one could construct a separate mosaic for each 1×1 tile and solve an individual set covering problem for each. This strategy decomposes a (potentially very) large problem into many smaller subproblems—a technique widely used in combinatorial optimization. From the perspective of the instance conflict graph $\mathcal{G}$, this approach corresponds to finding an edge clique cover for each of the $L \times W$ subgraphs of $\mathcal{G}$ associated with a tile, where a vertex is included in a given subgraph if its associated packing candidate overlaps with the corresponding tile, and an edge is included only if both of its endpoints are present.

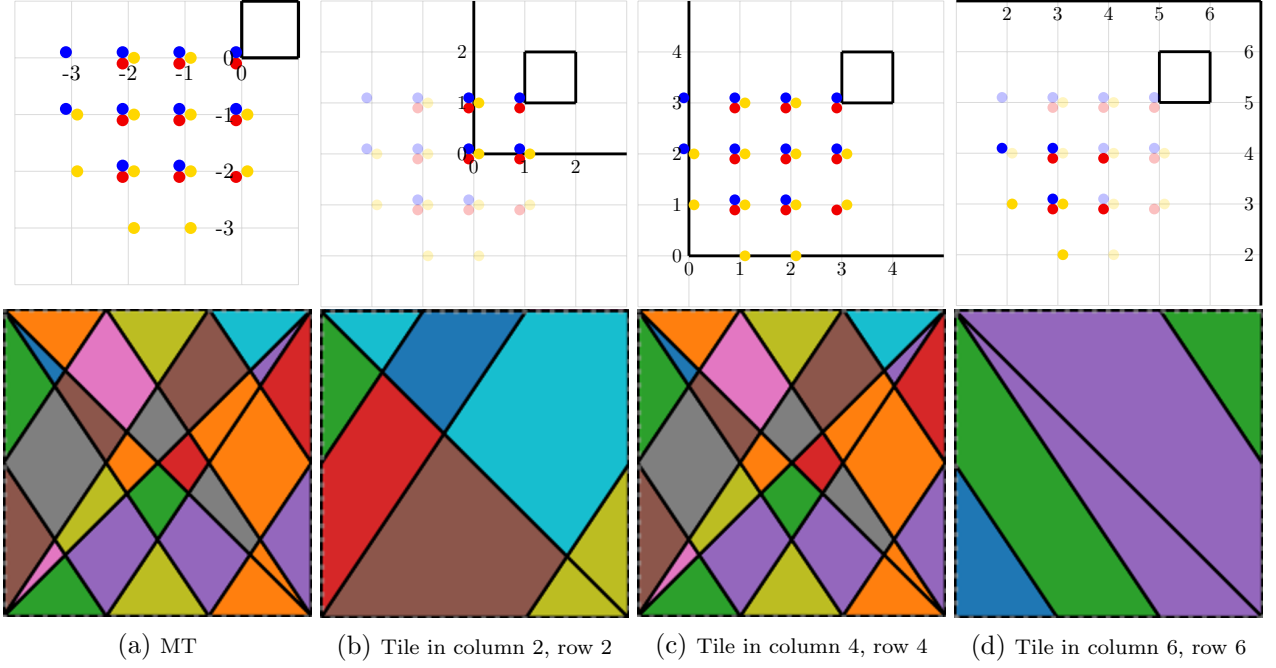
Note that any tile of the instance mosaic can be constructed using Algorithm 1 simply by plugging the coordinates of that tile into Step 1. An interesting feature of the mosaic, however, is that these 1×1 tiles are highly similar. Take, for example, the tile at the center of Figure 5 (in column 4, row 4), defined by the vertices $(3, 3)$, $(3, 4)$, $(4, 4)$, and $(4, 3)$. Not only is it identical to both the tile above it and the tile below it, but it is also the same (modulo a few fragment mergings) as the tiles on its left and right. In the following, we formalize this observation.

First, we introduce the concept of a 1×1 *master tile* (MT for short), which is essentially an artificial 1×1 tile constructed from the packing candidates of a strip with unrestricted dimensions and is, therefore, composed of the maximum possible number of fragments given the item types considered in the instance (see the bottom of Figure 9a). (We now omit the “ 1×1 ” qualifier for the sake of brevity.) In most instances, the mosaic includes at least one tile that shares the same configuration as the MT, and often many if the strip is large compared to the items. Conversely, if the strip is relatively small, the mosaic may not contain such a tile at all. In our running example, the tile at the center of the mosaic (see the bottom of Figure 9c) shares the same configuration as the MT. Note that the MT can also be constructed using Algorithm 1: after defining an arbitrary set of vertices for the tile—for example, $(0, 0)$, $(0, 1)$, $(1, 1)$, and $(1, 0)$ —one only needs to define the set $\mathcal{P}$ using coordinates relative to the tile instead of absolute coordinates on the strip (see the top of Figure 9a). We note that a concept analogous to the MT was recently used by Delorme and Malinowski (2026) to compute non-overlap constraints for circular objects in the context of a tower defense game.

Proposition 4. *When using the dotted-board model for a given irregular strip packing instance, any tile in the instance mosaic can be obtained from the MT by merging a subset of its fragments.*

Proof. Recall that both the MT and any tile t of the instance mosaic can be constructed using Algorithm 1, with the only difference lying in their respective sets of packing candidates. While the MT considers the complete set of all potential packing candidates $\mathcal{P}$ that could overlap an arbitrary tile (see the top of Figure 9a), the construction of tile t is restricted to a (transposed) subset $\mathcal{P}_t \subseteq \mathcal{P}$ consisting only of candidates that lie entirely within the strip boundaries (see the top of Figures 9b and 9d).

Figure 9: Packing candidates involved (solid) and not involved (faint) in the construction of a subset of tiles, with the resulting mosaic displayed below.



Since the order in which packing candidates are processed in Algorithm 1 does not affect the final fragments of a tile, we can analyze the construction of the MT in two consecutive phases. First, we process the subset of candidates $\mathcal{P}_t$, which returns exactly the fragment configuration of tile t . Second, we process the remaining candidates in $\mathcal{P} \setminus \mathcal{P}_t$. Each candidate processed in this second phase can only intersect existing fragments, thereby splitting them into smaller sub-fragments, but it can never remove an existing boundary established in the first phase. Consequently, the MT is a subdivision of tile t . It follows that tile t can be reconstructed from the MT by reversing this subdivision—namely, by merging the sub-fragments generated during the second phase. $\square$

We now define the concept of an *MT mosaic*, which is essentially a mosaic composed of $L \times W$ copies of the MT (see the left part of Figure 10).

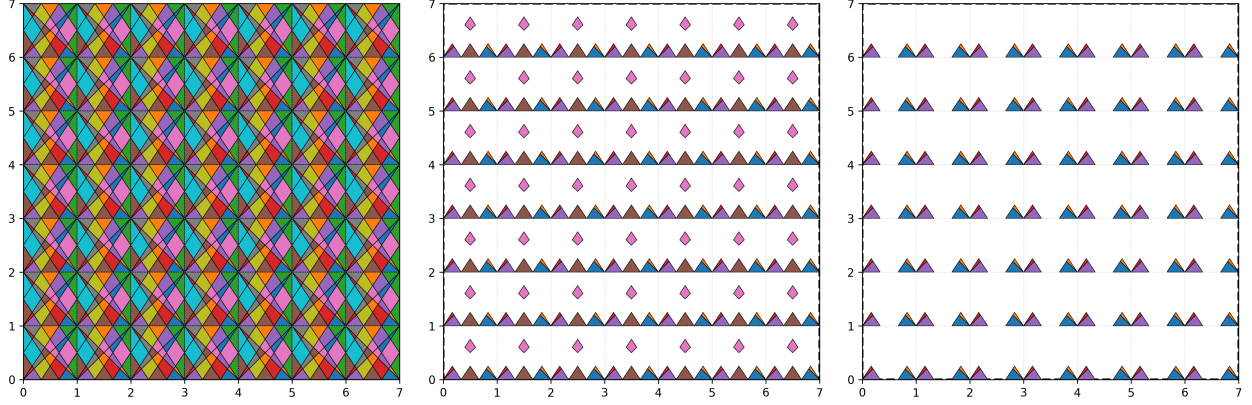
Proposition 5. *When using the dotted-board model for a given irregular strip packing instance, the collection of cliques formed by all disjoint fragments of the MT mosaic constitutes an edge clique cover of the conflict graph.*

Proof. This follows directly from Propositions 2 and 4. The latter ensures that each fragment of the mosaic matches (i.e., involves the exact same set of packing candidates as) at least one fragment of the MT mosaic, whereas the former establishes that the collection of cliques corresponding to all fragments of the mosaic forms an edge clique cover of the conflict graph. $\square$

In its current form, Proposition 5 is not particularly useful: in the running example, because the MT consists of 28 fragments, the MT mosaic contains $7 \times 7 \times 28 = 1372$ fragments (see the left part of Figure 10); this results in 1372 constraints of type (8), which is even greater than the 1266 initial pairwise incompatibility constraints. That being said, one can employ the exact same strategy described in Section 4 before duplicating the MT, namely, reducing the size of its edge clique cover (with or without optional edges) by solving a set covering problem. The resulting MT (which is not necessarily

unique) is hereafter referred to as an MT_{SC}^O when optional edges are considered, and as an MT_{SC} otherwise. In the running example, an MT_{SC} contains six fragments, resulting in 294 fragments in the MT_{SC} mosaic (see the middle part of Figure 10), while an MT_{SC}^O contains four fragments, resulting in 196 fragments in the MT_{SC}^O mosaic (see the right part of Figure 10).

Figure 10: MT mosaic (left), MT_{SC} mosaic (center), and MT_{SC}^O mosaic (right) for instance *three*.



5.1 Reducing the number of fragments in the MT

Using the MT mosaic offers several advantages over working with the standard mosaic. First, it is much faster to construct. Second, solving a set covering problem over the MT to obtain an MT_{SC}^O takes significantly less time than doing so over the standard mosaic. However, this speed comes at the cost of generating more constraints of type (8): in our running example, using the standard mosaic results in 22 clique constraints, compared to 196 for the MT_{SC}^O mosaic.

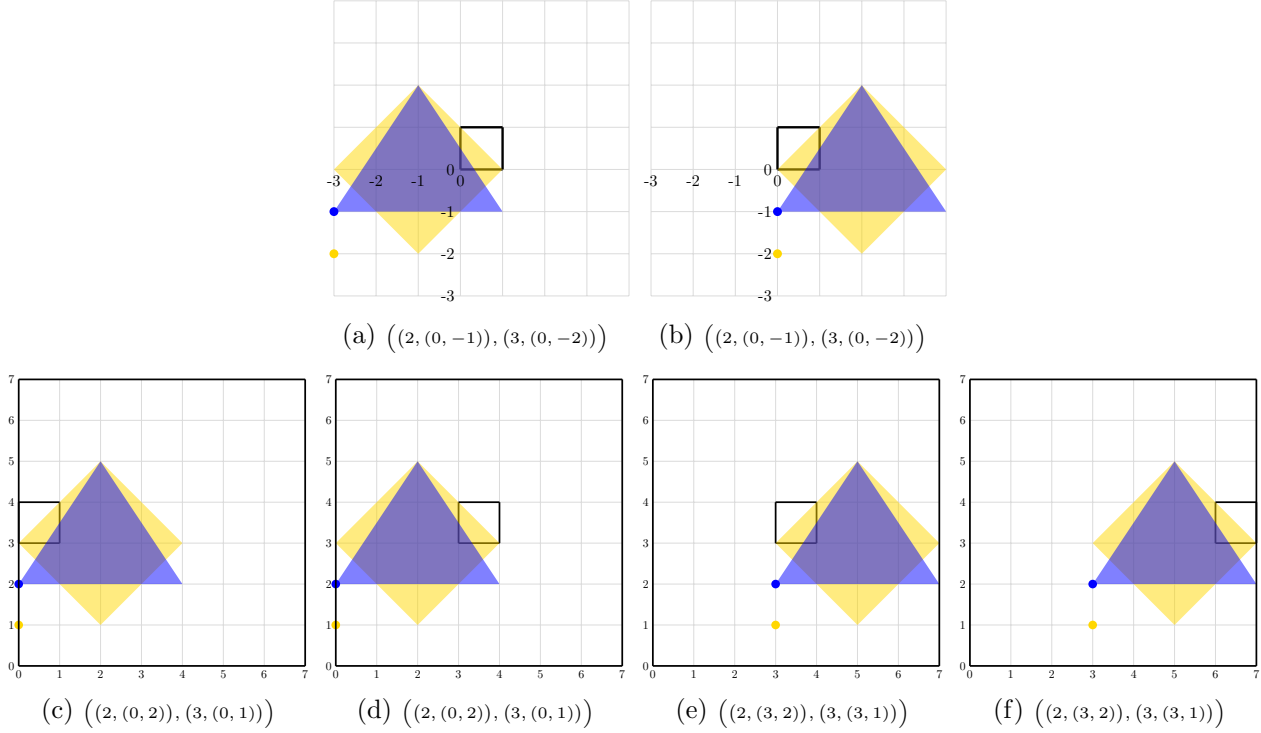
Further analysis shows that the number of fragments needed in an MT_{SC}^O can be reduced even more, primarily because the cliques associated with the resulting MT_{SC}^O mosaic end up covering the same edges of the conflict graph multiple times. Consider, for example, two relative incompatibilities in the MT: one between an item of type 2 (blue) at $(-3, -1)$ and an item of type 3 (yellow) at $(-3, -2)$ (Figure 11a), and another between an item of type 2 at $(0, -1)$ and an item of type 3 at $(0, -2)$ (Figure 11b). So far, our approach requires an MT_{SC}^O to cover both incompatibilities. Given the locations of the item overlaps (in the bottom-left and bottom-right corners of the MT, respectively), one fragment will be needed for each incompatibility. However, examining how the cliques of an MT_{SC}^O extend across specific tiles in the strip reveals redundant coverage:

- For the tile in column 1, row 4, these two incompatibilities translate into $\left((2, (-3, 2)), (3, (-3, 1))\right)$, which is ignored because it involves infeasible packing positions, and $\left((2, (0, 2)), (3, (0, 1))\right)$, as shown in Figure 11c.
- For the tile in column 4, row 4, they translate into $\left((2, (0, 2)), (3, (0, 1))\right)$ and $\left((2, (3, 2)), (3, (3, 1))\right)$, as shown in Figures 11d and 11e, respectively.
- For the tile in column 7, row 4, they translate into $\left((2, (3, 2)), (3, (3, 1))\right)$, as shown in Figure 11f, and $\left((2, (6, 2)), (3, (6, 1))\right)$, which is likewise ignored due to infeasible packing positions.

In other words, translating two relative incompatibilities into distinct tiles may result in the exact same “absolute” incompatibility, making one of them redundant. For example, translating

$\left((2, (-3, -1)), (3, (-3, -2))\right)$ to the tile in column 4, row 4, and $\left((2, (0, -1)), (3, (0, -2))\right)$ to the tile in column 1, row 4, both result in $\left((2, (0, 2)), (3, (0, 1))\right)$. We now show how to relax the covering constraints of relative incompatibilities in the MT so as to reduce redundant coverage when computing $\text{MT}_{\text{SC}}^{\text{O}}$.

Figure 11: Two relative incompatibilities in an $\text{MT}_{\text{SC}}^{\text{O}}$ (top) and their application to specific tiles in the strip (bottom).



We first define the concept of a *family* $k \in \mathcal{K}$ of MT incompatibilities: two MT incompatibilities (p_1, p_2) and (p'_1, p'_2) belong to the same family k if (i) $j(p_1) = j(p'_1)$ and $j(p_2) = j(p'_2)$, and (ii) $c(p_2) - c(p_1) = c(p'_2) - c(p'_1)$; that is, if both incompatibilities involve the same item types and p_2 has the same position relative to p_1 as p'_2 has relative to p'_1 . For example, $\left((2, (-3, -1)), (3, (-3, -2))\right)$ and $\left((2, (0, -1)), (3, (0, -2))\right)$ belong to the same family because both involve an item of type 2 and an item of type 3 (in that order), and the position of $(-3, -2)$ relative to $(-3, -1)$ (i.e., one unit below) is the same as that of $(0, -2)$ relative to $(0, -1)$. Our objective now is to find an *edge-family clique cover* of the MT, defined as a set of cliques whose edges cover at least one incompatibility from every family $k \in \mathcal{K}$.

We note that the size of an edge-family clique cover can also be reduced by solving a set covering problem: there is now one covering constraint of type (11) per family of MT incompatibilities rather than per edge, and the summation on the left-hand side includes all y -variables corresponding to fragments that cover at least one incompatibility of family k , rather than those covering a specific edge. We also emphasize that the concept of optional edges naturally extends to this context, giving rise to an *edge-family clique cover with optional families*. Indeed, if an edge $(p_1, p_2) \in \mathcal{E}$ is designated as optional (i.e., $j(p_1) = j(p_2)$ and $d_{j(p_1)} = 1$), then every edge belonging to the same family is likewise optional, and the constraint (11) associated with that family can be removed. The resulting reduced MT is hereafter referred to as an $\text{MT}_{\text{SC-F}}^{\text{O}}$ when optional edges are considered, and as an $\text{MT}_{\text{SC-F}}$ otherwise. To avoid confusion, we summarize the acronyms and main features of all MTs introduced

in this work in Table 1.

Table 1: List of MT acronyms and features.

Acronym	Fragment reduction (via set covering)	Optional edges	Conflict families
MT	✗	✗	✗
MT _{SC}	✓	✗	✗
MT _{SC} ^O	✓	✓	✗
MT _{SC-F}	✓	✗	✓
MT _{SC-F} ^O	✓	✓	✓

All five MTs discussed so far for the running example are shown in Figure 12. Although this relationship holds in the figure, we emphasize that an MT_{SC-F}^O need not be a subset of an MT_{SC-F}, an MT_{SC}^O, or an MT_{SC} in general.

Figure 12: The MT (left), an MT_{SC} (center-left), an MT_{SC}^O (center), an MT_{SC-F} (center-right), and an MT_{SC-F}^O (right) for instance *three*.

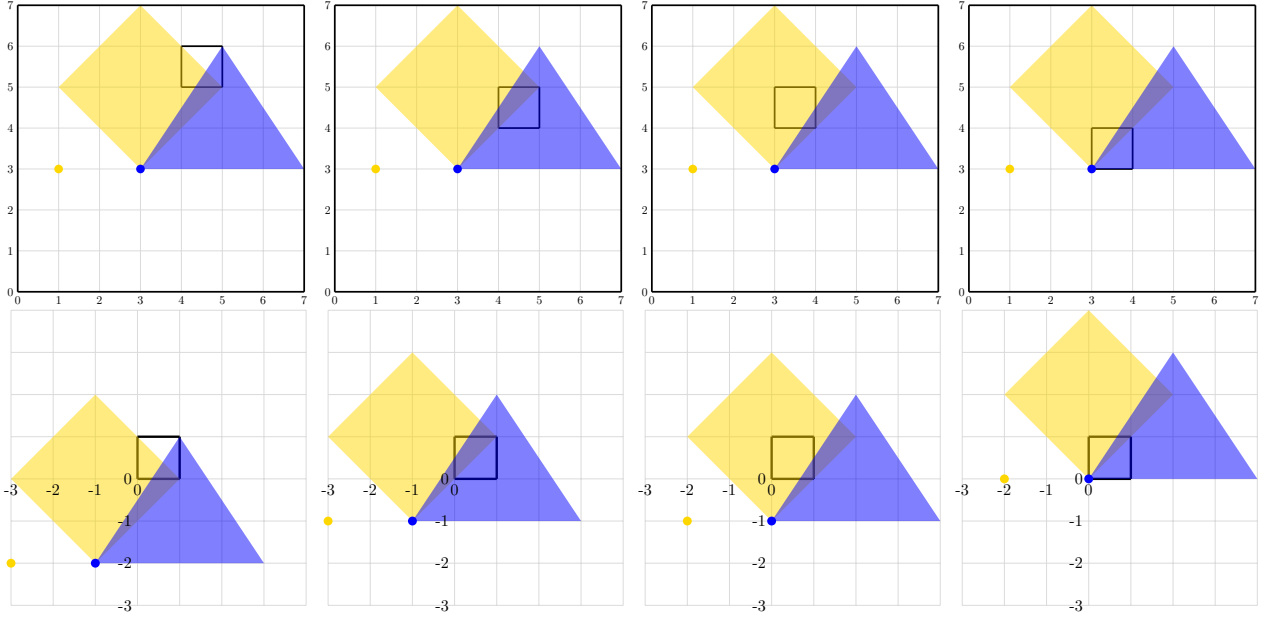


Proposition 6. *When using the dotted-board model for a given irregular strip packing instance, the collection of cliques formed by all disjoint fragments of an MT_{SC-F} mosaic constitutes an edge clique cover of the conflict graph.*

Proof. Let (p, p') be an edge in the conflict graph $\mathcal{G}$. By definition, the two packing candidates p and p' must physically overlap on a non-empty set of tiles $\mathcal{T}$ of the strip (i.e., $|\mathcal{T}| \geq 1$; see the top of Figure 13 for an example with edge $((2, (3, 3)), (3, (1, 3)))$, where $|\mathcal{T}| = 4$). For each tile $t \in \mathcal{T}$, let $c(p_t)$ and $c(p'_t)$ denote the relative positions of p and p' with respect to t . (In the example, $c(p_1) = (-1, -2)$, $c(p'_1) = (-3, -2)$, $c(p_2) = (-1, -1)$, $c(p'_2) = (-3, -1)$, $c(p_3) = (0, -1)$, $c(p'_3) = (-2, -1)$, $c(p_4) = (0, 0)$, and $c(p'_4) = (-2, 0)$; see the bottom of Figure 13.) By definition, the MT incompatibilities $((j(p), c(p_t)), (j(p'), c(p'_t)))$ for all $t \in \mathcal{T}$ belong to the same family $k \in \mathcal{K}$, since they involve the same item types and each p'_t has the same position relative to its corresponding p_t . Furthermore, no other incompatibilities can belong to this family (otherwise, the two packing candidates would physically overlap on an additional tile: such a tile cannot lie inside the strip by construction, nor can it lie outside the strip, since that would imply that both packing candidates are infeasible). Because an MT_{SC-F} contains at least one fragment whose corresponding clique covers at least one incompatibility from every family $k \in \mathcal{K}$, the edge (p, p') is necessarily covered in the resulting MT_{SC-F} mosaic. $\square$

The validity of using the collection of cliques formed by all disjoint fragments of an MT_{SC-F}^O mosaic in the dotted-board model follows directly from Proposition 6, since the only incompatibilities that may remain uncovered are those already forbidden by the demand constraints (3). In the remainder of this work, we focus on the most reduced versions of the mosaics, namely (i) the minimum mosaic-induced edge clique cover with optional edges, (ii) the MT_{SC}^O mosaic, and (iii) the MT_{SC-F}^O mosaic.

Figure 13: Set of tiles over which packing candidates $(2, (3, 3))$ and $(3, (1, 3))$ overlap (top) and the associated MT incompatibilities (bottom).



5.2 Reducing the number of packing candidates in the fragments of the MT

So far, we have suggested measures to reduce the number of MT fragments and hence the resulting number of constraints (8) needed in the dotted-board model. These measures relied on the fact that the cliques associated with the MT and $\text{MT}_{\text{SC}}^{\text{O}}$ mosaics ended up covering the same edges of the conflict graph multiple times, leading to the development of the $\text{MT}_{\text{SC-F}}^{\text{O}}$, which traded this coverage redundancy for a reduction in the required number of fragments. We now show how one can trade (some of) the remaining coverage redundancy in an $\text{MT}_{\text{SC-F}}^{\text{O}}$ to reduce the number of packing candidates included in its fragments, essentially reducing the number of variables involved in constraints (8) (also referred to as non-zero elements).

To illustrate how such a trade works, consider the following (theoretical) example with an $\text{MT}_{\text{SC-F}}^{\text{O}}$ composed of three fragments: $f_1 = \{p_1, p_2, p_3\}$, $f_2 = \{p_2, p_3, p_5\}$, and $f_3 = \{p_1, p_3, p_4\}$. These fragments involve five packing candidates, $p_1, \dots, p_5$, and seven incompatibilities: (p_1, p_2) , (p_1, p_3) , (p_1, p_4) , (p_2, p_3) , (p_2, p_5) , (p_3, p_4) , and (p_3, p_5) . While all three fragments are necessary to cover the seven incompatibilities, we note that incompatibilities (p_1, p_3) and (p_2, p_3) are each covered by two fragments. As a result, both incompatibilities would still be covered if packing candidate p_3 were removed from fragment f_1 . Therefore, one can trade the coverage redundancy of (p_1, p_3) and (p_2, p_3) for the removal of p_3 from f_1 , thereby reducing the number of non-zero elements in the resulting model by one.

Finding the maximum number of packing candidates that can be removed from the fragments of an $\text{MT}_{\text{SC-F}}^{\text{O}}$ is difficult in practice. We therefore propose a simple heuristic that inspects every packing candidate in each fragment of an $\text{MT}_{\text{SC-F}}^{\text{O}}$ and removes it whenever all incompatibilities between that candidate and the remaining elements in the fragment are already covered (either by other packing candidates in the same fragment or by other fragments).

5.3 Extending the approach to larger MTs

So far, we have considered two extreme cases: the mosaic formed by a 1×1 MT repeated $L \times W$ times, and the $L \times W$ mosaic formed by (what could be seen as) a single tile. While the former

requires less computational effort (for the running example, the 1×1 MT produced by Algorithm 1 contains only 28 fragments), it leads to a larger number of constraints of type (8) (98 in the $\text{MT}_{\text{SC-F}}^{\text{O}}$ mosaic). Conversely, the latter requires higher computational effort (the $L \times W$ mosaic contains 532 fragments), but results in fewer constraints of type (8) (22 in the version with optional edges). It is therefore natural to investigate intermediate cases between these two extremes, namely where an $l \times w$ MT is repeated $\lceil \frac{L}{l} \rceil \times \lceil \frac{W}{w} \rceil$ times, with $l \in \{2, \dots, \frac{L}{2}\}$ and $w \in \{2, \dots, \frac{W}{2}\}$.

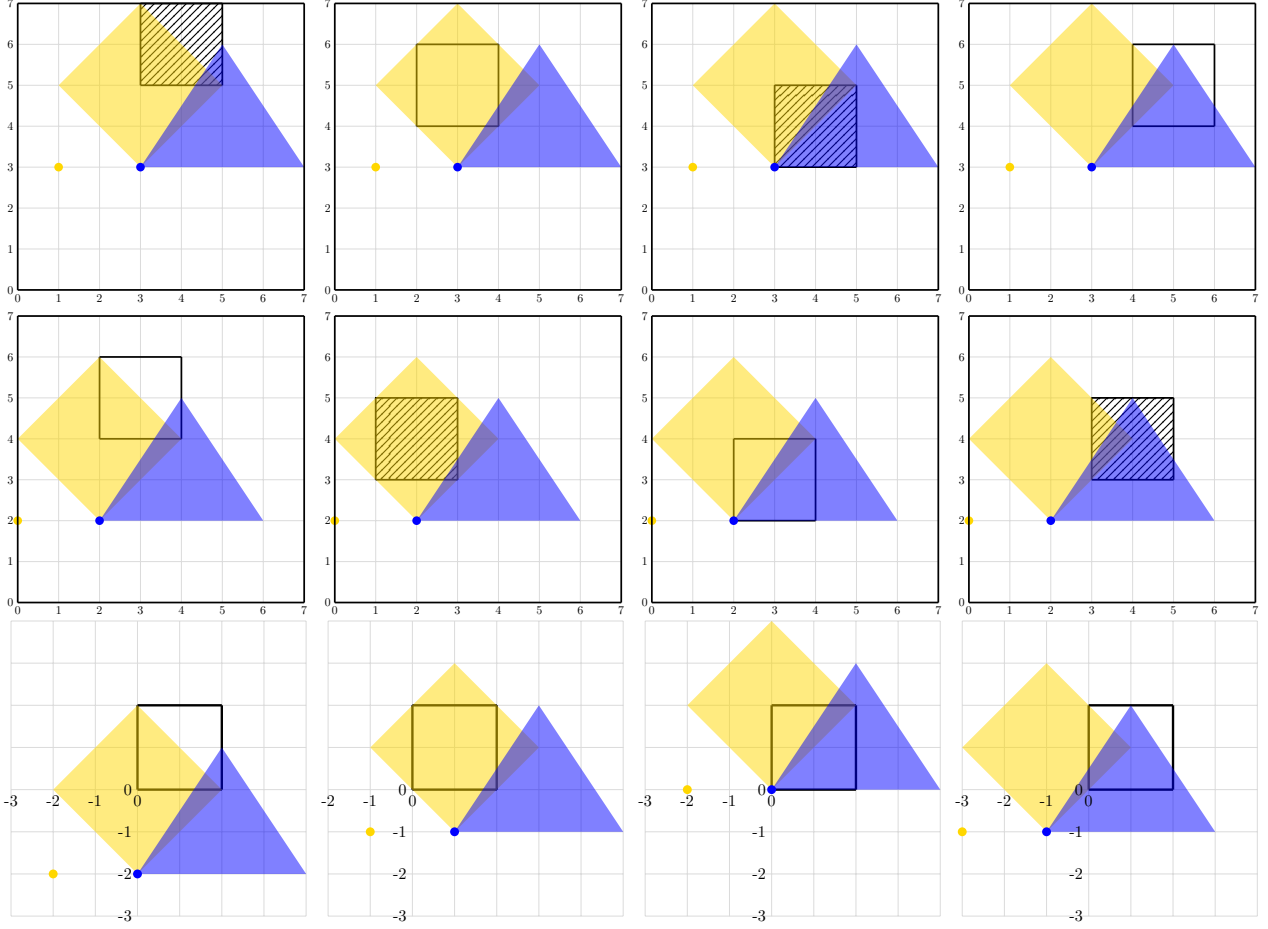
Whereas the procedure to construct an MT (via Algorithm 1) and the one to compute an $\text{MT}_{\text{SC}}^{\text{O}}$ (via model (10)–(12)) can be used for MTs of any size, constructing an $\text{MT}_{\text{SC-F}}^{\text{O}}$ requires extra care: for an $l \times w$ MT, a third condition is needed for two MT incompatibilities (p_1, p_2) and (p'_1, p'_2) to belong to the same family k , namely that $p'_1 - p_1 \equiv 0 \pmod{l}$ and $p'_2 - p_2 \equiv 0 \pmod{w}$. Note that if $l = w = 1$, this condition is always satisfied, which explains why it was not previously needed. An intuition as to why such an extra condition is required is given in Figure 14, where the bottom row displays four relative 2×2 MT incompatibilities: $\left((2, (0, -2)), (3, (-2, -2))\right)$, $\left((2, (0, 0)), (3, (-2, 0))\right)$, $\left((2, (1, -1)), (3, (-1, -1))\right)$, and $\left((2, (-1, -1)), (3, (-3, -1))\right)$, with the first two belonging to one family and the last two to another. Translating each of these into distinct 2×2 tiles as per the first row of the figure results in the exact same absolute incompatibility, $\left((2, (3, 3)), (3, (1, 3))\right)$. However, not all these 2×2 tiles will be considered in practice: if one starts from the tile with bottom-left corner $(0, 0)$, then the tiles with bottom-left corners $(2, 4)$ (second column) and $(4, 4)$ (fourth column) will be considered, whereas those with bottom-left corners $(3, 5)$ (first column) and $(3, 3)$ (third column) will not. That explains why only one of $\left((2, (1, -1)), (3, (-1, -1))\right)$ and $\left((2, (-1, -1)), (3, (-3, -1))\right)$ is needed, but also why neither can be replaced by $\left((2, (0, -2)), (3, (-2, -2))\right)$ or $\left((2, (0, 0)), (3, (-2, 0))\right)$. Similarly, translating the four relative incompatibilities into distinct 2×2 tiles as in the second row of the figure results in the same absolute incompatibility, in this case $\left((2, (2, 2)), (3, (0, 2))\right)$. Using the same reasoning, this illustrates why only one of $\left((2, (0, -2)), (3, (-2, -2))\right)$ and $\left((2, (0, 0)), (3, (-2, 0))\right)$ is required, but also why neither can be replaced by $\left((2, (1, -1)), (3, (-1, -1))\right)$ or $\left((2, (-1, -1)), (3, (-3, -1))\right)$.

We display both an $\text{MT}_{\text{SC}}^{\text{O}}$ and an $\text{MT}_{\text{SC-F}}^{\text{O}}$ mosaic for each MT size of 1×1 , 2×2 , 3×3 , and 4×4 for the running example in Figure 15. As can be seen, while the number of fragments decreases in the $\text{MT}_{\text{SC}}^{\text{O}}$ as the MT size increases, this is not the case for the $\text{MT}_{\text{SC-F}}^{\text{O}}$, where the number of fragments remains constant. The computational experiments in Section 6 confirm that larger MT tiles are beneficial primarily when using the $\text{MT}_{\text{SC}}^{\text{O}}$.

6 Computational experiments

In this section, we first detail the experimental setup; we then evaluate the proposed approaches against existing dotted-board-based models; next, we analyze the effectiveness of various strategies aimed at reducing the number of packing candidates in the MT fragments (i.e., the number of non-zero elements in the model); we then empirically compare alternative structures for modeling the objective function; and finally, we conclude with a performance analysis evaluating the impact of randomness and upgraded computing resources.

Figure 14: Subset of 2×2 tiles over which packing candidates $(2, (3, 3))$ and $(3, (1, 3))$ overlap (top), candidates $(2, (2, 2))$ and $(3, (0, 2))$ overlap (middle), and the associated MT incompatibilities (bottom).

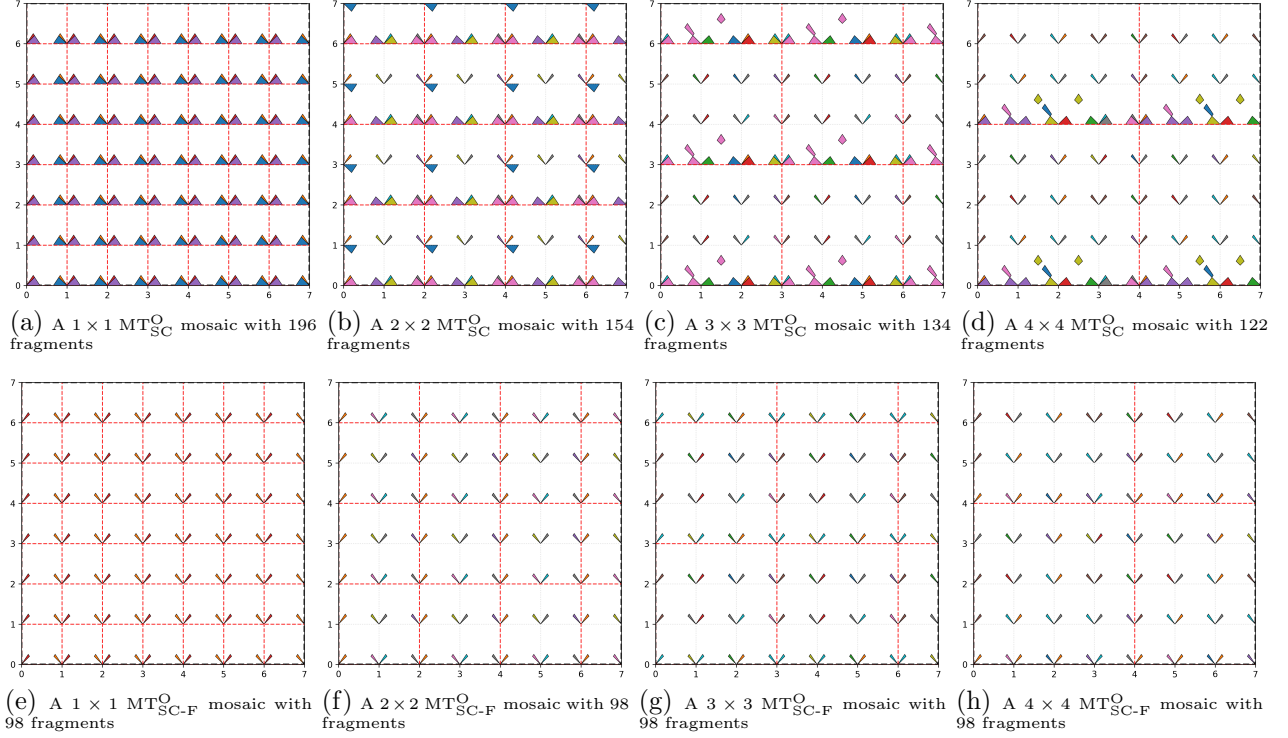


6.1 Experimental setup

Regarding the benchmark instances used in our experiments, we follow the strategy initially proposed by Rodrigues and Toledo (2017) and subsequently adopted by Lastra-Díaz and Ortuño (2026), which consists of selecting a specific subset of 81 instances from the literature spanning 12 categories: *tree*, *shapes*, *BLAZEWICZ*, *RCO*, *artif*, *shirts*, *dagli*, *fu*, *J1*, *J2*, *poly*, and *jakobs*. These instances are further described in Section A of the online supplement.

All tested approaches were implemented in C++, and the source code is available at <https://github.com/mdelorme2/ISPP-Mosaic>. All computational tests were executed on a virtual machine AMD EPYC-Rome Processor with 2.00 GHz and 64 GB of RAM, running under Ubuntu 20. The ILP models were solved using Gurobi 13.0.3. Unless stated otherwise, all tests were run on a single core with a 3600-second time limit per instance, excluding preprocessing time. The Clipper2 library (Johnson, 2025) was used to detect pairwise incompatibilities as well as to construct the mosaics. To prevent geometric errors such as missed intersections or false micro-overlaps, the precision parameter of Clipper2 was increased from the default 2 decimal places to 8 decimal places, and any polygon with an area below 10^{-8} units was considered degenerate and therefore discarded. For the sake of conciseness, we sometimes refer to "the performance of a model" in the following, which means the "performance obtained by solving the model with the specified ILP solver".

Figure 15: An $\text{MT}_{\text{SC}}^{\text{O}}$ (top) and an $\text{MT}_{\text{SC-F}}^{\text{O}}$ (bottom) mosaic for MTs of various sizes for instance *three*.



6.2 Evaluation of the proposed approaches against existing dotted-board-based models

We first evaluate the performance of the proposed approaches against (our own reimplementations of) existing dotted-board-based models. For all approaches, the simplest structure (i.e., (1) and (2)) was used to keep track of the strip length, and no techniques aimed at reducing the number of packing candidates in the fragments of the MT (i.e., the number of non-zero elements in the model) were applied. Various implementations of these two features are evaluated in subsequent experiments. The tested approaches in this set of experiments are listed below:

- **PAIRWISE**: the classical dotted-board model (1)–(6) introduced by Toledo et al. (2013);
- **CLIQUE $\times$ X**: the clique-based version of the dotted-board model (1)–(3), (5), (6), (8) proposed by Rodrigues and Toledo (2017). Following Lastra-Díaz and Ortuño (2026), we used the ECC8 software library of Conte et al. (2020) to compute a minimum edge clique cover. Since the output of the heuristic is non-deterministic, we tested two variants: one where the heuristic is executed once (**CLIQUE $\times$ 1**), and another where it is executed ten times, selecting the best outcome as the edge clique cover (**CLIQUE $\times$ 10**);
- **MOSAIC**: the clique-based version of the dotted-board model in which cliques are determined via a minimum mosaic-induced edge clique cover with optional edges. This approach is not expected to perform well, as computing the mosaic for the entire strip is both computationally expensive and memory-intensive. However, when it terminates, it provides a lower bound on the minimum number of fragments achievable with the proposed mosaic approaches.
- **MT-1 $\times$ 1**: the clique-based version of the dotted-board model in which cliques correspond to the

fragments of an $\text{MT}_{\text{SC-F}}^{\text{O}}$ mosaic. For the sake of comparison, we also test the variants expected to underperform (i.e., those based on an MT_{SC} and an $\text{MT}_{\text{SC}}^{\text{O}}$ mosaic) to assess whether the suggested improvements are truly beneficial.

- **MT- $\mathbf{X} \times \mathbf{X}$** : the clique-based version of the dotted-board model in which cliques correspond to the fragments of an $\text{MT}_{\text{SC-F}}^{\text{O}}$ mosaic where the initial MT produced by Algorithm 1 is larger. We tested two MT sizes, namely, 2×2 and 4×4 . Since little change in the total number of fragments could be observed for the $\text{MT}_{\text{SC-F}}^{\text{O}}$ mosaic, we also tested the underperforming $\text{MT}_{\text{SC}}^{\text{O}}$ mosaic variant to demonstrate that larger MTs are more relevant when MT incompatibility families are not considered.

We report in Table 2 the performance measures of the tested approaches across the 81 benchmark instances: the number of instances solved to optimality (column “# opt.”), the average total CPU time in seconds per run, including those that hit the time limit (“ T_t ”), the average preprocessing time (“ T_p ”), which consists of all the time spent before starting to solve the dotted-board-based model, the average lower (“LB”) and upper bounds (“UB”) obtained at the end of the time limit, and the number of instances for which a feasible solution could be found (“# feas.”). Additionally, we report the average number of variables (“# var.”), constraints (“# cons.”), and non-zero elements (“# nz.”) in the resulting dotted-board-based model. Because state-of-the-art ILP solvers feature effective internal routines aimed at reducing the model size, we also provide these last three metrics for the presolved model (i.e., after invoking `model.presolve()`). All metrics could be computed for all instances and all approaches, except for the upper bound, as some approaches occasionally reached the time limit without finding a feasible solution. In such cases, the upper bound was set to the strip length L to avoid reporting averages over different subsets of instances.

Table 2: Performance of the proposed approaches against existing dotted-board-based models.

Approach	Performance						Model size			Presolved model size		
	# opt.	T_t	T_p	LB	UB	# feas.	# var.	# cons.	# nz.	# var.	# cons.	# nz.
PAIRWISE	42	1849	0.2	19.75	22.42	81	3747	1980 229	3 964 186	3741	41 724	623 267
CLIQUEx1	48	1620	15.3	20.43	22.26	81	3747	6580	152 208	3726	3005	160 468
CLIQUEx10	46	1766	136.4	20.44	22.40	80	3747	6545	151 534	3725	2971	159 771
MT-1x1 ($\text{MT}_{\text{SC-F}}^{\text{O}}$)	44	1703	3	19.94	22.31	80	3747	7057	687 079	3730	3224	669 210
MT-1x1 ($\text{MT}_{\text{SC}}^{\text{O}}$)	43	1757	10.9	19.64	22.25	80	3747	15 017	2 562 037	3736	9963	2 471 760
MT-1x1 (MT_{SC})	43	1754	12.2	19.62	22.26	80	3747	15 417	2 689 727	3736	10 224	2 587 000

Probably the most striking outcome of these experiments is that **PAIRWISE** already performs relatively well, with 42 instances solved to optimality and a feasible solution found for every instance in the benchmark. A similar approach tested by Rodrigues and Toledo (2017) solved 39 instances to optimality and found a feasible solution for 62 instances. The improvements observed in our experiments can be attributed to a more modern experimental setup, specifically regarding ILP solvers, which have improved significantly in recent years (Clautiaux and Ljubić, 2025). Whereas the **PAIRWISE** ILP models are very large before presolve, averaging close to 2 million constraints and 4 million non-zero elements, Gurobi’s internal routines are particularly effective at reducing these values to slightly more than 40 thousand and 623 thousand, respectively.

Regarding raw efficiency, **CLIQUEx1** and **CLIQUEx10** appear to perform best, with the former solving two additional instances to optimality while requiring the ECC8 heuristic to be run only once, resulting in a much shorter average preprocessing time than the latter. Given the experimental setup (the same one-hour time limit per run, excluding preprocessing) and the nature of the two

approaches, we can safely attribute this performance difference to randomness. Note that the fact that randomness may play a role, even if only to a minor extent, in determining whether an approach is able to solve a specific instance to optimality has already been documented for other C&P problems (Akçay and Delorme, 2025, 2026). Considering that the model sizes (both before and after presolve) are extremely similar (with that of **CLIQUE** $\times$ **10** being smaller by design, but only by a thin margin) this suggests that there is little to be gained from running the **ECC8** heuristic more than once, especially given the relatively high cost of each run.

Focusing on our new approach, **MT** $_{1\times 1}$, it is clear that considering conflict families is beneficial, as evidenced by consistent improvements across almost all measured metrics (i.e., more instances solved to optimality, lower total computation and preprocessing times, tighter bounds, and smaller model sizes both before and after presolve) when this feature is applied. As far as optional edges are concerned, considering them results only in a very minor model reduction while slightly decreasing the preprocessing time (specifically, the set covering models are solved faster when this feature is included). Although the most effective variant of **MT** $_{1\times 1}$ requires roughly one-fifth of the preprocessing time of **CLIQUE** $\times$ **1**, it does not yet perform as well according to most metrics. This can be explained by the fact that, while the average number of constraints in **MT** $_{1\times 1}$ is relatively similar to that of **CLIQUE** $\times$ **1** (albeit slightly higher), the number of non-zero elements is more than four times larger, resulting in models that take longer to solve.

To conclude this first set of experiments, we report in Table 3 the number of non-overlapping constraints (dubbed “NOC”, recorded before presolve) for the **MOSAIC** approach across all instances for which the full mosaic could be constructed and the resulting minimum mosaic-induced edge clique cover with optional edges could be solved using the set covering formulation (10)–(12). For the sake of comparison, we also include this metric for **CLIQUE** $\times$ **10** and the **MT** $_{1\times 1}$, **MT** $_{2\times 2}$, and **MT** $_{4\times 4}$ variants based on an $\text{MT}_{\text{SC-F}}^{\text{O}}$ mosaic and an $\text{MT}_{\text{SC}}^{\text{O}}$ mosaic. For all mosaic-based approaches, we also report the MT efficiency (“ME”), computed as the number of fragments contained in the MT after solving the set-covering model divided by the number of 1×1 tiles contained in the MT. For example, the 2×2 $\text{MT}_{\text{SC}}^{\text{O}}$ has 12 fragments for instance *three* (as shown in Figure 15); hence, its MTE is equal to $\frac{12}{4} = 3$. Although the tile must be repeated 16 times to cover the entire strip, the number of NOC is not $12 \times 16 = 192$ but 154, as 38 fragments fall outside the strip. A more complete version of this table, which also includes the total computation time and preprocessing time, is available in Section B of the online supplement.

First and foremost, it is clear that **MOSAIC** is not viable on its own, as it can only terminate the preprocessing phase for 38 out of 81 instances. For the other instances, the 64 GB of RAM was insufficient either to generate the mosaic or to solve the resulting set covering problem (the process was always terminated due to an out-of-memory error during model creation or presolve). In the latter case, one could of course use a heuristic to determine which fragments to keep, but this would defeat the purpose of these experiments, which is to assess the minimum number of fragments (and hence non-overlap constraints) achievable with a mosaic approach.

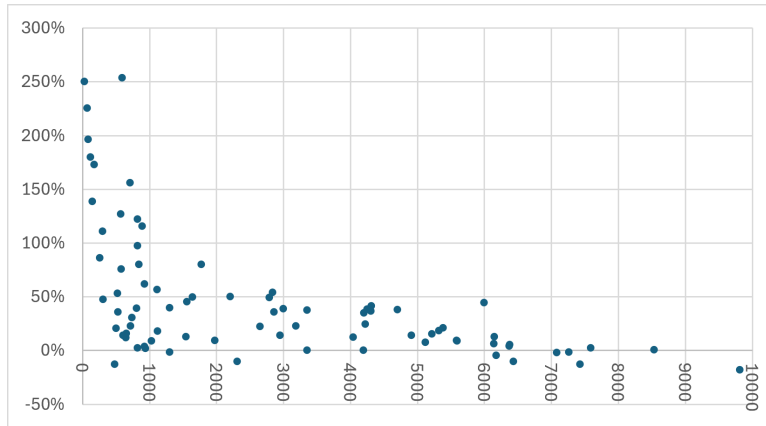
Second, it is interesting to note that there is still room for improvement when searching for a minimum edge clique cover, as the **ECC8** software library of Conte et al. (2020) returns covers that are sometimes much larger than those obtained by **MOSAIC**, especially for *shapes* instances. Although this can partially be attributed to the fact that **ECC8** does not support optional edges, this alone does not explain the discrepancy (for example, optional edges do not apply to *shapes* instances since every item demand is strictly greater than 1).

Table 3: Instance-specific performance metrics for the evaluated dotted-board-based approaches.

Instance	CLIQUE $\times 10$			MT $_{SC-F}^O$						MT $_{SC}^O$					
				MT $_{1\times 1}$		MT $_{2\times 2}$		MT $_{4\times 4}$		MT $_{1\times 1}$		MT $_{2\times 2}$		MT $_{4\times 4}$	
	NOC	NOC	ME	NOC	ME	NOC	ME	NOC	ME	NOC	ME	NOC	ME	NOC	ME
three	28	22	0.4	98	2	98	2	98	2	196	4	154	3	122	2.4
threep2	71	69	0.9	231	3	192	2.5	195	2.5	462	6	319	4	247	3.2
threep2w9	82	70	0.9	243	3	202	2.5	205	2.5	486	6	333	4	273	3.2
threep3	120	124	1.1	336	3	280	2.5	284	2.5	672	6	464	4	360	3.2
threep3w9	147	134	1.1	351	3	292	2.5	296	2.5	702	6	481	4	394	3.2
shapes4	296	201	0.6	624	2	624	2	624	2	936	3	708	2.3	648	2.1
shapes8	1007	552	1	1680	3	1120	2	1120	2	2240	4	1680	3	1400	2.5
SHAPES2	890	579	0.9	1920	3	1280	2	1280	2	2560	4	1920	3	1600	2.5
SHAPES4	2947	1396	1.2	3360	3	2240	2	2240	2	4480	4	3360	3	2800	2.5
SHAPES5	4192	1858	1.3	4200	3	2800	2	2800	2	5600	4	4200	3	3490	2.5
SHAPES7	6431	2716	1.4	5760	3	3840	2	3840	2	7680	4	5760	3	4800	2.5
blasz2	821	757	1.9	1616	4	1617	4	1617	4	6059	15	3638	9	2488	6.3
BLAZEWCZ1	258	246	2.1	480	4	480	4	480	4	1800	15	1048	8.8	692	5.9
BLAZEWCZ2	735	663	2.8	960	4	960	4	960	4	3840	16	2336	9.8	1560	6.7
BLAZEWCZ3	1120	969	2.9	1320	4	1320	4	1320	4	5280	16	3212	9.8	2139	6.7
BLAZEWCZ4	1542	1325	3	1740	4	1740	4	1740	4	6960	16	4230	9.8	2828	6.7
RCO1	176	197	1.6	480	4	420	3.5	420	3.5	960	8	628	5.3	502	4.3
RCO2	580	568	2.2	1020	4	900	3.5	892	3.5	2295	9	1467	5.8	1139	4.5
RCO3	926	900	2.4	1500	4	1320	3.5	1312	3.5	3375	9	2159	5.8	1673	4.5
RCO4	1112	1066	2.5	1740	4	1530	3.5	1522	3.5	3915	9	2505	5.8	1940	4.5
RCO5	1644	1564	2.5	2460	4	2160	3.5	2152	3.5	5535	9	3543	5.8	2741	4.5
artifl_2	571	622	2.9	1296	6	1296	6	1296	6	3024	14	2040	9.5	1628	7.6
artif2_4	1559	1489	3.9	2268	6	2268	6	2268	6	5292	14	3668	9.8	2939	7.8
fu5	216	283	0.4	1368	2	1368	2	1368	2	1368	2	1368	2	1368	2
j1-10-10-0	498	379	1.9	600	3	600	3	600	3	800	4	700	3.5	650	3.3
j1-10-10-1	657	512	2.7	760	4	760	4	760	4	950	5	850	4.5	800	4.3
j1-10-10-2	482	356	1.7	420	2	420	2	420	2	420	2	420	2	420	2
j1-10-10-3	604	435	1.9	690	3	690	3	690	3	1840	8	1200	5.3	918	4.1
j1-10-10-4	305	254	1.7	450	3	450	3	450	3	450	3	450	3	450	3
j1-12-20-0	644	486	2	720	3	720	3	720	3	960	4	840	3.5	780	3.3
j1-12-20-1	716	568	2.6	880	4	880	4	880	4	1100	5	980	4.5	920	4.3
j1-12-20-2	805	591	2.1	1120	4	1120	4	1120	4	1120	4	1120	4	1120	4
j1-12-20-3	523	446	2.2	800	4	800	4	800	4	1000	5	900	4.5	840	4.3
j1-12-20-4	940	719	2.2	960	3	960	3	960	3	1920	6	1440	4.5	1200	3.8
j1-14-20-0	819	591	2.1	840	3	840	3	840	3	1120	4	980	3.5	900	3.3
j1-14-20-1	1027	787	2.8	1120	4	1120	4	1120	4	1400	5	1260	4.5	1180	4.3
j1-14-20-3	530	395	1.6	720	3	720	3	720	3	720	3	720	3	720	3
j1-14-20-4	1300	943	2.9	1280	4	1280	4	1280	4	2560	8	1920	6	1600	5

Next, it appears that using larger MTs is not particularly beneficial when conflict families are considered, as doing so rarely reduces the number of non-overlap constraints needed while increasing both the preprocessing time and the memory consumption. Larger MTs seem more useful in the configuration where conflict families are not considered; however, as previously discussed, such a configuration is subpar.

Finally, although **MT $_{1\times 1}$** requires a higher number of non-overlap constraints than **CLIQUE $\times 10$** across most instances (72 out of 81), the relative overhead, computed as $\frac{NOC_{MT_{1\times 1}} - NOC_{CLIQUE\times 10}}{NOC_{CLIQUE\times 10}}$, decreases as $NOC_{CLIQUE\times 10}$ increases, sometimes even reaching a negative value, as shown in Figure 16.

Figure 16: Relative non-overlap constraint overhead of **MT $_{1\times 1}$** over **CLIQUE $\times 10$** as a function of the non-overlap constraint count in **CLIQUE $\times 10$** .

6.3 Evaluation of strategies for reducing non-zero elements

We briefly described in Section 5 a technique aimed at trading the coverage redundancy that is still present in an $\text{MT}_{\text{SC-F}}^{\text{O}}$ for a reduction in the number of packing candidates included in its fragments, which essentially reduces the number of non-zero elements in the resulting ILP model (the main drawback of **MT_1** $\times$ **1**). The steps of this technique are straightforward: while there exists at least one unprocessed fragment in the $\text{MT}_{\text{SC-F}}^{\text{O}}$, select a fragment to process, loop over each packing candidate included in that fragment, and remove a candidate whenever possible (i.e., if no incompatibility in the current $\text{MT}_{\text{SC-F}}^{\text{O}}$ is exclusively covered by that candidate). We evaluated various implementations of these steps. The first two considerations relate to the order in which fragments are processed. We tested ordering the fragments according to the number of packing candidates they involve (dubbed “p”) and according to the number of conflicts they cover that are not already covered by previously processed fragments (“c”). For each of these two criteria, we tested both non-increasing (“d”) and non-decreasing (“i”) orderings. The final consideration relates to the number of fragments left untouched (i.e., not processed at all), where we tested leaving 0, the last 1, and the last 2 fragments untouched.

We report in Table 4 the performance measures of **MT_1** $\times$ **1** after reducing the number of non-zero elements with each of these $2 \times 2 \times 3 = 12$ configurations. To evaluate the individual impact of each of the three considerations, we also include aggregated results in the table.

Table 4: Performance of **MT_1** $\times$ **1** after reducing the number of non-zero elements.

Configuration			Performance						Model size			Presolved model size		
p/c	d/i	0/1/2	# opt.	T _{tot}	T _{pre}	LB	UB	# feas.	# var.	# cons.	# nz.	# var.	# cons.	# nz.
p	d	0	43	1722	3	20.09	22.48	79	3747	7058	156 285	3732	2894	158 666
p	d	1	47	1567	3	20.40	22.30	81	3747	7058	221 618	3730	2984	221 893
p	d	2	46	1639	3.2	20.30	22.37	81	3747	7058	333 498	3731	3040	327 289
p	i	0	45	1713	3.1	20.21	22.52	81	3747	7058	145 598	3728	2858	148 554
p	i	1	48	1539	3.1	20.31	22.41	80	3747	7058	211 386	3726	2934	211 442
p	i	2	47	1614	3.1	20.32	22.32	80	3747	7058	328 759	3729	2983	322 336
c	d	0	44	1698	2.9	20.17	22.44	81	3747	7058	157 340	3732	2912	160 044
c	d	1	46	1603	2.9	20.51	22.27	80	3747	7058	222 988	3730	2963	223 138
c	d	2	48	1559	2.9	20.38	22.26	81	3747	7058	333 689	3731	3015	328 511
c	i	0	44	1772	3	20.25	22.48	80	3747	7058	142 740	3728	2838	146 090
c	i	1	47	1576	3	20.41	22.31	80	3747	7058	209 088	3726	2914	209 294
c	i	2	48	1584	3	20.20	22.27	81	3747	7058	325 356	3727	2956	319 585
p	-	-	276	1632	3.1	20.27	22.40	482	3747	7058	232 857	3729	2949	231 697
c	-	-	277	1632	3	20.32	22.34	483	3747	7058	231 867	3729	2933	231 110
-	d	-	274	1631	3	20.31	22.35	483	3747	7058	237 570	3731	2968	236 590
-	i	-	279	1633	3.1	20.28	22.38	482	3747	7058	227 155	3727	2914	226 217
-	-	0	176	1726	3	20.18	22.48	321	3747	7058	150 491	3730	2875	153 339
-	-	1	188	1571	3	20.40	22.32	321	3747	7058	216 270	3728	2949	216 442
-	-	2	189	1599	3.1	20.30	22.31	323	3747	7058	330 326	3730	2998	324 430

Interestingly, while the -/-/0 configurations result in the smallest model sizes, they also perform the worst. Comparing the -/d/- and -/i/- configurations, the latter solve more instances to optimality, whereas the former typically produce better lower bounds. Finally, comparing the p/-/- and c/-/- configurations, the aggregated results of the latter are slightly better than those of the former, though only by a very thin margin. Given these conclusions, but also considering the individual performance of each configuration, we decided to focus primarily on configuration p/i/1, while running limited additional tests on c/d/1 due to its strong lower and upper bounds.

6.4 Evaluation of alternative structures for modeling the objective function

In the initial version of the dotted-board model presented by Toledo et al. (2013), a non-negative continuous variable z was used to keep track of the strip length, together with a set of constraints of type (2) (one per packing candidate) to ensure that the strip is long enough to accommodate any selected packing candidate. Although a few improvements were proposed later on by Rodrigues and Toledo (2017) in the form of constraints (7) (utilizing an instance lower bound) and (9) (utilizing a vertex clique cover), few alternative modeling structures have been explored in the literature.

First, we show how our approaches can also be used to derive a vertex clique cover. Assuming that every packing candidate is involved in at least one pairwise incompatibility in $\mathcal{I}$ (a trivial assumption in practice), Proposition 5 implies that the collection of cliques formed by all disjoint fragments of the MT mosaic constitutes a vertex clique cover of the conflict graph, in addition to an edge clique cover. Following the procedures described in Section 5, the size of this vertex clique cover can be reduced by solving a set covering problem on the MT before duplicating it. The only modification concerns constraints (11), which must be updated to cover vertices instead of edges. Furthermore, the concept of a conflict family naturally extends to packing candidate families, where two candidates belong to the same family if they correspond to the same item type. After solving the set covering problem (which typically results in a single fragment), we can also apply a non-zero-element reduction: a packing candidate can be removed from a fragment if another candidate (whether within the same fragment or another, if multiple fragments are selected) involves the same item type. We therefore tested three variants of **MT_1×1**: one in which neither packing candidate families nor non-zero-element reduction was considered (**MT_1×1_OBJ1**), one in which only packing candidate families were used (**MT_1×1_OBJ2**), and one in which both were applied (**MT_1×1_OBJ3**).

Drawing inspiration from the regular strip packing literature (Akçay and Delorme, 2026; Castro and Oliveira, 2011; Cid-Garcia and Rios-Solis, 2021), we also attempted to solve **MT_1×1** using destructive bounds, a technique in which the strip length L is fixed to a valid lower bound $\underline{z}$ and iteratively increased until a feasible solution is found (noting that $\underline{z} + 1$ becomes a valid lower bound if no solution with value $\underline{z}$ exists). We refer to the resulting approach as **MT_1×1_OBJ4**. While destructive bounds have, to the best of our knowledge, never been applied to the dotted-board model for irregular strip packing, a similar concept was recently used in the parallel branch-and-bound-and-check algorithm of Lastra-Díaz and Ortuño (2026).

Finally, inspired by a recent model initially proposed for the bin packing problem with contiguity constraints (Akçay and Delorme, 2025), we introduce a new modeling structure in which the non-negative continuous variable z is replaced with a set of L binary variables, z_l ($l = 1, \dots, L$), one for each $1 \times W$ vertical strip. Variable z_l takes the value 1 if there are no items occupying the l -th vertical strip and 0 otherwise. The resulting dotted-board model becomes:

$$\min \quad L - \sum_{l=1}^L z_l \quad (13)$$

$$\text{s.t.} \quad x_p \leq 1 - z_{l(p)} \quad p \in \mathcal{P}, \quad (14)$$

$$\sum_{\substack{p \in \mathcal{P} \\ j(p)=j}} x_p = d_j \quad j \in \mathcal{J}, \quad (15)$$

$$\sum_{p \in q} x_p \leq 1 \quad q \in \mathcal{Q}(\mathcal{E}), \quad (16)$$

$$z_l \leq z_{l+1} \quad l = 1, \dots, L-1, \quad (17)$$

$$x_p \in \{0, 1\} \quad p \in \mathcal{P}, \quad (18)$$

$$z_l \in \{0, 1\} \quad l = 1, \dots, L, \quad (19)$$

where $l(p)$ denotes the rightmost vertical strip occupied by packing candidate p . The resulting **MT_1×1** approach can naturally be enhanced by replacing constraints (14) with

$$\sum_{p \in q} x_p \leq 1 - z_{l(q)} \quad q \in \mathcal{Q}(\mathcal{V}), \quad (20)$$

where $l(q)$ denotes the vertical strip occupied by the fragment corresponding to clique q . Note that constraints (20) are valid only when $\mathcal{Q}(\mathcal{V})$ is a mosaic-induced vertex clique cover, as $l(q)$ is undefined if clique q does not originate from a fragment. Furthermore, when packing candidate families are considered, only candidates with the minimum abscissa within a family can cover it, and if non-zero-element reduction is applied, one such candidate must be retained per family. **MT_1×1_OBJ5** incorporates enhanced constraints (20) along with both packing candidate families and non-zero-element reduction. Finally, in **MT_1×1_OBJ6**, we strengthen non-overlap constraints (16) to include the decision variables z_l as follows:

$$\sum_{p \in q} x_p \leq 1 - z_{l(q)} \quad q \in \mathcal{Q}(\mathcal{E}). \quad (21)$$

Similarly, constraints (21) are valid only when $\mathcal{Q}(\mathcal{E})$ is a mosaic-induced edge clique cover, as $l(q)$ is undefined if clique q does not originate from a fragment.

We report in Table 5 the performance measures of **MT_1×1** for the six tested structures for modeling the objective function.

Table 5: Performance of **MT_1×1** for the six tested structures for modeling the objective function.

Approach	Performance						Model size			Presolved model size		
	# opt.	T _t	T _p	LB	UB	# feas.	# var.	# cons.	# nz.	# var.	# cons.	# nz.
MT_1×1_OBJ1 (p/i/1)	45	1663	3	20.15	22.25	80	3747	5445	647 215	3830	4894	622 507
MT_1×1_OBJ2 (p/i/1)	49	1560	2.9	20.38	22.19	81	3747	4035	341 736	3729	3299	341 678
MT_1×1_OBJ3 (p/i/1)	49	1548	2.9	19.91	22.25	81	3747	4035	208 363	3737	3227	213 405
MT_1×1_OBJ4 (p/i/1)	48	1628	3.1	20.68	23.25	48	2937	2761	157 249	2892	2113	159 555
MT_1×1_OBJ5 (p/i/1)	49	1535	3	20.65	22.44	76	3770	4058	208 410	3756	3263	213 500
MT_1×1_OBJ6 (p/i/1)	49	1519	3	20.79	22.41	75	3770	4058	211 713	3744	3151	213 762
MT_1×1_OBJ6 (c/d/1)	48	1585	3.1	20.70	22.44	75	3770	4058	223 315	3742	3170	223 714

First of all, it is clear that **MT_1×1_OBJ1** underperforms, indicating that considering packing candidate families when computing a minimum vertex cover with our mosaic approach is important. As for the non-zero-element reduction, whereas it was an important element when using an edge clique cover to derive the non-overlap constraints, it seems less important when using a minimum vertex cover to derive objective-function-related constraints (20). Destructive bounds seem useful if one aims for a high-quality lower bound and smaller model sizes (although multiple ILP models must be solved, one each time the lower bound is increased by one unit), though they appear less effective when comparing the number of optimal solutions found and average computation time. As expected, they clearly underperform when focusing on the number of feasible solutions found, since, by design, any feasible solution is optimal when destructive bounds are used. Finally, the newly introduced structure using decision variables z_l in the objective function seems to be quite effective,

with **MT_1×1_OBJ5** combining the smaller average model size of **MT_1×1_OBJ3** and the stronger average lower bounds of **MT_1×1_OBJ4**. Meanwhile, **MT_1×1_OBJ6**, which also incorporates the z_l variables into the non-overlap constraints, performs slightly better in terms of average lower bound and computation time. Finally, the variant of **MT_1×1_OBJ6** using configuration c/d/1 offers no competitive advantage over configuration p/i/1.

6.5 Performance evaluation

At this stage, experiments have led us to favor **MT_1×1_OBJ6** with configuration p/i/1, although we observed that other objective functions and configurations resulted in relatively similar performance. In this final set of experiments, we further investigate the performance of our best approach. First, we measure the impact of randomness by running the approach with 10 different random seeds (dubbed **MT_1×1#X**, where X is the random seed). According to the solver’s documentation, changing the seed “acts as a small perturbation to the solver, and typically leads to different solution paths” (Gurobi optimization, 2026). Second, we evaluate the performance of our best approach under increased computational resources: specifically, (i) allowing Gurobi to use up to 8 threads (setting the **Threads** parameter to 8), (ii) extending the optimization time limit to 5 hours, and (iii) combining both adjustments (i.e., 8 threads and a 5-hour time limit). We designate these setups as **MT_1×1_8T**, **MT_1×1_5H**, and **MT_1×1_8T5H**, respectively. These setups are inspired by previous studies in the literature: Rodrigues and Toledo (2017) reported experiments using a time limit of 18 000s, while the parallel branch-and-bound-and-check algorithm of Lastra-Díaz and Ortuño (2026) was executed on a computer allowing up to 8 parallel threads.

We report in Table 6 the performance measures of **MT_1×1** across the 13 tested setups. Note that because we report the average total CPU time rather than total wall-clock time, values in column “ T_t ” may exceed the nominal time limit for approaches using up to 8 threads. Furthermore, thread utilization is dynamically managed by the ILP solver and is not constant throughout the run, which is why a run may terminate before the total CPU time reaches 8 times the time limit.

Table 6: Performance of **MT_1×1** for the 13 tested setups.

Approach	Performance						Model size			Presolved model size		
	# opt.	T_t	T_p	LB	UB	# feas.	# var.	# cons.	# nz.	# var.	# cons.	# nz.
MT_1×1#1	48	1584	3	20.74	22.46	73	3770	4058	211 713	3741	3148	214 055
MT_1×1#2	48	1553	3	20.74	22.46	75	3770	4058	211 713	3742	3150	213 799
MT_1×1#3	49	1513	3	20.72	22.46	73	3770	4058	211 713	3741	3146	214 361
MT_1×1#4	49	1560	2.9	20.69	22.32	76	3770	4058	211 713	3741	3144	214 638
MT_1×1#5	50	1543	2.9	20.70	22.47	74	3770	4058	211 713	3742	3150	213 799
MT_1×1#6	50	1506	3.1	20.69	22.48	73	3770	4058	211 713	3741	3148	214 055
MT_1×1#7	48	1624	2.9	20.74	22.44	75	3770	4058	211 713	3744	3149	214 018
MT_1×1#8	48	1607	2.9	20.72	22.49	73	3770	4058	211 713	3744	3149	214 018
MT_1×1#9	47	1629	2.9	20.75	22.38	74	3770	4058	211 713	3744	3149	214 018
MT_1×1#10	47	1611	3	20.72	22.48	75	3770	4058	211 713	3741	3146	214 361
MT_1×1_8T	55	7143	3.1	20.90	22.06	81	3770	4058	211 713	3744	3151	213 762
MT_1×1_5H	54	6840	3	20.91	22.23	79	3770	4058	211 713	3744	3151	213 762
MT_1×1_8T5H	61	34 990	3.1	21.09	21.96	81	3770	4058	211 713	3744	3151	213 762

First of all, it is interesting to notice that some metrics are noticeably more sensitive to random seed selection than others. For example, the model size after presolve and the average lower and upper bounds remain relatively stable across all ten seeds. On the other hand, the average total computation time can deviate by up to 120 seconds between the best and worst outcomes, and the

number of optimal solutions found varies by up to 3. Reassuringly, the results reported in Table 5 for **MT_1×1_OBJ6**, while slightly better than the average across seeds, are representative and not anomalous. Exactly 44 out of the 81 instances were solved by all ten seeds, and 52 instances were solved by at least one seed. Interestingly, for the 8 instances solved by only a subset of seeds, the discrepancy almost always stemmed either from the lower bound (i.e., some seeds closed the gap while others failed) or from the upper bound (i.e., some seeds were “lucky” when calling heuristics and could find an optimal solution, whereas others were not).

Regarding the impact of increased computational resources, it is clear that allowing Gurobi to use up to eight threads instead of one improves the performance of our best **MT_1×1** variant, allowing it to solve up to 55 instances to optimality. Increasing the time limit to 5 hours is also beneficial, allowing our approach to solve 54 benchmark instances. As expected, combining these two resource increases leads to further improvements, with 61 out of the 81 tested instances solved to optimality.

Finally, we point out that our eight-thread approach appears competitive with that of Lastra-Díaz and Ortuño (2026), who report solving 56 instances to optimality, only one more than **MT_1×1.8T**. A direct comparison is difficult, however, because the experiments were conducted on different hardware running different operating systems, and the underlying code was implemented in different programming languages. Furthermore, the two approaches excel on different benchmark instances: the parallel branch-and-bound-and-check algorithm of Lastra-Díaz and Ortuño (2026) solved five *J2* and three *poly* instances (characterized by their low overall item counts) that our eight-thread approach could not, whereas the reverse was true for two *BLAZEWCZ*, two *RCO*, one *artif*, and two *shirts* instances (characterized by high item demands).

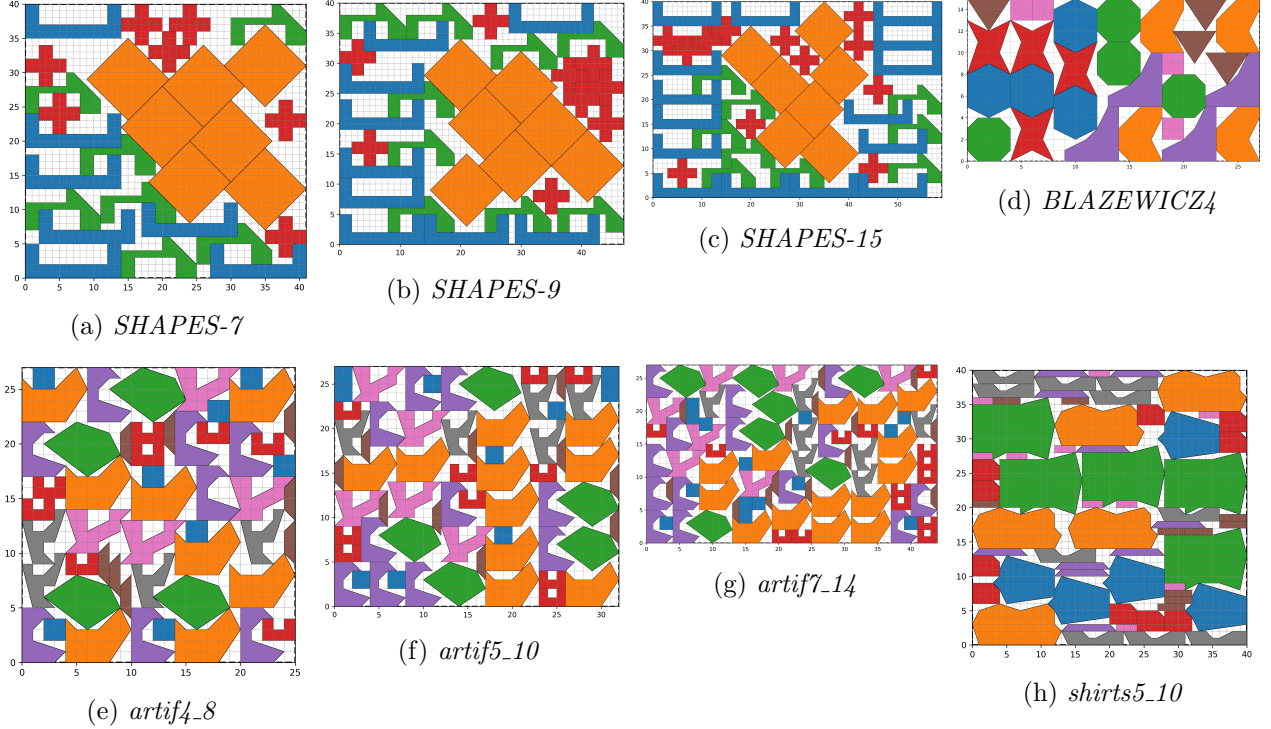
To assist the C&P community, we provide further results at the instance level in Table 7. Specifically, we report the best instance-specific lower and upper bounds reached by the three tested approaches of Lastra-Díaz and Ortuño (2026) (two dotted-board-based models and their parallel branch-and-bound-and-check algorithm, dubbed “LO26”), as well as their counterparts for the 43 (11 + 12 + 7 + 13) implementations tested in this section. For both sets of bounds, a gray highlight indicates an improvement over the other. In addition, we report the detailed outcomes for **MT_1×1.8T** and **MT_1×1.8T5H**, with the latter accounting for most of the bound improvements shown in the table. For clarity, we only report the model size before presolve, and we do so only once, as it is identical for both approaches. A more complete version of this table, which includes the metrics for all 81 benchmark instances, is available in Section B of the online supplement.

Table 7: Instance-specific performance metrics for **MT_1×1.8T** and **MT_1×1.8T5H**.

Instance	LO26		This paper		MT_1×1.8T					MT_1×1.8T5H					MT_1×1 model size		
	LB _p	UB _p	LB _h	UB _h	# opt.	T _t	T _p	LB	UB	# opt.	T _t	T _p	LB	UB	# var.	# cons.	# nz.
SHAPES-4	23	25	25	25	0	26 497.1	0.1	24	26	1	44 679.2	0.1	25	25	2496	4511	118 300
SHAPES-7	35	42	36	41	0	20 494.1	0.1	36	41	0	129 829	0.1	36	41	5216	7731	251 380
SHAPES-9	38	52	41	47	0	14 690.5	0.1	40	48	0	117 566	0.1	41	48	6032	8697	291 304
SHAPES-15	47	64	50	59	0	12 212.6	0.1	49	64	0	91 589.7	0.1	50	64	7800	10 790	377 806
BLAZEWCZ4	26	28	27	27	0	27 235.4	0	26	27	1	123 058	0	27	27	2204	2210	49 008
BLAZEWCZ5	31	34	32	34	0	26 654.3	0	32	34	0	136 137	0	32	34	2792	2742	62 126
RCO4	28	29	29	29	1	5055.3	0	29	29	1	5071.7	0	29	29	2204	2210	45 268
artif2-4	12	13	13	13	1	7267.4	0.1	13	13	1	7279.8	0.1	13	13	2113	2667	50 931
artif3-6	17	19	18	19	0	24 361.1	0.1	18	19	0	136 442	0.1	18	19	3271	3807	80 799
artif4-8	22	26	23	25	0	20 389.5	0.1	23	26	0	125 217	0.1	23	26	4815	5327	120 623
artif5-10	27	33	28	32	0	14 197.9	0.1	28	32	0	105 930	0.1	28	32	5973	6467	150 491
artif7-14	38	46	39	44	0	16 184.4	0.1	38	46	0	100 936	0.1	39	45	8675	9127	220 183
artif	41	49	42	49	0	13 460.6	0.1	38	50	0	92 259.3	0.1	42	50	9640	10 077	245 073
shirts4-8	31	32	32	32	1	2180.8	0.4	32	32	1	2176.9	0.4	32	32	8515	7042	318 877
shirts5-10	38	41	39	40	0	15 195.8	0.4	39	41	0	118 301	0.4	39	40	10 608	8449	404 452
fu	31	34	34	34	0	16 898.7	21.3	33	34	1	23 007.3	22.8	34	34	8680	10 156	1 003 785
j2-12-35-0	29	29	28	29	0	20 349.3	8.5	27	30	0	125 852	7.5	28	29	6271	7636	589 599
j2-12-35-1	25	25	25	25	0	19 369.5	4.8	24	26	0	117 659	4.8	24	25	6129	7144	481 793
j2-14-35-0	30	30	28	30	0	13 966.6	12.9	28	32	0	109 927	12.1	28	30	8752	8376	729 006
j2-14-35-1	26	29	27	29	0	16 004.8	10.5	26	29	0	120 017	9.8	27	29	8653	8130	704 317
j2-14-35-3	24	26	24	27	0	18 629.7	5.9	24	27	0	124 034	5.9	24	27	7581	7146	560 321
poly1a	15	15	15	16	0	22 384.2	21	15	16	0	131 539	21.4	15	16	5638	7232	361 322
poly1b	18	18	17	18	0	17 363.1	37.3	17	19	0	126 055	39.2	17	19	6879	6755	464 684

As a result of our experiments, six new instances were solved to optimality compared to the results of Lastra-Díaz and Ortuño (2026), namely *SHAPES-4*, *BLAZEWCZ4*, *RCO4*, *artif2-4*, *shirts4-8*, and *fu*. Three of these instances were closed by **MT₁×1.8T**, whereas all six were closed by **MT₁×1.8T5H**. In addition, 11 lower bound improvements were obtained (6 by **MT₁×1.8T** and all 11 by **MT₁×1.8T5H**), along with 7 upper bound improvements (3 shared by both **MT₁×1.8T** and **MT₁×1.8T5H**, and the remaining 4 by other approaches deemed subpar in previous experiments). We report the item layouts corresponding to these 7 upper bound improvements as well as *BLAZEWCZ4* (which was also solved partly thanks to an improved upper bound) in Figure 17.

Figure 17: Visual representation of the 8 improved solutions.



7 Conclusions

In this work, we developed alternative ways to solve the dotted-board model for the irregular strip packing problem compared to those presented by Toledo et al. (2013) and Rodrigues and Toledo (2017). Specifically, our mosaic approach decomposes the strip into fragments and uses a subset of these fragments as clique constraints, whereas Rodrigues and Toledo (2017) solved a (costly) minimum edge clique cover to derive these clique constraints and Toledo et al. (2013) used a large number of pairwise incompatibility constraints instead. Although we showed that constructing the mosaic for the whole strip was impractical, we demonstrated that constructing it for a single 1×1 master tile and repeating such a tile throughout the entire strip could be done reasonably fast. We also discussed various improvements aimed at making the overall approach competitive, specifically by reducing the number of fragments considered, reducing the number of non-zero elements in the resulting model, and devising alternative objective functions. Our best approach outperforms a re-implementation of the model of Rodrigues and Toledo (2017) (a re-implementation inspired by the protocol described by Lastra-Díaz and Ortuño (2026)), but only by a thin margin.

From an optimization perspective, our conclusion is certainly not that our best approach should immediately replace the model described by Rodrigues and Toledo (2017): in fact, as can be seen in our online repository, the latter is incredibly simple to re-implement if the `Clipper2` library (Johnson, 2025) is used to detect pairwise incompatibilities and the `ECC8` software library (Conte et al., 2020) is used to find a minimum edge clique cover. Conversely, ours requires additional intermediate functions, notably to build the mosaic and reduce its size. We also do not claim that the current mosaic approach is as good as it can get. First, although the improvements brought by considering larger master tiles seemed minor, they might deserve further investigation. Second, the preprocessing time, while already divided by five compared to our re-implementation of the model by Rodrigues and Toledo (2017), can still reach several dozen seconds for large instances. There are avenues to reduce this time further, for example, by solving the set covering problem by means of heuristics. What our results have shown, however, is that there is much more to the dotted-board model than what has been established in the literature: one does not have to use a time-consuming preprocessing to make it work; using clique constraints to forbid item overlap is not that different from the type of non-overlapping constraints proposed by Beasley (1985) for the regular strip packing problem; the objective function is not set in stone, as several alternatives exist (some of which we have described, with others likely remaining to be discovered); and more importantly, while we empirically showed that it is good to reduce the number of constraints in the model, we also observed that there is a fine balance to be found regarding the number of non-zero elements (variants with too many or too few non-zero elements underperformed in our experiments). In practice, this means that even when using the approach described by Rodrigues and Toledo (2017), constructing the cliques is not entirely straightforward, as it could be beneficial to increase the size of some cliques by covering incompatibilities that are already covered, but only to a limited extent. Finally, our experiments suggest that there is currently no single “go-to” approach that should be used universally when solving an irregular strip packing instance. Instead, there appears to be a constellation of complementary approaches (including ours, as well as that proposed by Lastra-Díaz and Ortuño (2026)), with certain methods performing better depending on instance characteristics.

From the problem perspective, we found that the benchmark instances currently used in the literature are not particularly diverse: some instances are very easy and can be solved by all approaches, whereas others are very difficult and cannot be solved by any. In fact, out of the 81 tested instances, major performance differences among approaches were observed on roughly 20. Instance space analysis, as implemented in MATILDA (Smith-Miles and Muñoz, 2023), could be particularly useful in this context to identify under-represented regions of the instance space and guide the generation of more diverse benchmark instances. We therefore encourage further research on problem instance generation, as has been done for other cutting and packing problems. The main difficulty for irregular strip packing lies in balancing the realism of the generated instances with the diversity required for meaningful algorithmic assessment.

Acknowledgments

This work used the Dutch national e-infrastructure with the support of the SURF Cooperative using grant no. EINF-17056. We also thank Joris Hendriks for his valuable contributions during the initial phase of our research on this problem.

References

- F.B. Akçay and M. Delorme. Solving the parallel processor scheduling and bin packing problems with contiguity constraints: mathematical models and computational studies. *European Journal of Operational Research*, 323:701–723, 2025.
- F.B. Akçay and M. Delorme. Solving the strip packing problem with a decomposition framework and a generic solver: Implementation, tuning, and reinforcement-learning-based hybridization. *Computers & Operations Research*, 185:107276, 2026.
- C. Alves, P. Brás, J.M. Valério de Carvalho, and T. Pinto. New constructive algorithms for leather nesting in the automotive industry. *Computers & Operations Research*, 39:1487–1505, 2012.
- R.C. Jr Art. An approach to the two dimensional irregular cutting stock problem. Technical report, Massachusetts Institute of Technology, 1966. URL <http://hdl.handle.net/1721.1/97782>.
- R. Baldacci, M.A. Boschetti, M. Ganovelli, and V. Maniezzo. Algorithms for nesting with defects. *Discrete Applied Mathematics*, 163:17–33, 2014.
- J. Balogh, J. Békési, G. Dósa, L. Epstein, and A. Levin. A new lower bound for classic online bin packing. *Algorithmica*, 83:2047–2062, 2021.
- J.E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, pages 49–64, 1985.
- J.A. Bennell and J.F. Oliveira. The geometry of nesting problems: A tutorial. *European journal of operational research*, 184:397–415, 2008.
- J.A. Bennell and J.F. Oliveira. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:S93–S105, 2009.
- P.M. Castro and I.E. Grossmann. From time representation in scheduling to the solution of strip packing problems. *Computers & Chemical Engineering*, 44:45–57, 2012.
- P.M. Castro and J.F. Oliveira. Scheduling inspired models for two-dimensional packing problems. *European Journal of Operational Research*, 215:45–56, 2011.
- L.H. Cherri, L.R. Mundim, M. Andretta, F.M.B. Toledo, J.F. Oliveira, and M.A. Carravilla. Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, 253:570–583, 2016.
- L.H. Cherri, A.C. Cherri, M.A. Carravilla, J.F. Oliveira, F.M.B. Toledo, and A.C.G. Vianna. An innovative data structure to handle the geometry of nesting problems. *International Journal of Production Research*, 56:7085–7102, 2018.
- L.H. Cherri, A.C. Cherri, E.F. Silva, and J.F. Oliveira. A branch-and-cut algorithm for nesting problems with guillotine constraints. *European Journal of Operational Research*, 2026.
- N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25:30–44, 1977.
- N.M. Cid-Garcia and Y.A. Rios-Solis. Exact solutions for the 2D-strip packing problem using the positions-and-covering methodology. *Plos one*, 16:e0245267, 2021.
- F. Clautiaux and I. Ljubić. Last fifty years of integer linear programming: A focus on recent practical advances. *European Journal of Operational Research*, 324:707–731, 2025.

- A. Conte, R. Grossi, and A. Marino. Large-scale clique cover of real-world networks. *Information and Computation*, 270:104464, 2020.
- J.-F. Côté and M. Iori. The meet-in-the-middle principle for cutting and packing problems. *INFORMS Journal on Computing*, 30:646–661, 2018.
- J.-F. Côté, M. Dell’Amico, and M. Iori. Combinatorial Benders’ cuts for the strip packing problem. *Operations Research*, 62:643–661, 2014.
- R. F. da Silva and R. Schouery. Solving cutting stock problems via an extended ryan-foster branching scheme and fast column generation. *INFORMS Journal on Computing*, 2025.
- M. Delorme and J. Malinowski. Modeling bloons tower defense as a temporal two-dimensional knapsack problem with irregular shapes and side constraints: integer programming-based approaches. *International Transactions in Operational Research*, 2026.
- M. Delorme and J. Martinovic. Extending the reflect flow formulation to variable-sized one-dimensional cutting and skiving stock problems. *European Journal of Operational Research*, 334:51–69, 2026.
- M. Delorme, M. Iori, and S. Martello. Logic based benders’ decomposition for orthogonal stock cutting problems. *Computers & Operations Research*, 78:290–298, 2017.
- K.A. Dowsland and W.B. Dowsland. Solution approaches to irregular nesting problems. *European Journal of Operational Research*, 84:506–521, 1995.
- J. Gardeyn, G.V. Berghe, and T. Wauters. An open-source heuristic to reboot 2d nesting research. Technical report, 2026. URL <https://arxiv.org/abs/2509.13329>.
- N. Giovenali, G. Bruno, and P. Chiabert. Comparative evaluation of irregular shape strip-packing algorithms. In *International Conference on Innovative Intelligent Industrial Production and Logistics*, pages 115–127, 2024.
- B. Guo, Y. Zhang, J. Hu, J. Li, F. Wu, Q. Peng, and Q. Zhang. Two-dimensional irregular packing problems: A review. *Frontiers in Mechanical Engineering*, 8:966691, 2022.
- B. Guo, Y. Wang, Z. Wang, S. Huang, Z. Xu, C. Huang, and Q. Peng. Irregular multi-plate packing based on the dotted board model. *Computers & Industrial Engineering*, 209:111439, 2025. doi: 10.1016/j.cie.2025.111439.
- Gurobi optimization. Gurobi online documentation. <https://www.gurobi.com/documentation/current/refman/seed.html>, 2026. accessed 24 September 2026.
- W. Han, J.A. Bennell, X. Zhao, and X. Song. Construction heuristics for two-dimensional irregular shape bin packing with guillotine constraints. *European journal of operational research*, 230:495–504, 2013.
- J. Heistermann and T. Lengauer. The nesting problem in the leather manufacturing industry. *Annals of Operations Research*, 57:147–173, 1995.
- J.C. Herz. Recursive computational procedure for two-dimensional stock cutting. *IBM Journal of Research and Development*, 16:462–469, 1972.
- E. Hopper and B.C.H. Turton. A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16:257–300, 2001.
- A. Johnson. Clipper2 – a polygon clipping, offsetting and triangulation library (in C++, C# & Delphi). <https://github.com/AngusJohnson/Clipper2>, 2025. Version 2.0.1.
- R.M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.

- L.T. Kou, L.J. Stockmeyer, and C.-K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, 21:135–139, 1978.
- C. Lamas-Fernandez, J.A. Bennell, and A. Martinez-Sykora. Voxel-based solution approaches to the three-dimensional irregular packing problem. *Operations research*, 71:1298–1317, 2023.
- J. J. Lastra-Díaz and M. T. Ortuño. Mixed-integer programming models for irregular strip packing based on vertical slices and feasibility cuts. *European Journal of Operational Research*, 313(1):69–91, 2024. doi: 10.1016/j.ejor.2023.08.009.
- J.J. Lastra-Díaz and M.T. Ortuño. A parallel branch-and-bound-and-check algorithm for nesting. *Computers & Operations Research*, 194:107535, 2026.
- A.A.S. Leao, F.M.B. Toledo, J.F. Oliveira, and M.A. Carravilla. A semi-continuous MIP model for the irregular strip packing problem. *International Journal of Production Research*, 54:712–721, 2016.
- A.A.S. Leao, F.M.B. Toledo, J.F. Oliveira, M.A. Carravilla, and R. Alvarez-Valdés. Irregular packing problems: A review of mathematical models. *European journal of operational research*, 282:803–822, 2020.
- C. Paquay, M. Schyns, and S. Limbourg. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, 23:187–213, 2016.
- M.O. Rodrigues. Fast constructive and improvement heuristics for edge clique covering. *Discrete Optimization*, 39:100628, 2021.
- M.O. Rodrigues and F.M.B. Toledo. A clique covering MIP model for the irregular strip packing problem. *Computers & Operations Research*, 87:221–234, 2017.
- G. Scheithauer and J. Terno. The g4-heuristic for the pallet loading problem. *Journal of the Operational Research Society*, 47:511–522, 1996.
- S.A. Segenreich and L.M.P.F. Braga. Optimal nesting of general plane figures: A monte carlo heuristical approach. *Computers & Graphics*, 10:229–237, 1986.
- K. Smith-Miles and M. A. Muñoz. Instance space analysis for algorithm testing: Methodology and software tools. *ACM Computing Surveys*, 55(12):1–31, 2023. doi: 10.1145/3572895.
- F.M.B. Toledo, M.A. Carravilla, C. Ribeiro, J.F. Oliveira, and A.M. Gomes. The dotted-board model: A new MIP model for nesting irregular shapes. *International Journal of Production Economics*, 145:478–487, 2013.

APPENDIX

A Benchmark instance description

A summary of the main instance characteristics, namely the total number of items (column “ $\sum d_j$ ”), the number of item types (“ $|\mathcal{J}|$ ”), the strip width (“ W ”) and length (“ L ”), the number of packing candidates (“ $|\mathcal{P}|$ ”), and the number of pairwise incompatibilities (“ $|\mathcal{I}|$ ”), is presented in Table A.1. (Note that these values slightly differ from those reported by Rodrigues and Toledo (2017), but they do exactly match those reported by Lastra-Díaz and Ortuño (2026).)

Table A.1: Main characteristics of the 81 benchmark instances.

Instance	Items		Strip		Incompatibilities		Instance	Items		Strip		Incompatibilities	
	$\sum d_j$	$ \mathcal{J} $	W	L	$ \mathcal{P} $	$ \mathcal{I} $		$\sum d_j$	$ \mathcal{J} $	W	L	$ \mathcal{P} $	$ \mathcal{I} $
three	3	3	7	7	61	1266	fu8	8	7	38	24	2772	1 616 281
threep2	6	3	7	11	117	3006	fu9	9	8	38	29	4202	3 446 231
threep2w9	6	3	9	9	127	3394	fu10	10	9	38	34	5982	6 365 641
threep3	9	3	7	16	187	5181	fu	12	11	38	38	8642	11 355 356
threep3w9	9	3	9	13	207	6150	J1-10-10-0	10	9	10	20	1063	118 778
shapes4	4	4	13	24	389	46 060	J1-10-10-1	10	10	10	19	1065	154 905
shapes8	8	4	20	28	1008	201 702	J1-10-10-2	10	9	10	21	1073	140 430
SHAPES2	8	4	40	16	848	92 766	J1-10-10-3	10	10	10	23	1284	232 436
SHAPES4	16	4	40	28	2468	604 259	J1-10-10-4	10	10	10	15	896	96 289
SHAPES5	20	4	40	35	3413	939 916	J1-12-20-0	12	11	20	12	1597	260 436
SHAPES7	28	4	40	48	5168	1 563 279	J1-12-20-1	12	11	20	11	1377	207 624
SHAPES9	34	4	40	54	5978	1 850 985	J1-12-20-2	12	12	20	14	2039	445 512
SHAPES15	43	4	40	67	7733	2 474 348	J1-12-20-3	12	12	20	10	1363	172 873
blasz2	20	4	15	27	1080	91 670	J1-12-20-4	12	12	20	16	2408	580 191
BLAZEWCZ1	7	7	15	8	432	35 462	J1-14-20-0	14	13	20	14	2344	469 901
BLAZEWCZ2	14	7	15	16	1096	120 814	J1-14-20-1	14	13	20	14	2319	459 468
BLAZEWCZ3	21	7	15	22	1594	184 828	J1-14-20-2	14	14	20	16	2781	776 783
BLAZEWCZ4	28	7	15	29	2175	259 511	J1-14-20-3	14	13	20	12	1979	348 312
BLAZEWCZ5	35	7	15	36	2756	334 194	J1-14-20-4	14	13	20	16	2631	634 924
RCO1	7	7	15	8	432	36 049	J2-10-35-0	10	9	35	28	4486	3 505 202
RCO2	14	7	15	17	1179	133 924	J2-10-35-1	10	9	35	28	4572	3 566 129
RCO3	21	7	15	25	1843	220 924	J2-10-35-2	10	10	35	27	4892	3 811 214
RCO4	28	7	15	29	2175	264 424	J2-10-35-3	10	9	35	25	4328	2 872 285
RCO5	35	7	15	41	3171	394 924	J2-10-35-4	10	10	35	22	3992	2 678 511
artif1_2	13	8	27	8	947	68 540	J2-12-35-0	12	11	35	31	6240	6 493 492
artif2_4	26	8	27	14	2099	250 220	J2-12-35-1	12	11	35	29	6100	5 234 224
artif3_6	39	8	27	20	3251	437 228	J2-12-35-2	12	10	35	30	5852	4 829 349
artif4_8	52	8	27	28	4787	686 572	J2-12-35-3	12	11	35	25	5420	4 143 829
artif5_10	65	8	27	34	5939	873 580	J2-12-35-4	12	11	35	29	6256	5 439 275
artif6_12	78	8	27	41	7283	1 091 756	J2-14-35-0	14	13	35	34	8718	10 155 708
artif7_14	91	8	27	48	8627	1 309 932	J2-14-35-1	14	13	35	33	8620	9 308 609
artif	99	8	27	53	9587	1 465 772	J2-14-35-2	14	11	35	33	7484	6 813 466
shirts1_2	13	8	40	13	1924	147 101	J2-14-35-3	14	13	35	29	7552	7 557 702
shirts2_4	26	8	40	20	4010	736 227	J2-14-35-4	14	13	35	31	8056	8 157 073
shirts3_6	39	8	40	26	5798	1 364 984	poly1a	15	15	40	18	5620	3 283 976
shirts4_8	52	8	40	35	8480	2 319 578	poly1b	15	15	40	21	6858	4 922 042
shirts5_10	65	8	40	42	10 566	3 062 040	poly1c	15	15	40	14	4437	1 513 472
dagli1	10	10	60	25	6911	6 170 439	poly1d	15	15	40	14	4527	1 946 264
fu5	5	4	38	18	721	120 663	poly1e	15	15	40	13	3638	1 301 254
fu6	6	5	38	24	1722	734 345	jakobs1	25	22	40	13	7460	2 846 369
fu7	7	6	38	24	2172	1 124 614							

B Supplementary tables

Table B.1: Instance-specific performance metrics for the evaluated dotted-board-based approaches.

Instance	CLIQUE $\times 10$				MOSAIC			MT-1 $\times 1$ (MT $_{SC-F}^O$)				MT-2 $\times 2$ (MT $_{SC-F}^O$)				MT-4 $\times 4$ (MT $_{SC-F}^O$)				MT-1 $\times 1$ (MT $_{SC}^O$)				MT-2 $\times 2$ (MT $_{SC}^O$)				MT-4 $\times 4$ (MT $_{SC}^O$)				
	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME	T $_t$	T $_p$	NOC	ME
three	0.2	0.2	28	0.4	0.1	0.1	22	0.4	0	0	98	2	0	0	98	2	0.1	0.1	98	2	0	0	196	4	0	0	154	3	0.1	0.1	122	2.4
threep2	0.3	0.3	71	0.9	0.2	0.2	69	0.9	0	0	231	3	0	0	192	2.5	0.1	0.1	195	2.5	0	0	462	6	0	0	319	4	0.1	0.1	247	3.2
threep2w9	0.4	0.3	82	0.9	0.3	0.3	70	0.9	0	0	243	3	0	0	202	2.5	0.1	0.1	205	2.5	0	0	486	6	0	0	333	4	0.1	0.1	273	3.2
threep3	0.5	0.5	120	1.1	0.4	0.4	124	1.1	0.1	0	336	3	0.1	0	280	2.5	0.1	0.1	284	2.5	0.1	0	672	6	0.1	0	464	4	0.1	0.1	360	3.2
threep3w9	0.6	0.5	147	1.1	0.7	0.6	134	1.1	0.1	0	351	3	0.1	0	292	2.5	0.2	0.1	296	2.5	0.1	0	702	6	0.1	0	481	4	0.2	0.1	394	3.2
shapes4	3.5	3.4	296	1.3	1.3	201	0.6	0.1	0	624	2	0.3	0.2	624	2	2	1.9	624	2	0.1	0	936	3	0.2	0.1	708	2.3	0.5	0.4	648	2.1	
shapes8	27.3	13.6	1007	20.7	11.4	552	1	10.4	0.1	1680	3	12.3	0.4	1120	2	9.9	3	1120	2	16.1	0	2240	4	12.7	0.1	1680	3	12	0.5	1400	2.5	
SHAPES2	7.5	6.9	890	6.4	5.6	579	0.9	2.5	0.1	1920	3	1.9	0.4	1280	2	4.9	3	1280	2	0.9	0	2560	4	2	0.1	1920	3	1.5	0.5	1600	2.5	
SHAPES4	T.L.	35	2947	T.L.	47.3	1396	1.2	T.L.	0.1	3360	3	T.L.	0.4	2240	2	T.L.	3.1	2240	2	T.L.	0	4480	4	T.L.	0.1	3360	3	T.L.	0.5	2800	2.5	
SHAPES5	T.L.	53.5	4192	T.L.	79.2	1858	1.3	T.L.	0.1	4200	3	T.L.	0.4	2800	2	T.L.	3.1	2800	2	T.L.	0	5600	4	T.L.	0.1	4200	3	T.L.	0.5	3490	2.5	
SHAPES7	T.L.	85	6431	T.L.	145.1	2716	1.4	T.L.	0.1	5760	3	T.L.	0.4	3840	2	T.L.	3.1	3840	2	T.L.	0	7680	4	T.L.	0.1	5760	3	T.L.	0.5	4800	2.5	
blas2	13.7	6.3	821	74.4	64.8	757	1.9	4.9	0	1616	4	9.6	0.1	1617	4	16.9	2.5	1617	4	15.4	0	6059	15	9.2	0.1	3638	9	6.9	0.9	2488	6.3	
BLAZEWC1	2.8	2.7	258	11	10.9	246	2.1	0.2	0	480	4	0.9	0.7	480	4	10.6	10.4	480	4	0.3	0.1	1800	15	1.1	0.7	1048	8.8	4.7	4.5	692	5.9	
BLAZEWC2	12.6	7.4	735	71.3	60.6	663	2.8	7	0	960	4	7.9	0.9	960	4	17.8	11.5	960	4	30.7	0.1	3840	16	13.9	0.8	2336	9.8	16.5	5.2	1560	6.7	
BLAZEWC3	67.8	10.8	1120	660.4	108.6	969	2.9	701	0	1320	4	240.1	0.9	1320	4	537.9	11.6	1320	4	808.5	0.1	5280	16	560.1	0.8	3212	9.8	1227.7	5.2	2139	6.7	
BLAZEWCZ4	T.L.	14.8	1542	T.L.	176.5	1325	3	T.L.	0	1740	4	T.L.	0.9	1740	4	T.L.	11.6	1740	4	T.L.	0.1	6960	16	T.L.	0.8	4230	9.8	T.L.	5.2	2828	6.7	
RCO1	2.6	2.5	176	5.8	5.7	197	1.6	0.2	0	480	4	0.4	0.3	420	3.5	4.9	4.8	420	3.5	0.2	0	960	8	0.6	0.3	628	5.3	2.5	2.2	502	4.3	
RCO2	10	8.4	580	51	47.3	568	2.2	1.7	0	1020	4	3.2	0.3	900	3.5	10	5	892	3.5	3.6	0	2295	9	3.5	0.4	1467	5.8	6	2.6	1130	4.5	
RCO3	205.9	12.3	926	275.7	94.9	900	2.4	132.6	0	1500	4	142.8	0.3	1320	3.5	285.1	5.2	1312	3.5	131.9	0	3375	9	110.4	0.4	2159	5.8	147.6	2.6	1673	4.5	
RCO4	T.L.	13.7	1112	T.L.	120.4	1066	2.5	T.L.	0	1740	4	T.L.	0.4	1530	3.5	T.L.	5.1	1522	3.5	T.L.	0	3915	9	T.L.	0.4	2505	5.8	T.L.	2.6	1940	4.5	
RCO5	T.L.	18.3	1644	T.L.	204.2	1564	2.5	T.L.	0	2460	4	T.L.	0.3	2160	3.5	T.L.	5.1	2152	3.5	T.L.	0	5535	9	T.L.	0.4	3543	5.8	T.L.	2.6	2741	4.5	
artifl_2	5.5	4.5	571	17.3	16.4	622	2.9	2	0.1	1296	6	2.3	1.2	1296	6	8.1	7.4	1296	6	2.4	0.1	3024	14	2.2	1.4	2040	9.5	7.5	6.7	1628	7.6	
artif2.4	T.L.	14	1559	T.L.	161.6	1489	3.9	T.L.	0.1	2268	6	T.L.	1.3	2268	6	T.L.	8	2268	6	T.L.	0.1	5292	14	T.L.	1.5	3668	9.8	T.L.	7.2	2939	7.8	
fu5	6.3	6.1	216	17.6	17.3	283	0.4	0.7	0.2	1368	2	1.8	1.4	1368	2	13.7	13.3	1368	2	0.6	0.1	1368	2	1.7	1.2	1368	2	9.8	9.3	1368	2	
j1-10-10-0	8.8	7.3	498	5.3	3.6	379	1.9	1.4	0	600	3	1.8	0.2	600	3	3.6	2.1	600	3	1.3	0	800	4	1.8	0.1	700	3.5	1.9	0.6	650	3.3	
j1-10-10-1	12.5	9.6	657	9	5.4	512	2.7	3.1	0	760	4	1.9	0.5	760	4	5.8	4	760	4	3.5	0	950	5	1.9	0.2	850	4.5	5.3	1	800	4.3	
j1-10-10-2	9.2	8.1	482	3.6	2.6	356	1.7	1.1	0	420	2	1.2	0.3	420	2	3.5	2.3	420	2	1.1	0	420	2	0.9	0.1	420	2	1.5	0.4	420	2	
j1-10-10-3	21.3	13.4	604	67.2	58.2	435	1.9	7.5	0.1	690	3	9	1.6	690	3	27	15	690	3	16	0.1	1840	8	13.2	1.8	1200	5.3	18.1	9.2	918	4.1	
j1-10-10-4	8.5	5.8	305	4.2	1.9	254	1.7	2.1	0	450	3	2.9	0.2	450	3	4.3	1.6	450	3	2.1	0	450	3	2.8	0.1	450	3	3.1	0.3	450	3	
j1-12-20-0	17.1	15.7	644	10.1	9.3	486	2	0.8	0.1	720	3	1.3	0.5	720	3	6.4	5.6	720	3	1.3	0	960	4	2.3	0.2	840	3.5	3.9	1.1	780	3.3	
j1-12-20-1	19.3	12.7	716	15.3	7.5	568	2.6	3.6	0.1	880	4	6.1	0.5	880	4	11.9	4.5	880	4	2.9	0	1100	5	6.9	0.2	980	4.5	9.7	1	920	4.3	
j1-12-20-2	36.1	26.7	805	28.3	19.2	591	2.1	8.9	0.1	1120	4	11.1	1.3	1120	4	16.7	11.5	1120	4	8.8	0	1120	4	10.2	0.5	1120	4	8.5	3.2	1120	4	
j1-12-20-3	12	9.4	523	11.3	8.7	446	2.2	2	0	800	4	3.8	0.6	800	4	8.8	5	800	4	3.7	0	1000	5	2.2	0.4	900	4.5	5.1	2	840	4.3	
j1-12-20-4	36.5	30.4	940	85.3	79.7	719	2.2	9.1	0.2	960	3	12.9	1.9	960	3	29.1	16.2	960	3	9.7	0.1	1920	6	9.8	1.4	1440	4.5	24	7.7	1200	3.8	
j1-14-20-0	36.3	24.1	819	130.7	16.3	591	2.1	38.2	0.1	840	3	39.3	0.8	840	3	32.4	8.1	840	3	141.1	0	1120	4	76.3	0.2	980	3.5	14.7	1.4	900	3.3	
j1-14-20-1	64.3	24.7	1027	78.5	18	787	2.8	54.4	0.1	1120	4	51.7	0.8	1120	4	58	8	1120	4	45.5	0	1400	5	47.1	0.2	1260	4.5	41.2	1.4	1180	4.3	
j1-14-20-3	26.7	17.4	530	11.5	5.3	395	1.6	7.2	0.1	720	3	14.5	0.6	720	3	16.3	6.3	720	3	7.1	0	720	3	13.9	0.1	720	3	10.5	0.6	720	3	
j1-14-20-4	54.9	33	1300	178.1	145.6	943	2.9	28	0.2	1280	4	32.3	2.8	1280	4	67.7	24.2	1280	4	28.8	0.1	2560	8	17.5	2.7	1920	6	44.3	14.2	1600	5	

Table B.2: Instance-specific performance metrics for MT_1×1.8T and MT_1×1.8T5H.

Instance	LO26		This paper		MT_1×1.8T				MT_1×1.8T5H				MT_1×1 model size				
	LB _p	UB _p	LB _h	UB _h	# opt.	T _t	T _p	LB	UB	# opt.	T _t	T _p	LB	UB	# var.	# cons.	# nz.
three	6	6	6	6	1	0	0	6	6	1	0	0	6	6	68	156	968
threep2	10	10	10	10	1	0	0	10	10	1	0	0	10	10	128	321	1945
threep2w9	8	8	8	8	1	0	0	8	8	1	0	0	8	8	136	335	2099
threep3	14	14	14	14	1	0.1	0	14	14	1	0.1	0	14	14	203	466	3065
threep3w9	12	12	12	12	1	0.2	0	12	12	1	0.2	0	12	12	220	483	3363
shapes4	24	24	24	24	1	0.1	0	24	24	1	0.1	0	24	24	413	963	15 842
shapes8	26	26	26	26	1	51.6	0.1	26	26	1	51.3	0.1	26	26	1036	2271	46 400
SHAPES-2	14	14	14	14	1	0.6	0.1	14	14	1	0.6	0.1	14	14	864	2579	38 452
SHAPES-4	23	25	25	25	0	26 497.1	0.1	24	26	1	44 679.2	0.1	25	25	2496	4511	118 300
SHAPES-5	28	30	28	30	0	24 553	0.1	28	30	0	136 592	0.1	28	30	3448	5638	164 878
SHAPES-7	35	42	36	41	0	20 494.1	0.1	36	41	0	129 829	0.1	36	41	5216	7731	251 380
SHAPES-9	38	52	41	47	0	14 690.5	0.1	40	48	0	117 566	0.1	41	48	6032	8697	291 304
SHAPES-15	47	64	50	59	0	12 212.6	0.1	49	64	0	91 589.7	0.1	50	64	7800	10 790	377 806
blas2	26	26	26	26	1	29.2	0	26	26	1	29.1	0	26	26	1107	2055	30 421
BLAZEWCICZ1	8	8	8	8	1	0.5	0	8	8	1	0.5	0	8	8	440	614	9923
BLAZEWCICZ2	14	14	14	14	1	17.2	0	14	14	1	17.1	0	14	14	1112	1222	24 646
BLAZEWCICZ3	20	20	20	20	1	364.4	0	20	20	1	364.3	0	20	20	1616	1678	35 890
BLAZEWCICZ4	26	28	27	27	0	27 235.4	0	26	27	1	123 058	0	27	27	2204	2210	49 008
BLAZEWCICZ5	31	34	32	34	0	26 654.3	0	32	34	0	136 137	0	32	34	2792	2742	62 126
RCO1	8	8	8	8	1	0.1	0	8	8	1	0.1	0	8	8	440	614	8720
RCO2	15	15	15	15	1	3.1	0	15	15	1	3.1	0	15	15	1196	1298	24 496
RCO3	22	22	22	22	1	139.9	0	22	22	1	140	0	22	22	1868	1906	38 344
RCO4	28	29	29	29	1	5055.3	0	29	29	1	5071.7	0	29	29	2204	2210	45 268
RCO5	35	36	35	36	0	28 408	0	35	36	0	143 352	0	35	36	3212	3122	66 040
artifi1.2	7	7	7	7	1	0.6	0.1	7	7	1	0.6	0.1	7	7	955	1527	20 759
artifi2.4	12	13	13	13	1	7267.4	0.1	13	13	1	7279.8	0.1	13	13	2113	2667	50 931
artifi3.6	17	19	18	19	0	24 361.1	0.1	18	19	0	136 442	0.1	18	19	3271	3807	80 799
artifi4.8	22	26	23	25	0	20 389.5	0.1	23	26	0	125 217	0.1	23	26	4815	5327	120 623
artifi5.10	27	33	28	32	0	14 197.9	0.1	28	32	0	105 930	0.1	28	32	5973	6467	150 491
artifi6.12	33	38	33	38	0	11 358.5	0.1	33	39	0	111 774	0.1	33	38	7324	7797	185 337
artifi7.14	38	46	39	44	0	16 184.4	0.1	38	46	0	100 936	0.1	39	45	8675	9127	220 183
artifi	41	49	42	49	0	13 460.6	0.1	38	50	0	92 259.3	0.1	42	50	9640	10 077	245 073
shirts1.2	13	13	13	13	1	0.6	0.3	13	13	1	0.6	0.3	13	13	1937	2620	49 735
shirts2.4	17	17	17	17	1	52.4	0.4	17	17	1	52.3	0.4	17	17	4030	4027	135 502
shirts3.6	24	24	24	24	1	2922.1	0.4	24	24	1	2924.9	0.4	24	24	5824	5233	208 852
shirts4.8	31	32	32	32	1	2180.8	0.4	32	32	1	2176.9	0.4	32	32	8515	7042	318 877
shirts5.10	38	41	39	40	0	15 195.8	0.4	39	41	0	118 301	0.4	39	40	10 608	8449	404 452
daghl1	23	23	23	23	1	328.6	32.7	23	23	1	342.6	34	23	23	6936	7534	762 567
fu5	18	18	18	18	1	1.4	0.2	18	18	1	1.4	0.2	18	18	739	2073	72 548
fu6	23	23	23	23	1	12.9	0.8	23	23	1	12.9	0.8	23	23	1746	2764	200 508
fu7	24	24	24	24	1	14.9	1.4	24	24	1	14.9	1.4	24	24	2196	2765	237 858
fu8	24	24	24	24	1	13.5	2	24	24	1	13.6	2	24	24	2796	2766	271 458
fu9	25	25	25	25	1	214.8	3.9	25	25	1	215.5	3.9	25	25	4231	4444	427 842
fu10	29	29	29	29	1	233.9	8.9	29	29	1	233.7	8.2	29	29	6016	5210	638 822
fu	31	34	34	34	0	16 898.7	21.3	33	34	1	23 007.3	22.8	34	34	8680	10 156	1 003 785
j1-10-10-0	18	18	18	18	1	4.7	0	18	18	1	4.7	0	18	18	1083	828	22 362
j1-10-10-1	17	17	17	17	1	7.2	0.1	17	17	1	7.2	0.1	17	17	1084	978	26 193
j1-10-10-2	20	20	20	20	1	1.2	0	20	20	1	1.2	0	20	20	1094	659	23 698
j1-10-10-3	21	21	21	21	1	20.3	0.1	21	21	1	20.2	0.1	21	21	1307	952	28 669
j1-10-10-4	13	13	13	13	1	8.9	0	13	13	1	8.9	0	13	13	911	624	13 773
j1-12-20-0	12	12	12	12	1	6.8	0.1	12	12	1	6.7	0.1	12	12	1609	982	36 968
j1-12-20-1	10	10	10	10	1	17.5	0.1	10	10	1	17.5	0.1	10	10	1388	1121	32 109
j1-12-20-2	12	12	12	12	1	27.7	0.1	12	12	1	27.6	0.1	12	12	2053	1425	51 051
j1-12-20-3	8	8	8	8	1	1.8	0	8	8	1	1.7	0	8	8	1373	1021	23 586
j1-12-20-4	13	13	13	13	1	29.9	0.2	13	13	1	29.8	0.2	13	13	2424	1307	57 943
j1-14-20-0	12	12	12	12	1	56.2	0.1	12	12	1	55.9	0.1	12	12	2358	1146	45 534
j1-14-20-1	12	12	12	12	1	60.1	0.1	12	12	1	59.7	0.1	12	12	2333	1426	52 218
j1-14-20-2	14	14	14	14	1	134	0.3	14	14	1	134.2	0.3	14	14	2797	1309	60 704
j1-14-20-3	10	10	10	10	1	17.8	0.1	10	10	1	17.7	0.1	10	10	1991	984	32 440
j1-14-20-4	14	14	14	14	1	50.1	0.2	14	14	1	50	0.2	14	14	2647	1628	67 906
j2-10-35-0	24	24	24	24	1	476.9	3.5	24	24	1	476.8	3	24	24	4514	6896	432 094
j2-10-35-1	24	24	24	24	1	1605.7	2.6	24	24	1	1609.8	2.3	24	24	4600	6896	396 556
j2-10-35-2	23	23	23	23	1	987.1	3.4	23	23	1	987.3	3.4	23	23	4919	6651	403 895
j2-10-35-3	20	20	20	20	1	265.8	1.7	20	20	1	265.7	1.4	20	20	4353	5283	321 194
j2-10-35-4	19	19	19	19	1	542	2.1	19	19	1	543.2	1.8	19	19	4014	5421	278 616
j2-12-35-0	29	29	28	29	0	20 349.3	8.5	27	30	0	125 852	7.5	28	29	6271	7636	589 599
j2-12-35-1	25	25	25	25	0	19 369.5	4.8	24	26	0	117 659	4.8	24	25	6129	7144	481 793
j2-12-35-2	24	24	24	24	1	16 138.2	3.2	24	24	1	16 182.5	2.6	24	24	5882	7389	480 332
j2-12-35-3	21	21	21	21	1	21 976.9	2.8	21	21	1	22 027.7	2.3	21	21	5445	6160	373 273
j2-12-35-4	26	26	26	26	1	2497.4	5.1	26	26	1	2519.4	4.3	26	26	6285	7144	498 577
j2-14-35-0	30	30	28	30	0	13 966.6	12.9	28	32	0	109 927	12.1	28	30	8752	8376	729 006
j2-14-35-1	26	29	27	29	0	16 004.8	10.5	26	29	0	120 017	9.8	27	29	8653	8130	704 317
j2-14-35-2	26	26	26	26	0	22 138.1	4.6	25	26	1	91 646.5	3.8	26	26	7517	8128	608 357
j2-14-35-3	24	26	24	27	0	18 629.7	5.9	24	27	0	124 034	5.9	24	27	7581	7146	560 321
j2-14-35-4	26	26	26	26	0	17 737.3	8.4	25	26	1	35 060.5	7.5	26	26	8087	7638	629 679
poly1a	15	15	15	16	0	22 384.2	21	15	16	0	131 539	21.4	15	16	5638	7232	361 322
poly1b	18	18	17	18	0	17 363.1	37.3	17	19	0	126 055	39.2	17	19	6879	6755	464 684
poly1c	13	13	13	13	1	12.4	7.9	13	13	1	12.4	7.9	13	13	4451	4508	172 928
poly1d	12	12	12	12	1	7371.5	15.4	12	12	1	7401.8	15.8	12	12	4541	3948	203 209
poly1e	12	12	12	12	0	26 382.6	16	11	12	1	44 078.8	16.2	12	12	3651	4707	178 081
jakobs1	11	11	11	11	1	231.2	1.2	11	11	1	229.9	1	11	11			