UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

_____

AN INNOVATIVE DATA STRUCTURE TO HANDLE THE GEOMETRY OF NESTING PROBLEMS

LUIZ HENRIQUE CHERRI
ADRIANA CRISTINA CHERRI
MARIA ANTÓNIA CARRAVILLA
JOSÉ FERNANDO OLIVEIRA
FRANKLINA TOLEDO
ANDRÉA CARLA GONÇALVES VIANNA

Nº 411

_____

RELATÓRIOS TÉCNICOS

ICMC USP
SÃO CARLOS
45 anos
1971 - 2016

São Carlos – SP
Jun./2016

An innovative data structure to handle the geometry of nesting problems

Luiz Henrique Cherri · Adriana Cristina
Cherri · Maria Antónia Carravilla · José
Fernando Oliveira · Franklina Toledo · Andréa
Carla Gonçalves Vianna

Abstract As in many other combinatorial optimization problems, research on nesting problems (aka irregular packing problems) has evolved around the dichotomy between continuous (time consuming) and discrete (memory consuming) representations of the solution space. Recent research has been devoting increasing attention to discrete representations for the geometric layer of nesting problems, namely in mathematical programming based approaches. These approaches employ conventional regular meshes, and an increase in their precision has a high computational cost. In this paper, we propose a data structure to represent non-regular meshes, based on the geometry of each piece. It supports non-regular discrete geometric representations of the shapes and, by means of the proposed data structure, the discretization can be easily adapted to the instances, thus overcoming the precision loss associated with discrete representations and consequently allowing for a more efficient implementation of search methods for the nesting problem. Experiments are conducted with the dotted-board model - a recently published mesh-based binary programming model for nesting problems. In the light of both the scale of the instances, which are now solvable, and the quality of the solutions obtained, the results are very promising.

L. H. Cherri and F. M. B. Toledo
University of São Paulo, Avenida Trabalhador São-carlense, 400, 13566-590, São Carlos - SP, Brasil
E-mail: {lhcherri,fran}@icmc.usp.br

M. A. Carravilla and J. F. Oliveira
INESC TEC, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal
E-mail: {mac,jfo}@fe.up.pt

A. C. Cherri and A. C. G. Vianna
Universidade Estadual Paulista, Av. Eng. Luiz Edmundo Carrijo Coube, 14-01, 17033-360, Bauru - SP, Brasil
E-mail: {adriana,vianna}@fc.unesp.br

1 Introduction

Nesting problems (aka irregular packing problems) belong to the more general class of Cutting and Packing (C&P) problems and are characterized by the irregular shape of the pieces to cut or pack. Cutting and Packing problems are complex combinatorial optimization problems [13] in which small items (hereinafter called pieces) are assigned to larger objects (hereafter called boards) and laid down (placed) on those objects so that all pieces are contained in the board, no two pieces overlap, and waste (board areas unoccupied by pieces) is minimized. These problems are not only scientifically relevant but also economically and environmentally important, given their many industrial applications and the raw material savings they allow.

Cutting and Packing problems are usually classified according to the number of relevant dimensions. For instance, when planning the cutting of paper rolls, in which only the width of the rolls is relevant for the cutting process, one speaks of one-dimensional C&P problems. When planning how to cut wooden boards into smaller rectangular furniture parts, both having the same thickness, the C&P problems are said to be two-dimensional. Given the geometry of the pieces, they are actually called two-dimensional rectangular C&P problems. When planning how to pack parallelepiped boxes inside a container or truck, in transportation and logistics applications e.g., one deals with three-dimensional C&P problems. For a full characterization of C&P problems see [25].

Nesting problems are characterized by their two-dimensionality and the dealing with irregularly shaped pieces, whose geometry may require more than two parameters to define. Although some experiments have been made with circular arcs, almost all known approaches to this problem model the irregular pieces geometrically as polygons.

Industrial applications of nesting problems are to be found in, among others, the footwear, metalomechanics and garment industries. In the garment industry, long rolls of fabric are laid down in several layers on a table that is many times smaller than the length of the roll, wherefore the roll is, for planning purposes, considered to have infinite length. This problem is known in the literature as the irregular strip packing problem, and the aim is to arrange a set of irregular pieces on a strip so as to minimize the used up strip length.

The geometric problem underlying all C&P problems (the solution feasibility condition that pieces may not overlap) is particularly hard to solve in the nesting problem, given the irregular shapes of the pieces. Some geometric representations have been explored to treat this condition, such as raster points, direct trigonometry, phi-functions and nofit polygons (*NFP*). Raster points divide the board in discrete areas, thus reducing the geometric information data to a matrix, which can lead to a rough approximation. Direct trigonometry consists in representing pieces by their polygon segments, but evaluating if two polygons overlap is complex. In the phi-functions case, the pieces are represented by a set of linear and/or nonlinear functions leading to a nonlinear approach. Finally, the nofit polygon approach evaluates if two pieces overlap by verifying if a point is inside the nofit polygon generated by these two pieces. The nofit polygon combines the precision of direct trigonometry

with a fast overlap evaluation of raster points. Further details about these geometric representations can be found in [4].

Due to its practical relevance, there are many approaches in the literature to solve the nesting problem, but owing to its difficulty, the vast majority of the approaches are based on heuristics or metaheuristics. The developed approaches range from the simpler (although quite effective) one-pass heuristics, such as the bottom-left heuristic [21,9] or the more elaborate Jostle [8] and TOPOS [22] algorithms – a review and categorization of heuristic nesting algorithms can be found in [5] –, to the most complex and up-to-date metaheuristics, as e.g. simulated annealing [15], tabu search [2,19], guided local search [10,24], iterated local search [17] or cuckoo search [11]. Many of these approaches are indeed complex as they combine the metaheuristic search control structure with several lower level heuristics, and are hybridized with the resolution of linear or non-linear programming models. Some approaches based on tree-search methods, as beam-search [3], have also been proposed.

The high complexity of heuristic and metaheuristic algorithms, together with the development of hardware and optimization solvers, led the scientific community to start looking for optimization approaches to nesting problems. The optimal resolution of nesting problems was, for the first time, addressed in 2003 by Carravilla and Ribeiro in [6], with an approach using constraint logic programming. Gomes and Oliveira in 2006 [15], Fischetti and Luzzi in 2009 [12], Alvarez-Valdés et al. in 2013 [1] and Cherri at al. in [7], published mixed-integer programming models for the same problem. In all these exact approaches the coordinates of the placement points of the pieces are the decision variables and are continuous. Toledo et al. [23] published in 2013 the dotted-board model, where decision variables are dots on the board (a discrete model). More recently Leão et al. [20] proposed a semi-continuous model in which, for each piece, the x-axis placement variable is continuous and the y-axis placement variable is discrete.

The dotted-board model has proven to outperform all other models, in that it solves to proven optimality more complex problems (with more pieces), even though at the cost of some accuracy loss caused by the board discretization. Actually, with a discrete model, solutions are optimal for a given grid or discretization step, since a differently refined discretization might lead, although not inevitably, to better solutions. From the computational experiments made with the dotted-board model, two facts became clear: (1) checking if a dot is a feasible placement point is a time-consuming operation; (2) the same discretization level along the entire board leads to unnecessary detail in the case of some board regions or pieces and to lack of accuracy in others.

In this paper a new data structure to represent the board dots is proposed. This data structure is adaptable to the characteristics of each instance and enables the merging of the efficiency of discrete models with the accuracy of continuous variable models. Although the irregular strip packing problem – the most common irregular cutting and packing problem, classified by Wäscher et al. [25] as an Open Dimension Problem – will be dealt with, the hereby proposed new meshes can also be used in the resolution of the irregular variants of the Cutting Stock, Bin Packing, Knapsack, Placement or Identical Item Packing problems. Besides representing the board, the structure carries all the geometric information of the problem. In the new structure,

the dots needn't be equally spaced, which adds flexibility to the mesh definition. To the best of our knowledge, there is no other data structure supporting discrete placement positions with this type of flexibility. The influence of the mesh on the solution quality is verified by applying this structure to the dotted-board model proposed by Toledo et al. [23]. Since the optimal solution quality of the model is strongly dependent on the mesh used, our proposal opens new ways to apply the model in problems with diversified piece shapes.

2 Problem definition

The nesting problem consists in cutting a number of pieces from an either regular or irregular board. These pieces are represented by any shape, convex or concave, and may be of different types T. Each piece of type $t \in T$ can be rotated at a finite number of prefixed angles R_t.

Solving a nesting problem is finding where each piece should be placed on the board, so that the pieces do not overlap and are completely contained in the board. Feasible placement points are represented by their (x,y) coordinates. The objective and some constraints may vary depending on the specific application.

This paper focuses on the irregular strip packing problem, which consists in cutting a number of pieces from a rectangular board of fixed height (H) and infinite length. The objective is to minimize the board length (L) expended in the cutting while meeting the requirements of each piece type $t \in T$.

The data structure proposed in this paper utilizes a discrete geometric representation, where the (x,y) coordinates of the placement points may assume a finite number of values (dots). This structure is generated using the nofit polygon to evaluate if two pieces overlap, and the inner fit polygon to ensure that they lay entirely inside the board. Each piece is represented by a set of vertices and one of these vertices is chosen to be the reference point. It is not indifferent which vertex is chosen as reference point since, when pieces can only be placed at a finite number of dots, different reference points may lead to different solutions to the problem.

When considering piece types t and u at rotations r and s, respectively, the nofit polygon between them ($NFP_{tr,us}$) reduces the evaluation of the overlap between pieces of these two types to the analysis of the relative position between the reference point of a piece of type u at rotation s and the $NFP_{tr,us}$. The pieces intersect if the reference point of a piece of type u at rotation s is inside the $NFP_{tr,us}$ and they do not if the reference point of a piece of type u at rotation s lies outside $NFP_{tr,us}$. Should the reference point of a piece of type u be on the boundary of $NFP_{tr,us}$, then the pieces touch each other. Figure 1 illustrates the nofit polygon between a piece of type t at rotation r and a piece of type u at rotation s.

The inner fit polygon of a piece of type t at rotation r (IFP_{tr}) represents all the positions where the reference point of a piece of this type can be placed while keeping the piece entirely inside the board. For the irregular strip packing problem, the IFP is always a rectangle of fixed width and infinite length. Consider l_{tr}^{min} the horizontal distance from the leftmost piece vertex to the reference point, h_{tr}^{min} the vertical distance from the highest piece vertex to the reference point, and h_{tr}^{max} the vertical dis-
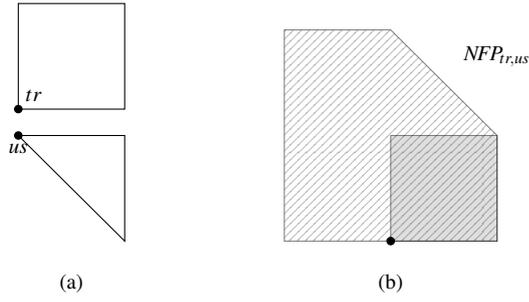
Fig. 1: In (a), a piece of type t at rotation r and a piece of type u at rotation s are illustrated. $NFP_{tr,us}$ is presented in (b).

tance from the lowest piece vertex to the reference point. The constant l_{tr}^{max} represents the distance of the piece reference point to its rightmost vertex, which is essential to evaluate the solution length. The piece type dimensions are illustrated in Figure 2, in which the reference point is highlighted. The IFP_{tr} is the rectangle ranging $[l_{tr}^{min}, \infty)$ in the horizontal axis and $[h_{tr}^{min}, H - h_{tr}^{max}]$ in the vertical axis, as presented in Figure 3.
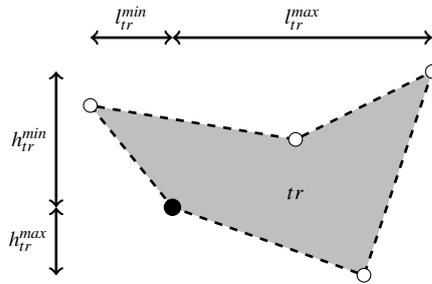

Fig. 2: Representation of a piece type, illustrating the horizontal and vertical distances from the reference point to the extreme vertices.

3 The proposed Dot Data Structure

When solving nesting problems, choosing and developing geometric tools to enforce that pieces do not overlap and are contained inside the board is an important task, since the efficiency and robustness of the solution methods generally depend on those tools. In this section, we present a data structure that stores all the geometric information necessary to the development of solution methods based on discretized feasible regions.

The proposed data structure based on the IFPs and NFPs allows an easy and efficient retrieval of the unviable dots for each piece type, once another piece of a
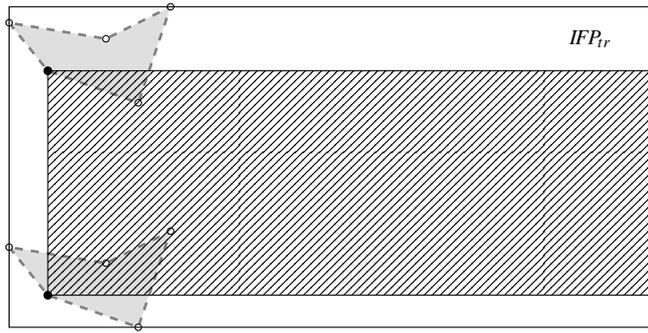
Fig. 3: Inner fit polygon of a piece of type t at rotation r.

given type is positioned at a viable dot. To achieve this goal, the structure comprises two levels. In the first level, a (dot, piece type) list of admissible piece types at each dot is computed in accordance with the IFP originated, and stored for later use. In the second level, for any given combination of every item in the aforementioned list with each piece type, a list of infeasible dots is computed in conformity with the resulting NFP, and again stored. As pieces are subject to rotation, by "piece type" a piece type at a given rotation angle is to be understood.
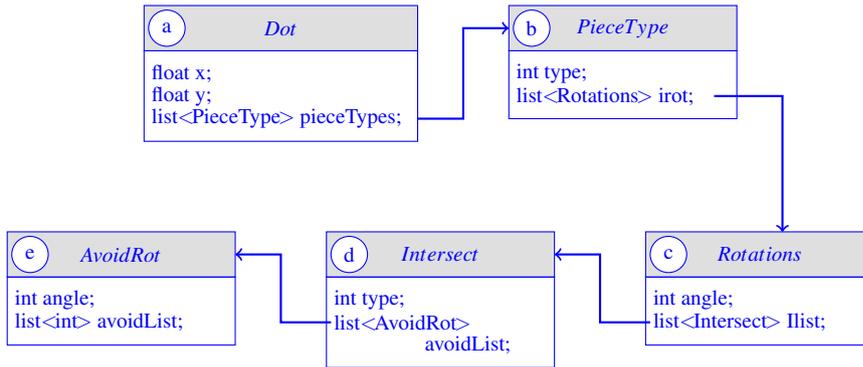

Fig. 4: Data types used in the data structure.

Any dot (Figure 4a) can only be added to the dot list if it lies inside the board and if at least one piece type can be placed at that dot. A piece type t at rotation r can only be added to the piece type list of a dot if the latter lies inside the IFP_{tr}, which ensures that a piece of this type is entirely inside the board (Figures 4b and 4c). The piece types at each dot of the structure are therefore always inside the board, i.e. the IFPs are embedded in the data structure. The set of possible placement positions for each piece type $t \in T$ at rotation $r \in R_t$, $D_{t[r]}$, can be obtained from this list.

If a piece of type t at rotation r is placed at dot d, its intersection list is a vector in which each position represents a piece type u. This vector contains the list of all the

possible rotations $s \in R_u$ of piece u (Figure 4d). Each one of these elements points to the list of forbidden dots for piece type u at rotation s (Figure 4e). The intersection list carries all the overlap information for each piece type placed at a dot ($\Phi^d_{t[r],u[s]}$), wherefore this list can be used as an alternative to the $NFP_{tr,us}$. Figure 5 shows an example of this structure.
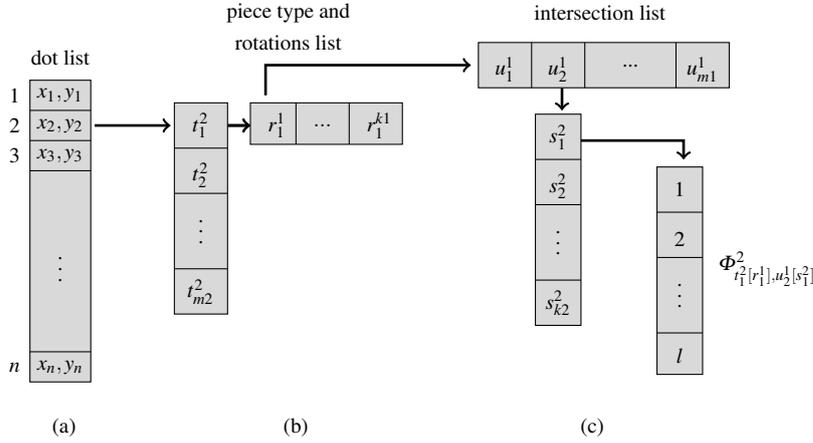

Fig. 5: The dot data structure. (a) the dot list and (b) the piece list of dot 2 together with the admissible rotations for piece type 1. The intersection list presented in (c) contains $\Phi^2_{t^2_1[r^1_1],u^1_2[s^2_1]}$, the set of dots where a piece of type u^1_2 at rotation s^2_1 overlaps piece of type t^2_1 at rotation r^1_1, when the later is placed at dot 2.

The structure is built in two phases. In the first phase, the dot list and the respective piece type list are generated as illustrated in Figure 5a, together with a list of admissible rotations at this dot (Figure 5b).

In the second phase, the intersection list for each piece type and rotation must be defined. For each piece of type t at rotation r in the piece list, an intersection list with each piece type $u \in T$ at rotation $s \in R_u$ is built. This list contains the dots inside the $NFP_{tr,us}$, *i.e.*, those that lead to an overlap between pieces of these two types: $\Phi^d_{t[r],u[s]}$ (Figure 5c). This process is repeated for each dot in the dot list.

It should be noticed that a rule to generate the dots needs not be specified, i.e. the dots may even be randomly distributed on the board. This characteristic is very useful in the methodology for the development of a solution, since the positioning of the dots on the board can be based on geometric information specific to each instance.

By way of example, figures 6 – 8 illustrate the application of the dot data structure to an instance consisting of an irregular board and two piece types, a triangle and a square. The triangle has four possible rotations, $0°$, $90°$, $180°$, $270°$, and the square has only one rotation $0°$ (Figure 6). Figure 7 shows an irregular board with a hole (shaded area). The dotted lines represent the nofit polygon between the triangle, already placed on the board, and the square that we want to add to the same board.

Fig. 6: An application of the dot data structure – piece types and their admissible rotations.
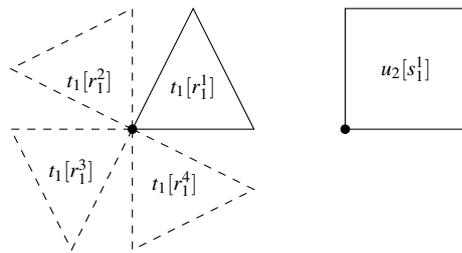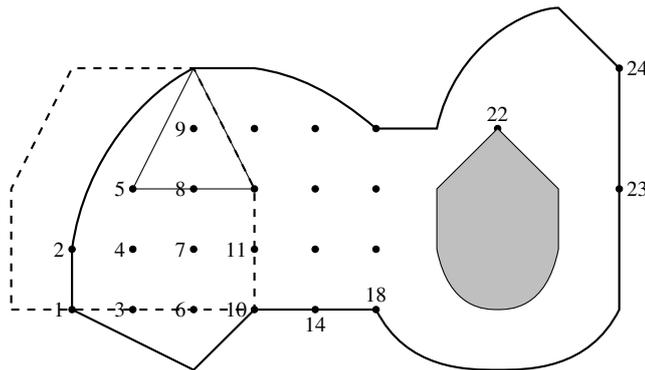

Fig. 7: An application of the dot data structure – board and feasible dots.
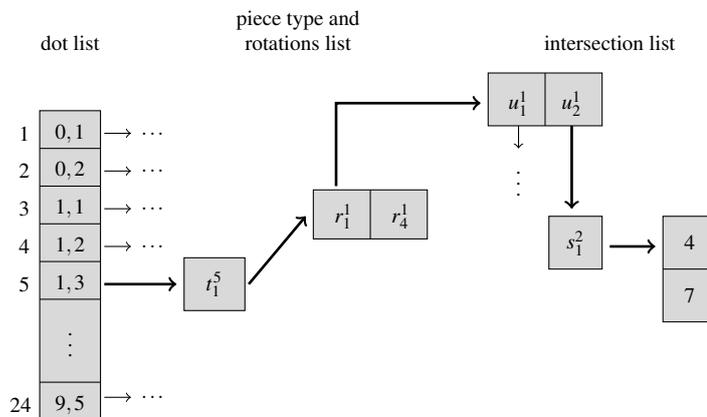

Fig. 8: An application of the dot data structure – intersection list with a piece of type 2 when a piece of type 1 is placed at dot 5.

The marks on the irregular board represent the dots where at least one of the piece types may be placed at one allowable rotation, wherefore these are the only dots in-

serted into the data structure. For example, at dot 1 only the triangle at rotation $0°$ may be positioned whereas at dot 11 the triangle can be positioned at rotations $0°$, $90°$, $180°$, as well as the square. As to dot 5, the square cannot be positioned there, but the triangle may, at rotations, $0°$ and $270°$. If we opt for the triangle at rotation $0°$, dots 4 and 7, which lie inside the nofit polygon between the two pieces, will be on the intersection list $\Phi^5_{t^5_1[r^1_1],u^1_2[s^2_1]}$ ($\Phi^5_{triangle[0°],square[0°]}$) (Figure 8).

It is important to highlight that this structure can be used in the resolution of all variations of nesting problems since it contains information about the admissible positioning points of the pieces on the board, and also on the overlap between the pieces, characteristics always present in all variations of nesting problems.

4 The dotted-board model

In this section the dotted-board model proposed in Toledo et al. [23] is adapted to the new dot data structure. As detailed in section 5, the versatility of this new data structure permits the design of two meshes in alternative to the original regular mesh for this model. The proposed dot data structure can additionally be applied and used in any approach that considers a discrete set of feasible placement positions.

In the dotted-board model, piece reference points can only be placed at discrete points of a regular mesh – the board dots. A regular mesh is defined by populating the board with dots at vertical and horizontal spacing Δy and Δx, respectively. Unlike the new meshes proposed in this paper, in a regular mesh this spacing remains invariable, irrespectively of piece type and rotation.

Any dot on the board is considered a feasible placement point so long as pieces are entirely contained inside the board and do not overlap. The containment in the board is warranted by the inner fit polygon constraints and the non-overlap is ensured by the nofit polygon constraints. Both sets of constraints have been adopted in the model of Toledo et al. [23]. However, since the dot data structure defined in Section 3 contains in itself the inner fit polygon constrains, and non-contained placements are not even included in the dot list, the inner fit polygon constraints needn't be considered. Moreover, since piece rotation was not permitted in the original dotted-board model, this model needs now to be extended to allow for different piece orientations, namely the decision variables of the dotted-board model must include this information. Hence, variable δ^d_{tr} is adopted, assuming value 1 if the reference point of a piece of type $t \in T$ at rotation $r \in R_t$ is placed at dot $d \in D_{t[r]}$, and 0 if otherwise.

The improved dotted-board model allowing for piece rotation is presented in (1) - (6).

$$\text{minimize} \quad L \tag{1}$$

$$\text{subject to:} \quad (d_x + l_{tr}^{max})\delta_{tr}^d \leq L \qquad \forall t \in T, r \in R_t, d \in D_{t[r]} \tag{2}$$

$$\delta_{tr}^d + \delta_{us}^{d'} \leq 1 \qquad \forall t \in T, u \in T, r \in R_t, s \in R_u,$$
$$d \in D_{t[r]}, d' \in \Phi_{t[r],u[s]}^d \tag{3}$$

$$\sum_{r \in R_t} \sum_{d \in D_{t[r]}} \delta_{tr}^d = q_t \qquad \forall t \in T \tag{4}$$

$$\delta_{tr}^d \in \{0,1\} \qquad \forall t \in T, r \in R_t, d \in D_{t[r]} \tag{5}$$

$$L \in \mathbb{R}_+ \tag{6}$$

The objective function (1) aims to minimize the used up board length. Consider that the reference point of a piece of type t at rotation r is placed at dot d with coordinates (d_x, d_y). Consider also that this piece type has the dimensions presented in Figure 2. To ensure that auxiliary variable L is equal or greater than the used board length, constraint (2) must hold. Constraint (3) guarantees, by making use of the nofit polygon, that the pieces do not overlap. Considering that a piece of type t at rotation r is placed at dot d, all dots d' leading to an intersection between this piece and a piece of type u at rotation s belong to the set $\Phi_{t[r],u[s]}^d$, *i.e.*, $d' \in \Phi_{t[r],u[s]}^d$. Constraint (4) ensures that, for each piece type t, the requested number of pieces (q_t) is placed. Constraints (5) and (6) define the domains of the variables.

The scope of the new model is broader than that of the model proposed in Toledo et al. [23] for it allows for piece rotation. Moreover, due to the new dot data structure, the model is mesh-type independent.

5 Mesh generation rules

In this section, some examples of mesh generation rules are presented, as enabled by the new data structure. The first one is a piece-based mesh, in which the distances between the dots are based on the distances between the piece type vertices, while the second one is an *NFP*-based mesh, based namely on a cloud of points belonging to the *NFP* of a given pair of piece types. The description of these meshes and the procedures to build them follows.

5.1 Piece-based mesh

Using the same mesh of dots for all piece types may not offer the best solution to some instances. One given mesh may be too refined for one piece and too coarse for another. Two problems arise from this fact:

a. unnecessary dots may be generated, which implies an increased complexity and computational burden;

b. some pieces may have excessively few positioning dots available, which prompts a bad fit between pieces, and a consequent increase in waste.

To overcome this problem, an approach in which each piece type has its own mesh is proposed. It should be noticed that there is no inter-dependence between meshes of two different piece types.

For each piece of type $t \in T$ at rotation $r \in R_t$, a specific mesh $D_{t[r]}$ is defined based on the constants Δx_{tr} and Δy_{tr}. The value of Δx_{tr} is the minimum horizontal distance between two vertices of piece type t at rotation r. The constant Δy_{tr} is obtained through the same procedure, but using the vertical distances between the piece type vertices. A minimum mesh resolution (bx_{tr} and by_{tr}) is imposed to avoid excessively refined meshes.

The mesh for each piece type t at rotation r is generated using constants Δx_{tr} and Δy_{tr} (refer to Figure 9(a), where $\Delta x_{tr} = 2.0$ and $\Delta y_{tr} = 3.0$).

The horizontal direction lines are generated at intervals of Δy_{tr} units (Figure 9(b)). In this case, two starting points are used, the highest and the lowest points on the board. This construction leads to a non-regular mesh but ensures that the pieces can always touch the boundaries of the board (Figure 9(c)). The vertical direction lines are generated in intervals of Δx_{tr} units. The dots at the crossings between the horizontal and vertical lines compose the $D_{t[r]}$ set (Figure 9(d)). In Figure 10 a piece-based mesh for two piece types is represented, and it is important to highlight that this type of dot distribution along the board would be very difficult to implement without the new dot data structure.

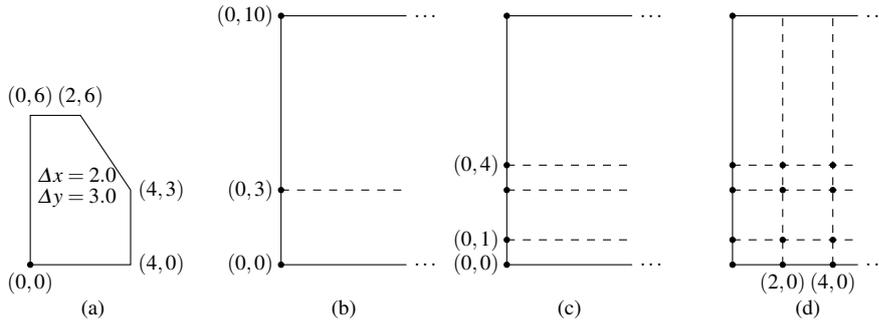Fig. 9: Example of a piece-based mesh for one piece type.

Algorithms 1 and 2 present the procedures to generate the dot list using the piece-based mesh. The procedure to build the intersection list is presented in Algorithm 3.

5.2 *NFP-based mesh*

Another approach to bring the geometric characteristics of the instances into the mesh is to generate a mesh based on the nofit polygon obtained from each pair of piece
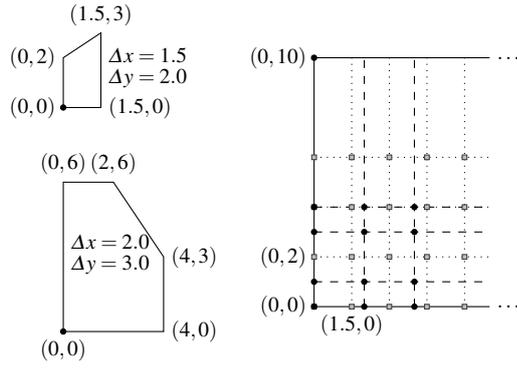
Fig. 10: An example of a piece-based mesh for two piece types. The squares and circles represent feasible placement dots for the smaller and larger piece types, respectively.

Algorithm 1: GENERATION OF A PIECE-BASED MESH

Input: T, R_t, W and H.
Output: $D[\ ]$ set of dots.

1 $D[\ ] \leftarrow \emptyset$;
2 **for** $t \in T$ **do**
3   **for** $r \in R_t$ **do**
4   calculate Δx_{tr} and Δy_{tr};
5          $x = l_{tr}^{min}$;
6   **while** $x \leq W - l_{tr}^{max}$ **do**
7              $y = h_{tr}^{min}$;
8   **while** $y \leq H - h_{tr}^{max}$ **do**
9                  $d = (x, y)$;
10                 $D[\ ] = \text{UPDATE\_DOT\_LIST}(D[\ ], d, t, r)$;
11                 $y = y + \Delta y_{tr}$;
12   **end**
13             $y = H - h_{tr}^{max}$;
14   **while** $y \geq h_{tr}^{min}$ **do**
15                 $d = (x, y)$;
16                 $D[\ ] = \text{UPDATE\_DOT\_LIST}(D[\ ], d, t, r)$;
17                 $y = y - \Delta y_{tr}$;
18   **end**
19             $x = x + \Delta x_{tr}$;
20   **end**
21   **end**
22 **end**
23 **return** $D[\ ]$;

---

types, thus promoting the fit between the pieces as, by definition, the interior of the nofit polygon represents the set of points where the pieces overlap.

To understand this mesh generation methodology, consider piece types t and u at rotations r and s, respectively, and the corresponding $NFP_{tr,us}$, as illustrated in Figure 11(a). Consider a piece of type t at rotation r (p_{tr}), placed at the lower left

---

Algorithm 2: UPDATE_DOT_LIST.

Input: $D[\ ]$, d, t and r.
Output: $D[\ ]$.

1 **if** $d \notin D[\ ]$ **then**
2   $D[\ ] = D[\ ] \bigcup d$;
3   $D[d] \leftarrow \emptyset$;
4 **end**
5 **if** $t \notin D[d]$ **then**
6   $D[d] = D[d] \bigcup t$;
7   $D[d][t] \leftarrow \emptyset$;
8 **end**
9 $D[d][t] = D[d][t] \bigcup r$;
10 $D[d][t][r] \leftarrow \emptyset$;
11 **return** $D[\ ]$;

Algorithm 3: INTERSECTION LIST GENERATION

Input: d, $D[\ ]$ and T.
Output: $\Phi^d_{t[r],u[s]}$.

1 $\Phi^d_{t[r],u[s]} \leftarrow \emptyset$;
2 **for** $d' \in D[\ ]$ **do**
3   **for** $t, u \in T$ **do**
4   **for** $r \in R_t$ and $s \in R_u$ **do**
5   **if** $d' \in NFP_{tr,us}$ **then**
6      $\Phi^d_{t[r],u[s]} = \Phi^d_{t[r],u[s]} \bigcup d'$;
7   **end**
8   **end**
9   **end**
10 **end**
11 **return** $\Phi^d_{t[r],u[s]}$;

---

feasible corner of the board. The vertices of the $NFP_{tr,us}$ that are inside the board are inserted into the dot list as feasible placement positions for piece type u at rotation s. The dot where p_{tr} is placed is added to the dot list of this piece type at the assigned rotation (Figure 11(b)). Piece p_{tr} is then translated vertically by $h_{tr}^{min} + h_{tr}^{max}$ and the same procedure is applied until p_{tr} reaches the uppermost admissible position on the board (Figure 11(c)). Piece p_{tr} is then translated horizontally by $l_{tr}^{min} + l_{tr}^{max}$ and the whole process is repeated as long as p_{tr} can be moved horizontally or vertically (Figure 11(d)). The NFP-based mesh is build by repeating this process for all pairs of piece types $t, u \in T$ at rotations $r \in R_t$ and $s \in R_s$, respectively.

Variations in this mesh generation methodology can be obtained by inserting some dots in the dot list, which are not vertices of the nofit polygons (e.g. the middle points on the edges of the nofit polygons), thus improving the accuracy of the approximation. As the NFP-based mesh is generated based on the nofit polygon of each pair of pieces, placements in positions where pieces touch each other are favored. However, as the translation of the pieces is discrete, gaps between pieces may still occur.
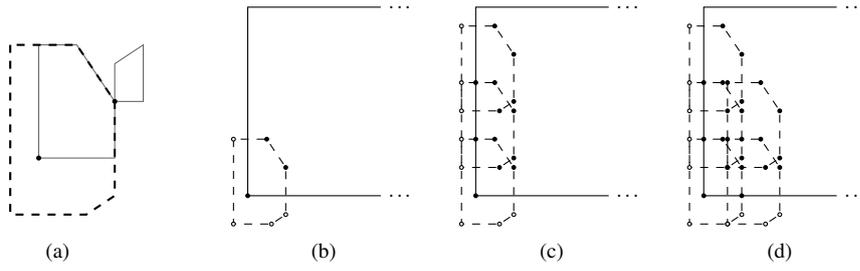
Fig. 11: An example of an *NFP*-based mesh for the piece types presented in Figure 10.

The procedure to generate the dot list using the *NFP*-based mesh is presented in Algorithm 4. Algorithm 3 can be used to generate the intersection list of each dot.

Algorithm 4: *NFP*-BASED MESH GENERATION

Input: T, R_t, $NFP_{tr,us}$, IFP_{tr}, W and H.
Output: $D[\ ]$.

1   $D[\ ] \leftarrow \emptyset$;
2   **for** $t \in T$ *and* $u \in T$ **do**
3   **for** $r \in R_t$ *and* $s \in R_u$ **do**
4          $\Delta x_{tr} = l_{tr}^{min} + l_{tr}^{max}$;
5          $\Delta y_{tr} = h_{tr}^{min} + h_{tr}^{max}$;
6          $x = l_{tr}^{min}$;
7   **while** $x \leq W - l_{tr}^{max}$ **do**
8             $y = h_{tr}^{min}$;
9   **while** $y \leq H - h_{tr}^{max}$ **do**
10                $d = (x,y)$;
11                $D[\ ] = $ UPDATE_DOT_LIST$(D[\ ],d,t,r)$;
12   **for** $d' \in NFP_{tr,us}$ **do**
13   **if** $d' \in IFP_{us}$ **then**
14                      $D[\ ] = $ UPDATE_DOT_LIST$(D[\ ],d',u,s)$;
15   **end**
16   **end**
17                $y = y + \Delta y_{tr}$;
18   **end**
19             $x = x + \Delta x_{tr}$;
20   **end**
21   **end**
22   **end**
23   **return** $D[\ ]$;

---

6 Complexity analyses of the structure in time and space

In this section, the time and space complexity of the proposed structure is discussed.

The time required to generate the data structure can be divided into two distinct phases: (i) the time consumed to create the dot list with the associated piece types list, and (ii) the time needed to analyze the overlapping of the piece types at multiple rotations. While the first phase depends on the rule to generate the dots, the second phase does not depend on this rule.

To establish the piece-based mesh, constants Δx_{tr} and Δy_{tr} need to be determined for each piece type, $t \in T$, $r \in R_t$. The number of dots generated on the board is approximately $\frac{H \times W}{\Delta x_{tr} \times \Delta y_{tr}}$, and therefore the maximum number of dots considered on the board for any given piece type is $\frac{2H \times W}{\overline{\Delta}}$, where $\overline{\Delta} = \min_{t \in T, r \in R_t}\{\Delta x_{tr} \times \Delta y_{tr}\}$. Let V_{tr} be the set of vertices of piece type t at rotation r and $\overline{V} = \max_{t \in T, r \in R_t}\{V_{tr}\}$. The analysis of all different pairs of vertices is required to find Δx_{tr} and Δy_{tr}, i.e. $\sum_{i=1}^{|V_{tr}|}|V_{tr}-i| = \frac{|V_{tr}-1| \times |V_{tr}|}{2}$. In the worst case, this value is $\frac{\overline{V}^2}{2}$, wherefore the number of operations performed in phase (i) is, in the worst possible case, as presented in Formula (7),

$$O\left(|T| \times \overline{R} \times \frac{2H \times W}{\overline{\Delta}} + |T| \times \overline{R} \times \frac{\overline{V}^2}{2}\right) \qquad (7)$$

where $|.|$ is the cardinality of the set, and $\overline{R} = \max t \in T\{|R_t|\}$. Notice that the height of the board H must be multiplied by two since the piece-based mesh generation procedure has two starting points in the y-axis.

On the other hand, the nofit polygon vertices are used to build the *NFP*-based mesh. Specifically, for each pair piece types t and u at rotations r and s, respectively, the vertices of $NFP_{tr,us}$ ($V_{NFP_{tr,us}}$) need to be placed on the board as mesh dots. This procedure is systematically repeated in $\Delta x_{tr} = l_{tr}^{min} + l_{tr}^{max}$ and $\Delta y_{tr} = h_{tr}^{min} + h_{tr}^{max}$ steps. The maximum number of vertices of $NFP_{tr,us}$ is $V_{NFP} = \max_{t,u \in T, r \in R_t, s \in R_u}\{\overline{V}_{NFP_{tr,us}}\}$ and, as in the piece-based mesh, the maximum number of steps is obtained when $\overline{\Delta}$ is $\min_{t \in T, r \in R_t}\{\Delta x_{tr} \times \Delta y_{tr}\}$. Formula (8) shows the worst case complexity to generate the phase (i) dots.

$$O\left(|T|^2 \times \overline{R}^2 \times \frac{H \times W}{\overline{\Delta}} \times \overline{V}_{NFP}\right) \qquad (8)$$

If a regular mesh is built, all the dots of a regular mesh with $\Delta x_{tr} = \Delta y_{tr} = \Delta$ need to be added to the board for each piece type. For this mesh, the worst case complexity of the phase (i) is presented by Formula (9).

$$O\left(\overline{T} \times \overline{R} \times \frac{H \times W}{\Delta^2}\right) \qquad (9)$$

Once the dot list is created, and the set of dots D is thus obtained, the positioning overlap needs to be examined for all pairs of piece types placed at different dots. In other words, for each piece type $t \in T$ and $u \in T$ at rotations $r \in R_t$ and $s \in R_u$, respectively, the overlapping needs to be analyzed for each possible combination

of dots where these piece types may be placed. The number of operations required to verify if overlapping occurs is proportional to $|V_{NFP_{tr,us}}|$. In the worst case, the structure generation complexity is given by Formula (10).

$$O\left(|D|^2 \times |T|^2 \times \overline{R^2} \times \overline{V}_{NFP}\right) \tag{10}$$

Hence, regarding the worst case, the time complexity to build the structure is given by the sum of Formula (10) with either Formula (7), Formula (8) or Formula (9), depending on the mesh considered. Since these formulas are based on a worst case estimation, the actual running time may exhibit better results.

The space complexity issue – that measures the amount of memory used to represent the data structure – is no less important. The structure consists of a list of $|D|$ dots. As depicted in Figures 4b and 4c, a piece type/rotation list is associated with each dot in this dot list, which includes all piece types that can be placed at the said dot at any given rotation. Finally, one intersection list is associated with each item on the aforementioned piece type/rotation list, which includes all those dots where the placement of any given second piece type at any given rotation leads to overlapping of the pieces. The size of this list is exactly $|\Phi^d_{t[r],u[s]}|$ and this list has at the most $\overline{\Phi} = \max_{d \in D, t, u \in T, r \in R_t, s \in R_u}\{|\Phi^d_{t[r],u[s]}|\}$ elements. Formula (11) represents thus the amount of memory used by the proposed structure.

$$O\left(|D| \times |T|^2 \times \overline{R}^2 \times \overline{\Phi}\right) \tag{11}$$

7 Computational experiments

This section presents the computational results obtained with the dotted-board model using the different meshes proposed in Section 5, and compares the results with those obtained with the regular mesh used in [23]. The adopted instances aim at demonstrating the advantages of the proposed meshes in the resolution of problems with specific characteristics.

7.1 Framework and instances

The computational experiments were run on an Intel Xeon E5-2450 @ 2.10GHz processor with 32 GB of memory. The algorithms were coded in C/C++ language and the models were solved using IBM ILOG CPLEX 12.6.1 optimization library.

For each mesh type proposed in Section 5, a different instance is used to evaluate the quality of the proposed mesh in the solving of problems with specific characteristics. The piece-based mesh should befit the geometry of problems with pieces of different sizes. These characteristics are common in several applications, such as furniture manufacturing and sheet-metal cutting industries. To represent these problems
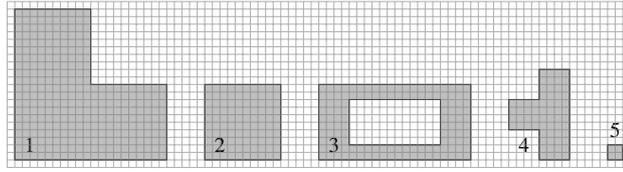
Fig. 12: The MF instance set of pieces.

the MF (Metalworking/Furniture) instance set is proposed. The instances in this set are composed of five piece types as presented in Figure 12.

Each instance of the MF set is created by using these piece types with four allowed rotations. Instance MF1 is composed of one unit of piece types 1, 2 and 3, three units of piece type 4 and twenty-six units of piece type 5. MFi instances are obtained by multiplying the piece type requirements by i. For all these instances, the board height is 30. Since an assessment of the board length is necessary, we adopt the initial value of 32 for MF1, which is multiplied by i for the MFi instances.

On the other hand, the *NFP*-based mesh should handle properly problems where the pairs of pieces fit well. Several problems in different applications can have these characteristics. To represent these problems the CS (Clothes/Shoes) instance set is adopted. Three piece types, represented in Figure 13, compose the instances of this set, and each piece type is allowed four rotations.
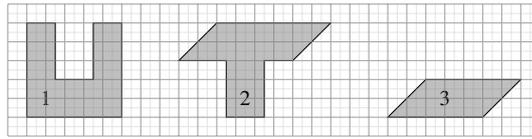

Fig. 13: The CS instance set of pieces.

The board for instance CS1 has a height of 11 units and an initial length of 13 units to accommodate two copies of each piece type presented in Figure 13. The other instances CSi that compose the set have the same board height and an initial board length of $13 \times i$ units. For each instance CSi the number of pieces of each type represented in Figure 13 is $2 \times i$.

Instances from the literature are also used in the computational experiments, and a 25% increase on the length of the known best solution is used as an initial estimate of the required board length. For some meshes, this board size can be clearly insufficient to find a feasible solution, but as the purpose of the problem is the minimization of the board length, any solution larger than this upper boundary is not minimally competitive.

These instances and their characteristics are shown in Table 1. The first column presents the instance name, and the second and third display the total number of pieces and piece types, respectively. The rotation step of the pieces is given in the fourth column. Although each orientation of a specific piece type is considered a

different piece type, a set of rotations of a polygon can lead to the same polygon and these symmetrical rotations can be eliminated from the model. The fifth column shows thus the number of pieces and their non symmetric rotations. The board dimensions are introduced in the sixth and seventh columns, respectively. Finally, we find a reference to the instance source in the last column. It should be noted that these instances were chosen because the model built with a regular mesh (with $\Delta = 1$) does not exceed the available memory when loaded.

Table 1: Test instances from the literature and their characteristics.

Instance	Pieces demand	Piece types	Rotation step	Rotated types	Board height	UB length	Origin
poly1a0	15	15	0	15	40	20	[16]
shapes0	43	4	0	4	40	73	[21]
shapes1	43	4	180	6	40	70	[21]
shirts	99	8	180	14	40	78	[8]
blaz1	28	7	180	10	15	34	[22]
blaz2	16	4	180	5	15	26	[22]
jakobs1	25	22	90	62	35	15	[18]
jakobs2	25	21	90	63	70	29	[18]
fu	12	11	90	34	38	39	[14]

7.2 Piece-based mesh computational results

The piece-based mesh presented in Section 5.1 is defined in accordance with the distances between the vertices of each piece type and is specially appropriate for instances with piece types of distinct shapes and sizes. Consequently, the piece-based mesh can contain dots where only a few pieces can be placed, which results in a model with less variables than a model build using the same number of dots in a regular mesh.

In this section, we present the computational results obtained using the piece-based mesh to solve the MF instance set. The meshes were generated adopting different minimum mesh refinements in the x and y axes for each piece type and rotation, namely bx_{tr} and by_{tr} defined as $\frac{1}{8} \times (l_{tr}^{min} + l_{tr}^{max})$ and $\frac{1}{8} \times (h_{tr}^{min} + h_{tr}^{max})$ respectively.

The computational results for the dotted-board model using the piece-based mesh are presented in Table 2. In this table, the first column shows the instance's designation. The second, third, fourth, fifth and six columns display the number of variables and constraints of the resulting model (variables, constraints), the best solution found so far (UB), the optimality gap as a percentage (GAP = $\frac{UB - LB}{UB}$, where LB is the solution's lower boundary provided by CPLEX at the end of execution), and the computational time (in seconds) achieved with the piece-based mesh. The sixth to the ninth columns exhibit the same contents obtained with a regular mesh of granularity $\Delta = 2$.

Computational experiments were also conducted using a regular mesh of granularity $\Delta = 1$, but the only instance for which a solution was found was MF1, a solution with 32 units of length and with a GAP of 12.91%. For the other instances

the model built using this granularity could not produce a feasible solution within the 3600 seconds time limit. For these reasons, the results of these experiments have not been considered in this work.

Table 2: Computational experiments using the piece-based mesh generation rule.

| | Piece-based mesh | | | | | Regular mesh ($\Delta = 2$) | | | | |
| | Number of | | UB | GAP | Time (sec) | Number of | | UB | GAP | Time (sec) |
Inst.	var.	constr.				var.	constr.			
MF1	1218	55897	28.0	0.00%	2.3	1213	289628	30.0	0.00%	32.7
MF2	2554	136035	56.0	0.00%	19.2	2989	1044705	58.0	3.91%	3600.0
MF3	3902	219455	84.0	0.00%	32.9	4765	1802881	86.0	2.79%	3600.0
MF4	5238	299400	112.0	0.00%	149.3	6541	2561057	118.0	5.53%	3600.0
MF5	6594	386691	140.0	0.00%	289.4	8317	3319233	146.0	4.57%	3600.0
MF6	7930	466969	168.0	0.00%	790.0	10093	4077409	-	-	-
MF8	10614	630527	224.0	0.00%	893.0	13645	5593761	-	-	-
MF10	13306	797763	280.0	0.47%	3600.3	17197	7110113	-	-	-
MF15	20018	1208835	480.0	12.92%	3600.5	26077	10900993	-	-	-

-: No feasible solution was found within the 3600 seconds time limit.

Using the piece-based mesh, optimality was proven for all instances except MF10 and MF15, whereas with the regular mesh with $\Delta = 2$, optimality could only be proven for the MF1 instance. Moreover, in the case of the regular mesh, only for the smallest instances, MF1 to MF5, was there a solution to be found. Conversely, with the piece-based mesh, it was possible to find feasible solutions for instances with as many as up to three times more pieces than with the regular mesh.

It should be noticed that the piece-based mesh and regular mesh generation rules create models with different solution spaces, wherefore distinct solutions - even optimal for their own mesh - may be reached. This occurs with for instance MF1, where the optimal solution obtained with the piece-based mesh model differs from the one obtained with the regular mesh.

For the instances used, the model built using the piece-based mesh found solutions for all instances, and for problems with almost the same number of variables, optimality was proven quicker. For example, for instance MF5, the mesh by pieces generates a model with 6594 variables, equivalent to the 6541 variables model obtained for instance MF4 with the regular mesh. Nevertheless, the piece-based mesh model proved the solution optimality whereas the regular mesh based model did not. This behavior was expected, since, when using the piece-based mesh, larger pieces have smaller sets of feasible dots, which reduces the number of constraints needed to avoid pieces overlap. The number of constraints of the model built on the piece-based mesh is, for all instances, one order of magnitude smaller than the equivalent model built on a regular mesh.This fact highlights the importance of an intelligent choice of a mesh that can directly improve the convergence of the models, and meets the need for a data structure that enables a simple representation of these special meshes.

The time required to generate the mesh of dots was in average 0.93 seconds for the piece-based mesh, varying between 0.02 and 3.92 seconds. By comparison, the

generation of the regular mesh needed in average 1.62 seconds, varying from 0.06 to 5.46 seconds.

The optimal solutions for the MF1 and MF2 instances obtained using the piece-based mesh are represented in Figures 14(a) and 14(b), respectively. Figures 15(a) and 15(b) show the optimal solutions obtained for the MF1 and MF2 instances with the regular mesh ($\Delta = 2$). Comparing the solutions obtained with the two meshes, it can be verified that better solutions have been achieved with the piece-based mesh, which better exploits the large piece's hole and the t-shape's concavities to place the small squares, whereas, given the concrete dimensions defined for the pieces, such placements are not possible with the regular mesh. Figure 16 represents the optimal solution for the MF8 instance obtained with the piece-based mesh.
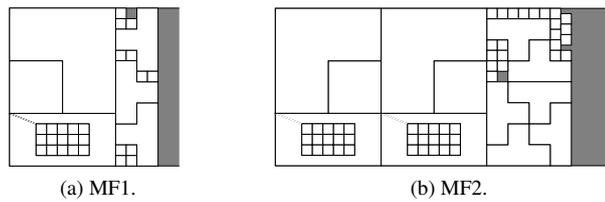

(a) MF1.   (b) MF2.

Fig. 14: Optimal solutions for the MF1 and MF2 instances, obtained with the piece-based mesh.

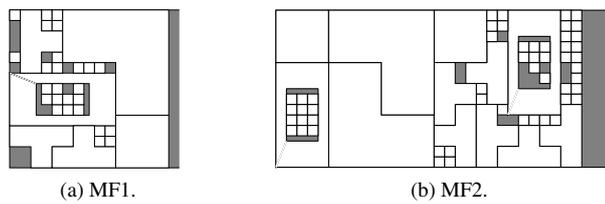(a) MF1.   (b) MF2.

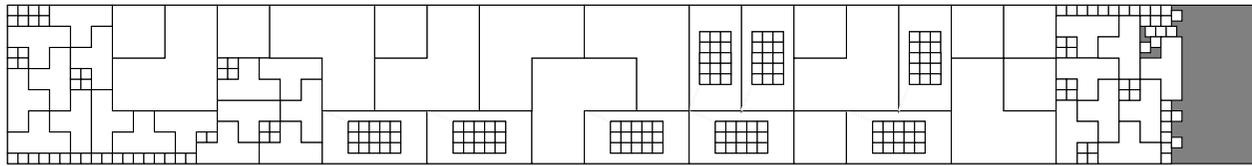Fig. 15: Optimal solutions for the MF1 and MF2 instances, obtained with the regular mesh.

Fig. 16: Solution for the MF8 instance obtained with the piece-based mesh.

7.3 *NFP*-based mesh computational results

In this section, the results of the computational experiments obtained for the CS instances using the *NFP*-based mesh (Section 5.2) are presented. The *NFP*-based mesh is created in an attempt to remove the regularity present in regular and piece-based meshes. The mesh is generated using all vertices and midpoints of the edges of the nofit polygons. The results obtained with the model built with the *NFP*-based mesh are compared with the results obtained with a regular mesh of granularity $\Delta = 0.5$. These results are presented in Table 3 where the columns have the same type of content as described in Table 2.

Table 3: Results of the computational experiments, obtained using *NFP*-based mesh and regular mesh generation rules.

| | *NFP*-based mesh | | | | | Regular mesh ($\Delta = 0.5$) | | | | |
| | Number of | | | | Time | Number of | | | | Time |
Inst.	var.	constr.	UB	GAP	(sec)	var.	constr.	UB	GAP	(sec)
CS1	457	30345	12.0	0.00%	2.4	2052	1095995	12.0	0.00%	1281.3
CS2	1562	156407	22.0	0.00%	105.8	5588	3962371	26.0	34.27%	3600.0
CS3	2521	265574	32.0	0.00%	1072.1	9124	6829131	39.0	34.26%	3600.0
CS4	3620	394039	43.5	21.29%	3600.3	12660	9695891	52.0	34.27%	3600.0
CS5	4683	514724	52.5	18.57%	3600.0	16196	12562651	65.0	34.26%	3600.0
CS6	5634	621599	63.0	18.57%	3600.0	-	15429411	-	-	-
CS7	6714	748089	81.5	26.56%	3600.0	-	18296171	-	-	-

-: No feasible solution was found within the 3600 seconds time limit.

Both mesh generation rules, the piece-based mesh one and the *NFP*-based mesh rule, use information about the pieces to build the mesh. However, the *NFP*-based mesh uses information that relates to the interaction between pairs of pieces rather than to individual pieces, as happens with the piece-based mesh, and holds thus more information regarding the instances. When using this *NFP*-based mesh, the number of variables of the model depends strongly on the pieces, as well as on the *NFP* shapes.

For larger problems, more optimal and more feasible solutions are found with the model built using the *NFP*-based mesh than with the one based on the regular mesh. For all instances except the first one, the *NFP*-based mesh produced also strictly better solutions when compared with those obtained with the regular mesh with $\Delta = 0.5$. What is more, when optimality is not proven, the optimality gap is smaller for the *NFP*-based mesh. In addition, the number of variables of the model built using the *NFP*-based mesh is approximately $\frac{1}{4}$ of that of the model built with the regular mesh. Above all, the number of constraints of the *NFP*-based mesh model is in average 26.5 times smaller then that of the regular mesh based one. This remarkable difference was expected, since, when the vertices of the *NFP* of a pair of piece types are used to generate the dots in the mesh, the intersection between these two piece types is naturally avoided. This reinforces the statement that the right choice of placement dots for each piece type can speed up the convergence time of the model.

By using the regular mesh with $\Delta = 0.5$, an optimal solution for the CS1 instance with the same quality as the one found with the *NFP*-based mesh model is reached.

Still, the computational time to find and prove the optimality of this solution is more than five hundred times faster in the case of the model built using the *NFP*-based mesh. As to the other instances, the model built with a regular mesh with $\Delta = 0.5$ produced bad quality solutions due to the massive number of variables, time constraints and framework resources.

The *NFP*-based mesh model required in average 0.13 seconds to generate the mesh of dots, namely from 0.01 seconds to 0.29 seconds, while the regular mesh based one took in average 2.91 seconds, i.e. form 0.20 seconds to 6.23 seconds.

Figures 17(a) and 17(b) show the optimal solutions obtained with the *NFP*-based mesh for the CS1 and CS2 instances, respectively. Using the regular mesh, the optimal solution was reached for the CS1 instance (Figure 18(a)) and a feasible solution was found for the CS2 instance (Figure 18(b)). Figure 19 displays the solution found for the CS6 instance, which could only be obtained with the model built using the *NFP*-based mesh.
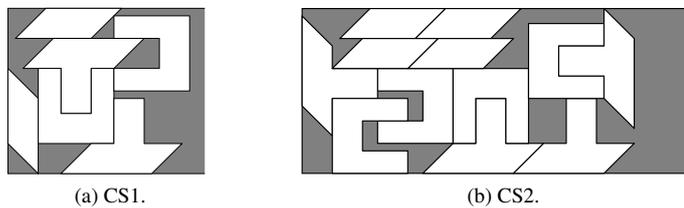

(a) CS1. (b) CS2.

Fig. 17: Optimal solutions for the CS1 and CS2 instances obtained with the *NFP*-based mesh.
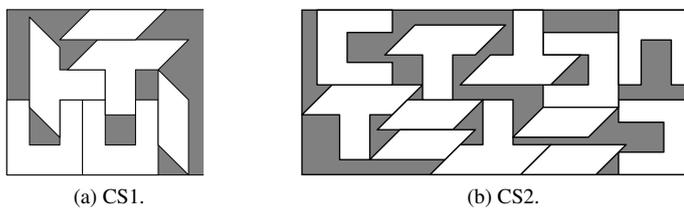

(a) CS1. (b) CS2.

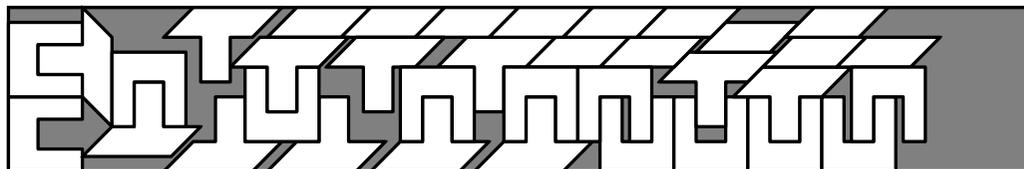Fig. 18: Optimal solution for the CS1 instance and a feasible solution for CS2 instance obtained with the regular mesh.

Fig. 19: Solution for the CS6 instance obtained by the *NFP*-based mesh.

7.4 Comparing the mesh generation rules

In the previous sections the emphasis is set on the proposed data structure, mainly on its versatility to represent different types of meshes that take advantage of special characteristics of the pieces. Using these meshes to solve instances that have the desired characteristics showed to be more promising than using a regular mesh.

Table 4 presents the computational results obtained for the MF and CS instance sets with the piece-based and the *NFP*-based meshes. The table columns display the same type of content presented in Table 2. As expected, better solutions have been obtained for the MF set of instances with the piece-based mesh model and for the CS instances set with the model built using the *NFP*-based mesh.

Table 4: Comparing results achieved with the piece-based and *NFP*-based mesh built models.

| | Piece-based mesh | | | | | *NFP*-based mesh | | | | |
| | Number of | | | | Time | Number of | | | | Time |
Inst.	var.	constr.	UB	GAP	(sec)	var.	constr.	UB	GAP	(sec)
MF1	1218	55897	28.0	0.00%	2.3	857	159540	30.0	0.00%	4.9
MF2	2554	136035	56.0	0.00%	19.2	2222	691006	58.0	0.00%	64.7
MF3	3902	219455	84.0	0.00%	32.9	3565	1220808	86.0	0.00%	2818.5
MF4	5238	299400	112.0	0.00%	149.3	4880	1737708	114.0	2.22%	3600.0
MF5	6594	386691	140.0	0.00%	289.4	6255	2278899	146.0	4.57%	3600.0
MF6	7930	466969	168.0	0.00%	790.0	7608	2811879	192.0	12.92%	3600.0
MF8	10614	630527	224.0	0.00%	893.0	10280	3862889	256.0	12.92%	3600.0
MF10	13306	797763	280.0	0.47%	3600.0	12969	4920342	-	-	-
MF15	20018	1208835	480.0	12.92%	3600.0	19716	7576733	-	-	-
CS1	275	26151	13.0	0.00%	0.7	457	30345	12.0	0.00%	2.4
CS2	701	84685	24.5	0.00%	13.1	1562	156407	22.0	0.00%	105.8
CS3	1102	138812	36.5	0.00%	209.4	2521	265574	32.0	0.00%	1072.1
CS4	1528	197522	47.0	0.00%	2896.3	3620	394039	43.5	21.29%	3600.0
CS5	1957	255648	59.0	3.39%	3600.0	4683	514724	52.5	18.57%	3600.0
CS6	2355	310356	71.0	27.51%	3600.0	5634	621599	63.0	18.57%	3600.0
CS7	2784	368350	83.0	27.66%	3600.0	6714	748089	81.5	26.56%	3600.0

-: No feasible solution was found within the 3600 seconds time limit.

These results sustain our hypothesis that the right choice of mesh is very important. For the MF instances, the solutions obtained with the piece-based mesh are always better than the ones obtained with the *NFP*-based mesh. Conversely, for the CS instances the best results are always obtained with the *NFP*-based mesh although in some cases the GAP is higher.

7.5 Computational experiments with instances from the literature

Table 5 presents the results attained with the various meshes for instances available in the literature. In the first column the instance name is presented. The second, third and fourth columns display - for the model built on a regular mesh - the best solution

found, the optimality GAP and the computational time in seconds necessary to reach the said solution, respectively . The fifth, sixth and seventh columns (eighth, ninth and tenth) present the same information for the *NFP*-based mesh (piece-based mesh).

Table 5: Computational experiments using various mesh generation rules.

Instance	Regular mesh			*NFP*-based mesh			Piece-based mesh		
	UB	GAP	Time (sec)	UB	GAP	Time (sec)	UB	GAP	Time (sec)
poly	19	31.60%	3600.0	19.0	31.60%	3600.0	**18.0**	0.00%	18.9
shapes0	-	-	-	**66.0**	35.80%	3600.0	-	-	-
shapes1	70.0	43.00%	3600.0	**60.0**	32.55%	3600.0	65.0	0.00%	1919.0
shirts	78.0	31.50%	3600.0	-	-	-	69.0	22.20%	3600.0
blaz1	29.0	26.40%	3600.0	**28.0**	23.80%	3600.0	**28.0**	0.00%	15.8
blaz2	**21.0**	0.00%	26.4	21	0.00%	911.56	24.0	0.00%	0.1
jakobs1	15.0	34.80%	3600.0	-	-	-	**12.0**	18.30%	3600.0
jakobs2	-	-	-	-	-	-	**26.0**	25.70%	3600.0
fu	-	-	-	-	-	-	**34.0**	0.00%	598.7

-: No feasible solution was found within the 3600 seconds time limit.

As different meshes lead to different solution spaces, it is natural that models based on regular, *NFP*-based and piece-based meshes produce different solutions, even if optimality may sometimes be proven in all cases. Different meshes can be more adequate for the geometric characteristics of different instances. One better solution is found with the model based on the regular mesh for one instance, while three best solutions are obtained with the model built on the *NFP*-based mesh, and five using the piece-based mesh.

Based on these results, it is possible to conclude that no single mesh of dots better befits the geometric characteristics of all instances. On the contrary, it is possible to explore the geometric characteristics of the various instances to derive new meshes from different applications (e.g. garment, metalomechanics, footware), which can represent these instances with more precision.

8 Conclusions

The dot data structure proposed in this paper has proven to be an efficient tool to represent special meshes for the dotted-board model that can be adapted to the characteristics of the instances to be solved. For the MF set of instances, using the piece-based mesh, optimality was proven for instances with up to 40 pieces, whilst for the same set of instances the regular mesh reached optimality only for the MF instance with 5 pieces. Similar, although not so sound results, were obtained with the *NFP*-based mesh and the CS instances.

In matheuristic approaches based on the dotted-board model, the proposed data structure may easily represent the meshes presented in this paper, i.e. the piece-based mesh or the *NFP*-based mesh or other special meshes. The dot data structure may also be used within heuristic approaches, such as the bottom-left heuristic, in which the dots can represent the feasible placement positions of the pieces. Note that in this

case, for each solution built by the heuristic, the mesh can change, thus allowing for different solutions for the same sequence of pieces.

Furthermore, problems with irregular boards can be easily considered using the dot structure by inserting in the dot list only the dots that keep the pieces inside the board. Boards with defects or regions where some pieces could not be cut can also be easily handled by controlling the pieces that can be placed at each dot.

References

1. Alvarez-Valdes, R., Martinez, A., Tamarit, J.: A branch & bound algorithm for cutting and packing irregularly shaped pieces. International Journal of Production Economics **145**(2), 463 – 477 (2013)
2. Bennell, J., Dowsland, K.: Hybridising tabu search with optimisation techniques for irregular stock cutting. Management Science **47**(8), 1160–1172 (2001)
3. Bennell, J., Song, X.: A beam search implementation for the irregular shape packing problem. Journal of Heuristics **16**(2), 167–188 (2010)
4. Bennell, J.A., Oliveira, J.F.: The geometry of nesting problems: A tutorial. European Journal of Operational Research **184**(2), 397 – 415 (2008)
5. Bennell, J.A., Oliveira, J.F.: A tutorial in irregular shape packing problems. Journal of the Operational Research Society (2009)
6. Carravilla, M.A., Ribeiro, C., Oliveira, J.F.: Solving nesting problems with non-convex polygons by constraint logic programming. International Transactions in Operational Research **10**(6), 651–663 (2003)
7. Cherri, L.H., Mundim, L.R., Andretta, M., Toledo, F.M.B., Oliveira, J.F., Carravilla, M.A.: Robust mixed-integer linear programming models for the irregular strip packing problem. European Journal of Operational Research (in press) pp. 1–14 (2016). DOI http://dx.doi.org/10.1016/j.ejor.2016.03.009
8. Dowsland, K.A., Dowsland, W.B., Bennell, J.A.: Jostling for position: Local improvement for irregular cutting patterns. The Journal of the Operational Research Society **49**(6), 647–658 (1998)
9. Dowsland, K.A., Vaid, S., Dowsland, W.B.: An algorithm for polygon placement using a bottom-left strategy. European Journal of Operational Research **141**(2), 371–381 (2002)
10. Egeblad, J., Nielsen, B.K., Odgaard, A.: Fast neighborhood search for two- and three-dimensional nesting problems. European Journal of Operational Research **183**(3), 1249–1266 (2007)
11. Elkeran, A.: A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. European Journal of Operational Research **231**, 757–769 (2013)
12. Fischetti, M., Luzzi, I.: Mixed-integer programming models for nesting problems. Journal of Heuristics **15**(3), 201–226 (2009)
13. Fowler, R.J., Paterson, M., Tanimoto, S.L.: Optimal packing and covering in the plane are np-complete. Information Processing Letters **12**(3), 133–137 (1981)
14. Fujita, K., Akagji, S., Kirokawa, N.: Hybrid approach for optimal nesting using a genetic algorithm and a local minimization algorithm. Advances in Design Automation; American Society of Mechanical Engineers **65**(1), 477–484 (1993)
15. Gomes, A.M., Oliveira, J.F.: Solving irregular strip packing problems by hybridising simulated annealing and linear programming. European Journal of Operational Research **171**(3), 811 – 829 (2006)
16. Hopper, E.: Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods. Ph.D. thesis, University of Wales, Cardiff (2000)
17. Imamichi, T., Yagiura, M., Nagamochi, H.: An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. Discrete Optimization **6**(4), 345 – 361 (2009)
18. Jakobs, S.: On genetic algorithms for the packing of polygons. European Journal of Operational Research **88**(1), 165–181 (1996)

19. Leung, S.C.H., Lin, Y., Zhang, D.: Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. Computers & Operations Research **39**(3), 678 – 686 (2012)
20. Leão, A., Toledo, F., Oliveira, J.F., Carravilla, M.A.: A semi-continuous MIP model for the irregular strip packing problem. International Journal of Production Research **54**, 712–721 (2016)
21. Oliveira, J.F., Ferreira, J.: Algorithms for nesting problems, applied simulated annealing. In Vidal, R.V.V. (Ed., Lecture Notes in Economics and Maths Systems) **396**(1), 255–274 (1993)
22. Oliveira, J.F., Gomes, A.M., Ferreira, J.S.: TOPOS - A new constructive algorithm for nesting problems. OR Spektrum **22**(2), 263–284 (2000)
23. Toledo, F.M.B., Carravilla, M.A., Ribeiro, C., Oliveira, J.F., Gomes, A.M.: The Dotted-Board Model: a new MIP model for nesting irregular shapes. International Journal of Production Economics **145**(2), 478 – 487 (2013)
24. Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K., Ibaraki, T.: Solving the irregular strip packing problem via guided local search for overlap minimization. International Transactions in Operational Research **16**(6), 661–683 (2009)
25. Wäscher, G., Haußner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research **183**(3), 1109 – 1130 (2007)