TITLE: The Dotted-Board Model: A new MIP model for nesting irregular shapes
AUTHORS: Franklina Maria Bragion Toledo; Maria Antónia Carravilla; Cristina Ribeiro; Jose Fernando Oliveira; Miguel M Gomes ;
SOURCE: INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS, VOLUME: 145, ISSUE: 2, PAGES: 478-487, PUBLISHED:2013
DOI: http://dx.doi.org/10.1016/j.ijpe.2013.04.009

The Dotted-Board Model: a new MIP model for nesting irregular shapes

Franklina M. B. Toledo[1] Maria Antónia Carravilla[2] Cristina Ribeiro[3]
José Fernando Oliveira[2] António Miguel Gomes[2]

fran@icmc.usp.br
{mac,mcr,jfo,agomes}@fe.up.pt

[1] Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
Caixa Postal 668, São Carlos, SP, CEP 13560-970, Brazil

[2] INESC TEC, Faculty of Engineering, University of Porto
Rua Dr. Roberto Frias S/N, 4200-465 PORTO, Portugal

[3] DEI-Faculdade de Engenharia da Universidade do Porto/ INESC TEC
Rua Dr. Roberto Frias S/N, 4200-465 PORTO, Portugal

Abstract

The nesting problem, also known as irregular packing problem, belongs to the generic class of cutting and packing (C&P) problems. It differs from other 2-D C&P problems in the irregular shape of the pieces. This paper proposes a new mixed-integer model in which binary decision variables are associated with each discrete point of the board (a dot) and with each piece type. It is much more flexible than previously proposed formulations and solves to optimality larger instances of the nesting problem, at the cost of having its precision dependent on board discretization. To date no results have been published concerning optimal solutions for nesting problems with more than 7 pieces. We ran computational experiments on 45 problem instances with the new model, solving to optimality 34 instances with a total number of pieces ranging from 16 to 56, depending on the number of piece types, grid resolution and the size of the board. A strong advantage of the model is its insensitivity to piece and board geometry, making it easy to extend to more complex problems such as non-convex boards, possibly with defects. Additionally, the number of binary variables does not depend on the total number of pieces but on the number of piece types, making the model particularly suitable for problems with few piece types. The discrete nature of the model requires a trade-off between grid resolution and problem size, as the number of binary variables grows with the square of the selected grid resolution and with board size.

Keywords: cutting and packing, nesting, irregular packing, mixed-integer programming models.

1 Introduction

1.1 The Nesting Problem

The nesting problem, also known as irregular packing problem, belongs to the more generic class of cutting and packing (C&P) problems. As in all C&P problems, one or more big objects have to be

divided into smaller items or pieces so that (1) unused regions of the big object, usually designated as waste, are minimized — input minimization problems, according to Wäscher typology, or (2) the value of the pieces cut from the big object is maximized — output maximization problem, according to the same typology [Wäscher et al., 2007]. The problem addressed in this paper is a two-dimensional (2D) C&P problem, i.e. only two of the dimensions of the objects are relevant, as the third dimension is equal for all objects and pieces. Moreover, the goal is to minimize the used length of a big rectangular object (the board), of unbound length, making it a 2D Open Dimension Problem. It is often called irregular strip packing.

The nesting problem differs from other 2D C&P problems in the irregular shape of the pieces. Irregular shapes are those that involve non-trivial handling of the geometry [Bennell and Oliveira, 2008]. This leaves out of our scope rectangles and circles, in which the evaluation of the distance between two pieces can be performed by simple coordinate comparison. The most common representation, in the literature, for the irregular shape of a piece is the polygon, and although geometric operations with polygons are not trivial, we will also assume that all pieces are represented by polygons. No additional constraints are imposed on the polygon characteristics, so we deal with the harder to tackle class of non-convex polygons.

Nesting problems are not only a scientific challenge, as they add the geometry handling complexity to the combinatorial optimization nature of C&P problems, but they are also most relevant in real-world applications. Indeed they arise in a wide variety of industrial production processes, including textile, garment, metalware, furniture and shoe industries. In all cases saving raw-materials represents an important contribution to the economic and environmental performance of companies.

1.2 State of the Art on the Resolution of Nesting Problems

Nesting problems have been addressed by many authors. However, given the problem complexity, to the best of the authors' knowledge there is no published exact mathematical programming based method to solve it, reaching guaranteed optimal solutions for the problem. Although a thorough tutorial on the nesting problem resolution is available [Bennell and Oliveira, 2009], the approaches that more closely relate to the work reported in this paper will be briefly highlighted.

Not surprisingly, heuristic and metaheuristic algorithms are the basis of the overwhelming majority of the published approaches. Among these, the most recent approaches with best performance are from Gomes and Oliveira, Egeblad et al, Imamichi et al, Bennell and Song and Leung et al, [Gomes and Oliveira, 2006, Egeblad et al., 2007, Imamichi et al., 2009, Bennell and Song, 2010, Leung et al., 2012]. A pioneering work on the exact resolution of non-convex nesting problems [Carravilla et al., 2003] used constraint logic programming as resolution method and has solved to optimality problems with up to 7 pieces. All the other approaches deal with approximations or simpler forms of the nesting problem.

The first approximation is geometric and consists in considering only rectangular shapes. The first exact solution procedure for the constrained (i.e. with an upper bound on the number of pieces of each type to cut or pack) 2D non-guillotine rectangular problem was proposed by John Beasley [Beasley, 1985] and was based on a binary integer programming formulation. The author assumes that all rectangle's lengths and widths are integers, leading to a discrete model, as pieces can only be placed on a set of discrete points of the board. More recent work on exact approaches to the rectangular strip packing problem (e.g. [Martello et al., 2003, Alvarez-Valdes et al., 2009]) resort to branch-and-bound algorithms but do not build on mathematical programming models. An approach based on the resolution of a mixed-integer linear programming model was recently

proposed. One of the dimensions of the strip is discretized and the other one is kept continuous, in what constitutes a semi-discrete model [Castro and Oliveira, 2011].

Mathematical programming models have been used in the resolution of nesting problems, but in the context of solving simpler sub-problems. Particularly relevant is the compaction problem in which, given a feasible solution of the nesting problem, a (non-integer) linear programming model is used to locally optimize the solution [Li and Milenkovic, 1995]. A set of coordinated continuous motions is applied to the pieces, aiming to minimize the board's length that is effectively used to pack them. While the relative positions of all pairs of pieces are kept, the absolute position of each piece may become substantially different. The non-integer property of the model allows the continuous motion of each piece. The compaction problem has been quite successfully hybridized with other resolution methods, such as simulated annealing [Gomes and Oliveira, 2006] or tabu search [Bennell and Dowsland, 2001], which take care of the generation of alternative initial solutions for the compaction model. It has also been used on its own, as a multi-start algorithm (e.g. [Stoyan et al., 1996]). It should be noticed that this compaction model has the potential to be extended to address the complete nesting problem, with the inclusion of binary variables. However, given the very small size of the instances that can be solved with it, it has never been proposed in the literature for that purpose.

Until now two research directions have been proposed concerning mathematical programming models for nesting problems. The first one was suggested by Beasley as an extension of his binary integer programming formulation for rectangles [Beasley, 1985]. However, as the extension of the non-overlapping constraints is not straightforward, the suggestion was never followed. This would be a discrete model, with each decision variable standing for a given piece being placed or not in a given (x, y) positioning point. The other modeling approach uses continuous decision variables (x, y) that are the placement coordinates of each piece, but the non-overlapping constraints require the use of auxiliary binary variables [Carravilla and Ribeiro, 2005, Gomes and Oliveira, 2006]. This is a continuous model which reduces to the previously discussed compaction model when the binary variables are fixed. This paper proposes a new mixed-integer discrete model in which binary decision variables are associated to each discrete point of the board (a dot) and each piece type; the variable is equal to one in case a piece of the corresponding type is positioned at that dot. This formulation is much more flexible than others previously proposed in the literature and allows solving to optimality larger instances of the nesting problem, at the cost of having its precision dependent on the board discretization step. It should be pointed out that although optimal solutions for this model, associated with a given discretization step, are achieved, these may not be optimal solutions for the original nesting problem when defined over continuous domains.

1.3 Organization of the paper

The paper is organized as follows. In Section 2 we present the data required to define an instance of the nesting problem, its geometric features and the required data pre-processing. In Section 3 the *Dotted-Board Model*, the new mathematical programming formulation for the nesting problem is presented. Section 4 describes the valid inequalities, the lower bounds and the initial solutions used to improve the resolution of the *Dotted-Board Model*. The computational experiments designed to test the model and the discussion of their results are presented in Section 5. In the last section, we draw some conclusions both the research lines opened by the new model and its limitations. Section 6 also includes several pointers to future research on this subject.

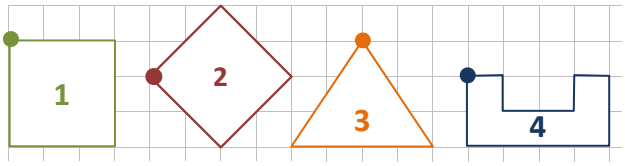

Figure 1: Piece types of the 4-piece instance.

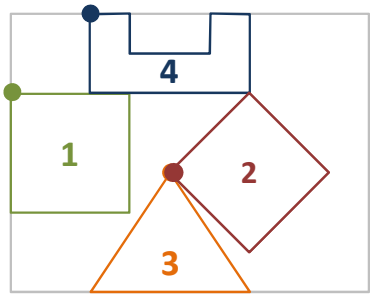

Figure 2: A feasible solution for the 4-piece instance.

2 Problem Representation and Geometric Features

To solve the nesting problems it is necessary to deal with several geometrical constraints involving the pieces and the board. In order to illustrate the representation of an instance of the nesting problem and the data pre-processing required to build the problem constraints, we will base our description, on a 4-piece instance (Figure 1) where piece types 1, 2 and 3 are convex and piece type 4 is non-convex. The instance has just one piece of each type. The small circle over one vertex of a piece type highlights the piece's reference point. The board is rectangular with a width of 7 and, given the pieces to be positioned, an upper bound on the length is 9. A feasible solution for the 4-piece instance is illustrated in Figure 2. A solution is completely described by the positions of the reference points of the pieces on the board.

2.1 Representation of the board

The board is rectangular, has a width W and, considering the set of pieces that must be positioned, it is possible to compute an upper bound L on the length. By placing a grid of appropriate resolution over the board, it can be represented by a set of dots (Figure 3).

- W – board width;
- L – upper bound on the board length;
- g_x – grid resolution in the x-dimension;
- g_y – grid resolution in the y-dimension;
- $C = \left\lfloor \frac{L}{g_x} \right\rfloor + 1$ – number of columns;

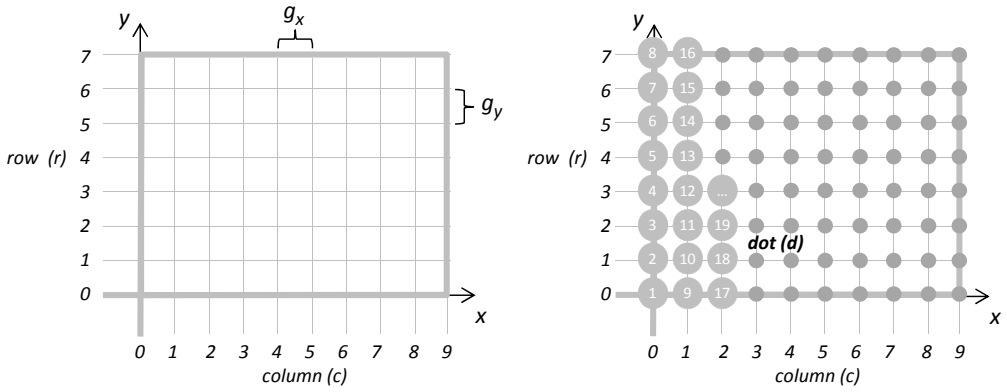

Figure 3: A dotted board.

- $R = \left\lfloor \frac{W}{g_y} \right\rfloor + 1$ – number of rows;
- $D = C \times R$ – total number of board dots.

2.2 Indices

- $t, u \in \mathcal{T}$; $\mathcal{T} = \{1, \ldots, T\}$ – t, u are piece types;
- $c \in \mathcal{C}$; $\mathcal{C} = \{0, \ldots, C-1\}$ – c is a board column;
- $r \in \mathcal{R}$; $\mathcal{R} = \{0, \ldots, R-1\}$ – r is a board row;
- $d \in \mathcal{D}$; $\mathcal{D} = \{1, \ldots, D\}$ – d is a board dot. A board dot can also be represented by a pair of values $(c, r) \in \mathcal{C} \times \mathcal{R}$, in which $c = \left\lfloor \frac{d}{R} \right\rfloor$ and $r = d - \left\lfloor \frac{d}{R} \right\rfloor \times R - 1$. Reciprocally, $d = c \times R + r + 1$.

2.3 Representation of the piece types

Each piece type t is represented by one polygon, defined by a set of vertices whose coordinates are relative to a reference point.

- $(0, 0)$ – reference point of piece type t ($t \in \mathcal{T}$);
- $\langle (x_t^1, y_t^1), (x_t^2, y_t^2), \ldots, (x_t^{v_t}, y_t^{v_t}) \rangle$ – list of vertices of piece type t, given in the clockwise sense ($t \in \mathcal{T}$);
- $x_t^m, x_t^M, y_t^m, y_t^M$ – limits of the rectangular envelope of piece type t, ($t \in \mathcal{T}$), where:
  $x_t^m = \min_{i \in \{1,\ldots,v_t\}} \{x_t^i\}$; $x_t^M = \max_{i \in \{1,\ldots,v_t\}} \{x_t^i\}$; $y_t^m = \min_{i \in \{1,\ldots,v_t\}} \{y_t^i\}$; $y_t^M = \max_{i \in \{1,\ldots,v_t\}} \{y_t^i\}$;
- q_t – number of pieces of type t that have to be positioned ($t \in \mathcal{T}$).

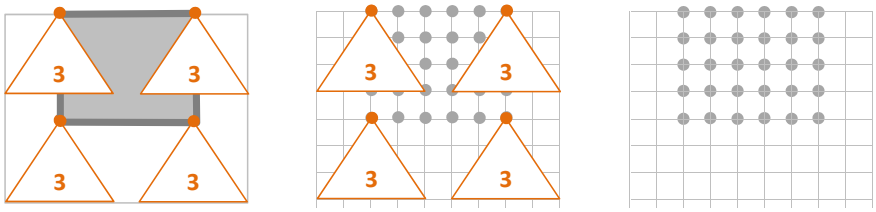

Figure 4: IFP_3 and $\mathcal{IFP}_3$ between piece type 3 and a rectangular board ($W = 7$ and $L = 9$).

2.4 Relation between the board and the piece types: the inner-fit polygon

A geometric constraint of the nesting problem is that each piece must be totally placed inside the board. To guarantee that the solutions verify these constraints, we used the inner-fit polygon concept (*IFP*) [Gomes and Oliveira, 2002]. Given a piece type t and a board, the inner-fit polygon of a piece type t relative to the board is the locus of all the admissible positioning points for the reference point of piece type t, those that guarantee that the piece is totally over the board. When the board is rectangular, the *IFP* is also a rectangle.

- IFP_t – inner-fit polygon between piece type t and the board ($t \in \mathcal{T}$).

If the board is represented by a set of $\mathcal{D}$ dots, the inner-fit polygon of a piece type t relative to the board is discrete and determines the set of dots where the reference point of the piece type t can be positioned, such that the piece is totally over the board.

- $\mathcal{IFP}_t$ – is the maximum subset of $\mathcal{D}$ such that a piece type t having its reference point at $d \in \mathcal{IFP}_t$ is totally over the board.

In Figure 4, the grey rectangle at the top is IFP_3, the locus of all the admissible positioning points for the reference point of piece type 3, and the set of dots is $\mathcal{IFP}_3$.

2.5 Relation between pairs of piece types: the nofit polygon

Another important type of constraint guarantees that the pieces do not overlap. This is enforced for each pair of pieces by the nofit polygon concept (*NFP*) (e.g. [Bennell and Dowsland, 2001, Bennell and Oliveira, 2008]).

- $NFP_{t,u}$ – nofit polygon between piece type t and piece type u ($t, u \in \mathcal{T}$), is the locus of the points such that, if the reference point of piece type u is inside $NFP_{t,u}$ the pieces overlap, if the reference point of piece type u is over the edges of $NFP_{t,u}$ the pieces touch each other, and if the reference point of piece type u is outside $NFP_{t,u}$, the pieces do not touch.

$NFP_{3,4}$, the *NFP* between piece 3 and piece 4 is the white area with a thick grey contour in Figure 5.

- $\mathcal{NFP}^d_{t,u}$ – if the board is represented by a set of $\mathcal{D}$ dots, $\mathcal{NFP}^d_{t,u}$ is the maximum subset of $\mathcal{D}$ such that when the reference point of piece type t is positioned at dot d, if piece type u has its reference point at any point of $\mathcal{NFP}^d_{t,u}$, then pieces u and t overlap ($t, u \in \mathcal{T}$; $d \in \mathcal{IFP}_t$).

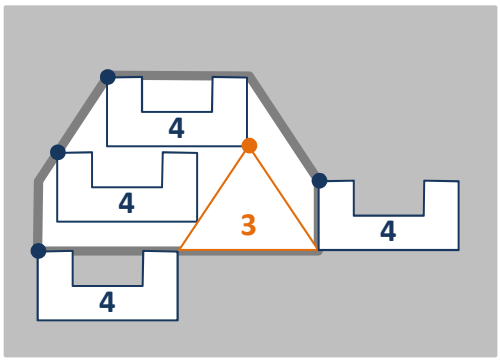

Figure 5: The nofit polygon between piece type 3 and piece type 4, $NFP_{3,4}$.

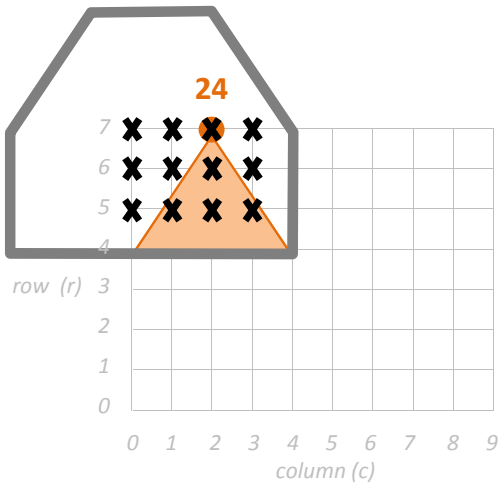

Figure 6: Set of dots in $\mathcal{NFP}_{3,1}^{24}$.

The set of dots of the $\mathcal{NFP}_{3,1}^{24}$ is marked with an **X** in Figure 6.

Both the nofit polygon and the inner-fit polygon are very powerful tools to deal with the geometric constraints of the nesting problem. Their main advantage is the possibility of precalculation, becoming a fixed cost in terms of computational times. However, computing these polygons is not straightforward and requires complex algorithms, mainly when dealing with non-convex pieces. The efficiency of all known algorithms decreases with the number of edges in each piece and with the number of concave components. One of the best approaches, which includes the capacity to deal with unconnected nofit polygons (nofit polygons composed by two or more simple polygons) was published by [Bennell and Song, 2008] and is based on the mathematical concept of Minkowski sum. The computational time required to compute the nofit polygons is therefore very dependent on the complexity of the geometry of the pieces, but in the computational experiments presented in this paper, for a set of 7 pieces (4 convex and 3 non-convex) with between 3 and 8 edges, computing 28 nofit polygons takes 0.057 seconds, i.e. a neglectable time when compared against the overall running times (section 5). However, for more complex geometries these times may increase by several orders of magnitude, which may be relevant if very fast algorithms are sought.

3 Dotted-Board Model

In this section, we propose a new mathematical programming model for the nesting problem. As in many other approaches proposed in the literature, we will assume that piece rotation is not allowed. The idea is to represent the board by a set of dots defined by placing a grid of appropriate resolution over the board (see Figure 3) and by considering the dots, instead of the coordinates of the positioning points of the pieces, as the decision variables of the problem. Henceforth, our model is called the *Dotted-Board Model*.

After defining the resolution of the grid, a binary decision variable δ_t^d is defined for each pair (dot, piece type) such that $d \in \mathcal{IFP}_t$.

$$\delta_t^d = \begin{cases} 1 & \text{if the reference point of a piece of type } t \text{ is positioned on dot } d; \\ 0 & \text{otherwise.} \end{cases}$$

Figure 7 illustrates a feasible solution for the 4-piece instance using a rectangular board with a grid with 10 columns ($C = 10$) and 8 rows ($R = 8$), is illustrated. In the example, the reference point of piece 1 is placed on dot $d = 6$, reference points of pieces 2 and 3 are placed on dot $d = 44$ and reference point of piece 4 is placed on dot $d = 24$. It is worth noting that the reference points of two pieces of different types can be positioned on the same dot (see pieces 2 and 3) if they do not overlap.

One of the major advantages of the *Dotted-Board Model* is that convex and non-convex pieces can be dealt with in the same way. In Figure 7 pieces 1, 2 and 3 are convex and piece 4 is non-convex.

The number of decision variables of the *Dotted-Board Model* is proportional to the number of piece types times the product of the dimensions of the board ($C \times R$); if we consider a fixed width, the length is expected to vary linearly with the number of pieces (you need a longer board for more pieces).

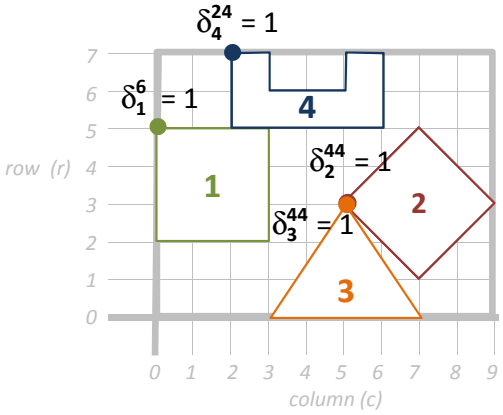

Figure 7: A feasible solution for the 4-piece example.

3.1 Model Constraints

The number of binary variables (δ_t^d) of the model is equal to $T \times D$, where T is the number of piece types and D is the number of board dots according to the grid resolution. For the 4-piece instance in Figure 7, $T = 4$ and $D = 10 \times 8 = 80$, therefore the number of binary variables is equal to $4 \times 80 = 320$.

As detailed above, the nesting problem has three basic sets of constraints:

1. Each piece needs to be positioned entirely inside the board;
2. All the pieces need to be positioned;
3. The pieces may not overlap.

The first type of constraints are enforced by defining the variables δ_t^d only for $d \in \mathcal{IFP}_t$.

As for the second type of constraints, if q_t is the number of pieces of type t that need to be positioned, the following constraints ensure that all the pieces are positioned over the board:

$$\sum_{d \in \mathcal{IFP}_t} \delta_t^d = q_t \quad \forall t \in \mathcal{T}.$$

The constraints that guarantee that each pair of piece types does not overlap are based on the nofit polygon concept, described in detail in Section 2, and can be written as:

$$\delta_u^e + \delta_t^d \leq 1 \quad \forall e \in \mathcal{NFP}_{t,u}^d, \forall t, u \in \mathcal{T}, \forall d \in \mathcal{IFP}_t.$$

3.2 Objective Function

A feasible solution for the nesting problem is obtained by placing all the pieces on the rectangular board. The objective is to find a feasible solution that minimizes the used length of the board.

The solution's length is given by the x-coordinate of the rightmost vertex of all positioned pieces. This represents the minimum length that the board has to have to allow cutting all pieces placed in that solution. As seen in Section 2, each piece is represented by a polygon with a reference point. Considering the dot where the piece was positioned, the minimum length of the board needed for the piece can be obtained. For example, in Figure 7, for piece 1, whose reference point is placed on dot $d = 6$, the minimum length of the board is 3. For pieces 2, 3 and 4, the minimum length of the board is respectively 9, 7 and 6. Therefore, the minimum length of the board for this solution is equal to $9 = \max\{3, 9, 7, 6\}$.

For each piece type t, the horizontal distance from its reference point to the end of the piece is given by x_t^M. Considering that the grid resolution in x is g_x, and that the piece is positioned in dot (c, r), the board length needed for piece type t is $c \times g_x + x_t^M$. For example, considering the layout represented in Figure 7, piece 2, which has an x_t^M of 4 and is placed on dot (5,3), will define a board length of $5 \times 1 + 4 = 9$, and this is the maximum value for this solution.

The minimum length of the board for the problem must be sufficient to include all the pieces and is given by:

$$minimize \max\{(c \times g_x + x_t^M) \times \delta_t^d\} \quad \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T}.$$

3.3 The complete model

Using the constraints and the objective function defined above, the *Dotted-Board Model* of the nesting problem can be written as:

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

Subject to:

$$(c \times g_x + x_t^M) \times \delta_t^d \leq \quad z \quad \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T}; \tag{2}$$

$$\sum_{d \in \mathcal{IFP}_t} \delta_t^d = \quad q_t \quad \forall t \in \mathcal{T}; \tag{3}$$

$$\delta_u^e + \delta_t^d \leq \quad 1 \quad \forall e \in \mathcal{NFP}_{t,u}^d, \forall t, u \in \mathcal{T}, \forall d \in \mathcal{IFP}_t; \tag{4}$$

$$\delta_t^d \in \quad \{0, 1\} \quad \forall d \in \mathcal{IFP}_t, \forall t \in \mathcal{T}; \tag{5}$$

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

The objective function (1) minimizes the length of the board. The board length z is defined as shown in constraints (2). Constraints (3) guarantee that all the pieces are positioned. Constraints (4) guarantee that the pieces do not overlap. Constraints (5) define the decision variables only for positions where the pieces are inside the board. The domain of variable z is defined by constraint (6).

4 Improving the Dotted-Board Model

4.1 Valid Inequalities

Two types of valid inequalities were added to the model. One of them concerns the number of reference points of piece types that can be positioned on the same dot. The other type of valid

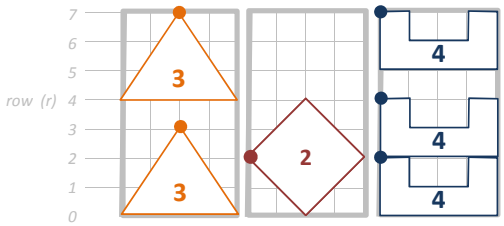

Figure 8: Maximum number of pieces of the same type that can be positioned in one column.

inequality is obtained by computing the maximum number of pieces of the same type that can fit along the width of the board.

In the *Dotted-Board Model* it is possible to have more than one reference point in one dot. Given an instance it is however possible to verify if there are pieces that cannot be positioned on the same dot. If piece types t and u, $t \neq u$, cannot be positioned on the same dot, then they belong to the set of incompatible pairs of piece types $\mathcal{I}$, and therefore:

$$\delta_t^d + \delta_u^d \leq 1 \quad \forall t, u \in \mathcal{T} : (t, u) \in \mathcal{I}; t \neq u; \forall d \in \mathcal{D}.$$

The other type of valid inequality relates one piece type and the width of the board. Given an instance with a board of width W and given the piece types that have to be positioned, it is possible to calculate, for each piece type, the maximum number of pieces of that type that can be positioned on dots of the same column (Figure 8):

$$\sum_{r \in \mathcal{R}} \delta_t^d \leq \left\lfloor \frac{W}{y_t^M - y_t^m} \right\rfloor \quad \forall t \in \mathcal{T}; \forall c \in \mathcal{C}; d = c \times R + r + 1.$$

4.2 Lower Bounds

Good lower bounds are very important tools to efficiently solve integer programming problems, especially when considering branch and bound methods. The *Dotted-Board Model* is basically defined by binary variables. Therefore, as expected, the value of the lower bound obtained by the solution of its linear relaxation is a weak bound. One simple lower bound for the nesting problem is based on the area of the pieces. This lower bound is obtained by the sum of area of the pieces divided by the width of the board. Another lower bound is based on the length of the longest piece. We compute both lower bounds and use the maximum value to incorporate in the model, improving the quality of the gaps.

4.3 Upper Bounds

For the *Dotted-Board Model*, the number of binary variables is quadratic on the dimensions of the board. The board width is known, but its length needs to be determined. Therefore, it is very important to obtain a suitable estimate for the length of the board. To this end, for each instance we first run the model for a limited amount of time, with a board length long enough to contain all

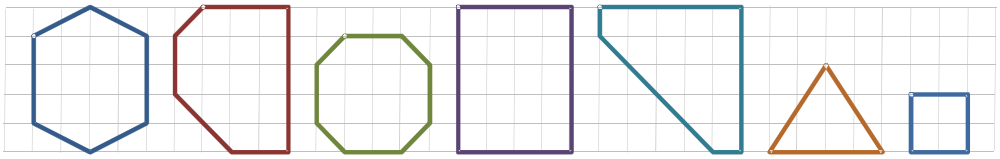

Figure 9: Piece types of the RCO instances.

pieces. If a feasible solution is generated we have an upper bound for the board length. Afterwards, the model is rerun but now with a smaller number of variables as the board length has already been reduced.

5 Computational experiments

In this section, we present the instances used and the results obtained in the computational experiments with the *Dotted-Board Model*. The instances were solved by the mixed-integer programming solver CPLEX 12.3. We used the default parameters of CPLEX except for the MIP tolerance that was set to 0.9 as the resolution of the grid is always 1 unit. Experiments were carried out on a HP Z800 workstation with two six-core Intel Xeon X5690 at 3.47 GHz with 48 GB RAM, running Linux. In all experiments, a time limit of 18 000 seconds was considered.

The computational experiments involved a total of six groups of instances, taken from the literature, that are fully described in section 5.1. For these instances we were able to solve until optimality problems with a total number of pieces ranging from 16 to 56, depending on the number of piece types, on the grid resolution and on the size of the board.

5.1 Instances

Six groups of instances were used to evaluate the performance of the model. Two groups of instances are closely related to each other, one called BLAZEWICZ, with non-convex pieces, and another one, RCO, where the pieces are the convex hull of the BLAZEWICZ pieces. Three other groups consist of different subsets of BLAZEWICZ's piece types.

The RCO instance group [Ribeiro et al., 1999] is based on a set of seven convex piece types, as represented in Figure 9. Within this group, the instances differ on the number of pieces of each piece type. RCOn stands for an instance where n pieces of each one of the 7 piece types have to be placed on the board, with n ranging from 1 to 5. The rectangular board has a width $W = 15$.

The BLAZEWICZ instance group [Oliveira et al., 2000] is composed by a set of seven piece types, as represented in Figure 10. Once again, the instances differ on the number of pieces of each piece type, and BLAZEWICZn stands for an instance where n pieces of each one of the 7 piece types have to be placed on the board, with n ranging from 1 to 5. The instance that usually appears in the literature is the one corresponding to $n = 4$. The rectangular board has a width $W = 15$.

The SHAPES instance group [Oliveira et al., 2000] is based on a set of four piece types, as represented in Figure 11. The 6 instances of this group vary on the number of pieces of each one of the 4 piece types, as shown on Table 1. These figures were chosen so that the total number of pieces was similar to the BLAZEWICZ and RCO groups, allowing us to draw some conclusions on

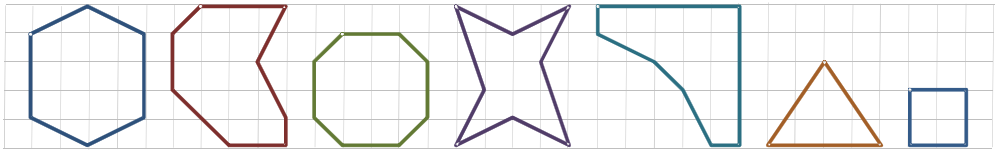

Figure 10: Piece types of the BLAZEWICZ instances.

Table 1: Number of pieces of each piece type for the SHAPESn group of instances.

Instance	Piece type 1	Piece type 2	Piece type 3	Piece type 4	Total number of pieces
SHAPES2	2	2	2	2	8
SHAPES4	4	4	4	4	16
SHAPES5	5	5	5	5	20
SHAPES7	7	7	7	7	28
SHAPES9	9	7	9	9	34
SHAPES15	15	7	9	12	43

the impact of characteristics, other than the piece number, on the model's behavior. The instance that usually appears in the literature is SHAPES15. The rectangular board has a width $W = 40$.

The other three instance groups are based on subsets of BLAZEWICZ's piece types. They were designed to analise the variations of the model with instance characteristics and model parameterization.

Group BLAZP2_n considers only piece type 2 of BLAZEWICZ, with number of pieces n, $n \in \{7, 14, 21, 28, 35\}$. The value of n was chosen so that the total number of pieces remains constant when compared with the BLAZEWICZ group. Group BLAZP4_n is similar to BLAZP2_n but it uses only piece type 4 from BLAZEWICZ. Finally, BLAZP2P4_m_n includes piece types 2 and 4, with

$$(m, n) \in \{(4, 3); (7, 7); (11, 10); (14, 14); (18, 17); (21, 21); (28, 28); (35, 35)\}$$

pieces of each type. This group includes the instances with the largest number of pieces and was designed to test the limitations of the model. For all these groups the rectangular board has a width $W = 15$ and a length L, large enough to fit all the pieces of the largest instance. Therefore, for these groups of instances, no upper bound different from the board length was used: $L = 60$

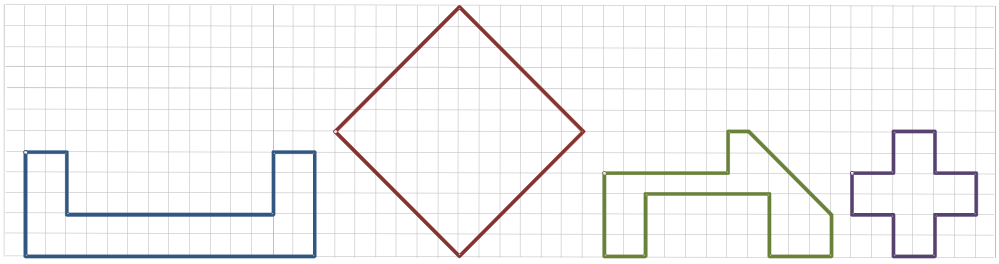

Figure 11: Piece types of the SHAPES instances.

Table 2: Computational results of the first phase of the computational experiments.

Instance	Total number of pieces	Number of dots	Number of binary variables	Upper Bound	Objective function	Gap (%)	Solution Time (sec.)
RCO1	7	144	1008	8	8	0	0.62
RCO2	14	256	1792	15	15	0	6.28
RCO3	21	368	2576	22	22	0	2393.42
RCO4	28	480	3360	29	29	3.5	TL
RCO5	35	608	4256	37	37	8.1	TL
BLAZEWICZ1	7	144	1008	8	8	0	0.69
BLAZEWICZ2	14	240	1680	15	14	0	15.98
BLAZEWICZ3	21	352	2464	22	20	0	5583.82
BLAZEWICZ4	28	480	3360	28	28	10.7	TL
BLAZEWICZ5	35	640	4480	37	35	14.3	TL
SHAPES2	8	615	2460	14	14	0	0.45
SHAPES4	16	1148	4592	27	25	0	17951.33
SHAPES5	20	1353	5412	32	30	13.3	TL
SHAPES7	28	2009	8036	48	45	39.4	TL
SHAPES9	34	2255	9020	54	54	40.4	TL
SHAPES15	43	2788	11152	67	67	40.8	TL

for all instances, except for instances BLAZP2P4_28_28 and BLAZP2P4_35_35 for which a length $L = 100$ was used was used, as a board of 15×60 would be too small to place all the pieces.

5.2 Computational Results – Phase 1

In the first phase of the computational experiments the 16 instances of groups RCOn, BLAZEWICZn and SHAPESn were solved, and the results are presented on Table 2. The instance name and the total number of pieces are in the first two columns. The third and fourth columns show the size of the model in terms of number of dots and number of binary variables. The values in the fifth column are the upper bounds as described in section 4.3. The remainder of the table concerns the results of each run. The value of the objective function is given in the sixth column, while in the seventh column the optimality gap is given, which is meaningful when the time limit is reached. That can be be seen in the eighth column where the execution time is presented (TL stands for time limit).

We were able to solve until proven optimality 8 instances: 3 from the RCO group, with a maximum of 21 pieces, 3 from the BLAZEWICZ group, also with a maximum of 21 pieces, and 2 from the SHAPES group, with a maximum of 16 pieces. For the other instances the resolution stopped after reaching the time limit. In those cases the optimality gap ranges from 3.5% to 40.8%. The optimal layouts for instances RCO3, BLAZEWICZ3 and SHAPES4 are presented in Figures 12, 13 and 14.

Within the same group of instances, the model's performance degrades as the number of pieces increases. Although the number of variables does not directly depend on the total number of pieces, more pieces need a larger board, increasing the number of dots and consequently the number of binary variables. For that same reason the SHAPES group was also more difficult to solve than

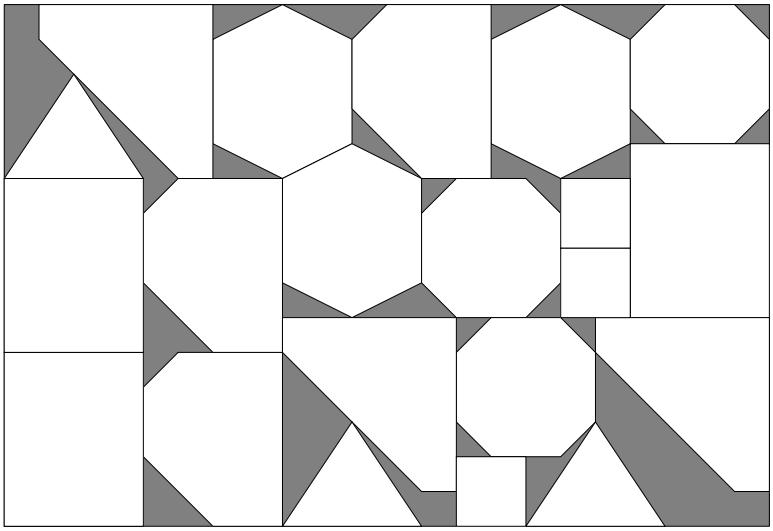

Figure 12: Optimal layout for instance RCO3.

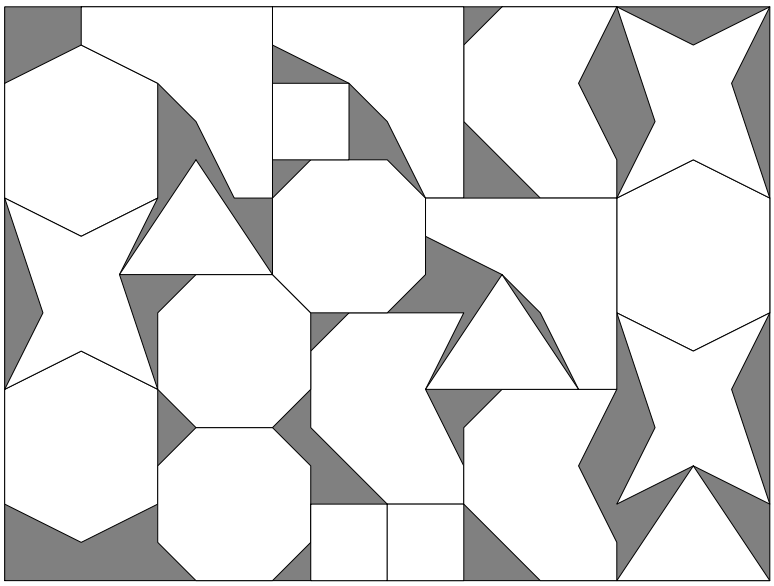

Figure 13: Optimal layout for instance BLAZEWICZ3.

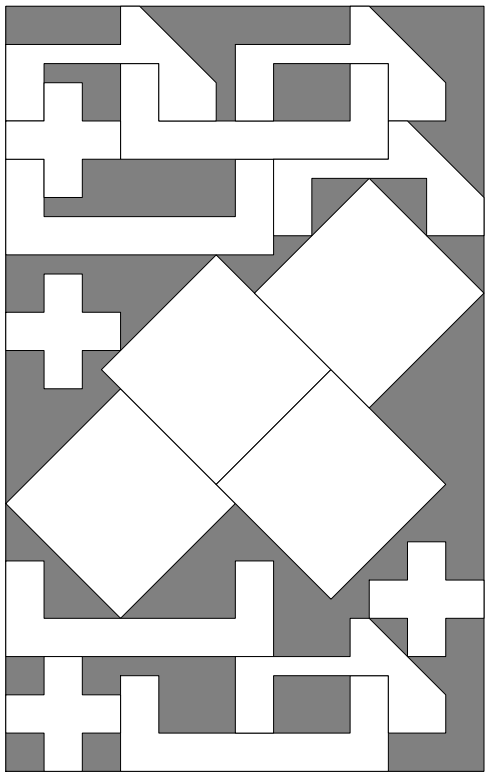

Figure 14: Optimal layout for instance SHAPES4.

Table 3: Computational results of the second phase of the computational experiments.

Instance	Total number of pieces	Upper Bound	Objective function	Gap (%)	Solution Time (sec.)
BLAZP2_7	7	60	12	0	0.19
BLAZP2_14	14	60	20	0	0.17
BLAZP2_21	21	60	28	0	0.16
BLAZP2_28	28	60	40	0	0.16
BLAZP2_35	35	60	48	0	0.15
BLAZP4_7	7	60	10	0	0.26
BLAZP4_14	14	60	19	0	0.44
BLAZP4_21	21	60	28	0	1.76
BLAZP4_28	28	60	37	0	7.90
BLAZP4_35	35	60	45	0	8.33
BLAZP2P4_4_3	7	60	11	0	0.84
BLAZP2P4_7_7	14	60	19	0	1.56
BLAZP2P4_11_10	21	60	28	0	2.97
BLAZP2P4_14_14	28	60	38	0	27.21
BLAZP2P4_18_17	35	60	47	0	100.36
BLAZP2P4_21_21	42	60	56	0	489.43
BLAZP2P4_28_28	56	100	74	0	6525.54
BLAZP2P4_35_35	70	100	93	7.5	TL

the other two groups. Although the number of piece types is smaller (4 against 7), which leads to less binary variables, the size of the pieces is larger and therefore more dots are used to discretize the board and more binary variables induced. The latter effect clearly dominates, and the overall result is that this group of instances is much harder to solve.

An important feature of the *Dotted-Board Model*, which we have anticipated in Section 3 and that is highlighted with the computational results, is that the lack of convexity has no impact on the model's performance. Groups RCO and BLAZEWICZ differ only on the pieces' convexity and the similarities on their performance support this conclusion.

5.3 Computational Results – Phase 2

In the previous section we claimed that the model's performance was not dependent on the total number of pieces (for the same board size), but only on the number of piece types. To verify this claim and to test the model to its limits, the instances of groups BLAZP2_n, BLAZP4_n and BLAZP2P4_m_n were run. In this second set of experiments the upper bound L was arbitrarily large. The results of this second phase of tests are presented on Table 3.

These results show that when considering just 1 or 2 piece types we are able to solve problems up to 56 pieces, while for the same kind of problem, BLAZEWICZ groups of instances with 7 piece types, the largest problem solved to optimality had 21 pieces. The instance with 70 pieces of two different types was not solved to optimality within the time limit and its resolution ended with a gap of 7.5%. By comparing the results of group BLAZP2_n (a group with an extremely good performance) or group BLAZP4_n (which has an intermediate performance) against BLAZP2P4_n_m

Table 4: Computational results with a refined grid.

Instance	Total number of pieces	Upper Bound	Objective function	Gap (%)	Solution Time (sec.)
BLAZP2_7	7	120	11.0	0	49.06
BLAZP2_14	14	120	18.0	0	13.18
BLAZP2_21	21	120	25.0	0	19.87
BLAZP2_28	28	120	35.5	0	499.76
BLAZP2_35	35	120	42.5	0	74.58
BLAZP4_7	7	120	10.0	0	46.05
BLAZP4_14	14	120	18.5	0	375.66
BLAZP4_21	21	120	28.0	19.64	18017.18
BLAZP2P4_4_3	7	120	10.5	0	129.48
BLAZP2P4_7_7	14	120	18.0	0	414.62
BLAZP2P4_11_10	21	120	27.0	14.81	18004.38

it can also be noticed that for the same total number of pieces, doubling the number of piece types multiplies the resolution time by a factor that varies from around 4 to 670.

Finally, the performance of the *Dotted-Board Model* is clearly dependent on the grid resolution. To experimentally verify this dependence an additional set of tests was run considering again the group of instances BLAZP2_n, BLAZP4_n and BLAZP2P4_n_m, and refining the grid by doubling its resolution. This corresponds to approximately 4 times the previous number of dots and, consequently, 4 times the number of binary variables.

With this refined grid we were only able to solve until optimality all the instances of group BLAZP2_n, the instances BLAZP4_7 and BLAZP4_14 and the instances BLAZP2P4_4_3 and BLAZP2P4_7_7. These results are in Table 4 and the optimal layouts for instance BLAZP4_14, with the initial grid and the refined grid, are presented in Figure 15.

To further explore the impact of grid resolution on the overall performance of the model, and to give an indication of the balance between solution quality and computation time that must be made when using this method, the 8 instances that were solved to optimality with both grid resolutions, were run again but now halving the resolution. For each instance, the values for the objective function and the solution time were normalized, dividing each by the maximum value when considering the 3 grid resolutions. The average is plotted on Figure 16. It becomes clear the need for adequate choice of the grid resolution in order to get good quality solutions within reasonable computational times.

6 Conclusions and further work

In this paper we present a new discrete model for the nesting problem, the *Dotted-Board Model*. This model builds on the discretization of the board, generating a grid of dots that become the feasible positioning points for the pieces. A binary decision variable is associated to each 'dot'/'piece type' pair. To guarantee the geometric feasibility of the layouts—no overlap among pieces and containment inside the board, we resort to the nofit polygon and inner-fit polygon concepts. So far there are no results published in the literature concerning optimal solutions for nesting problems

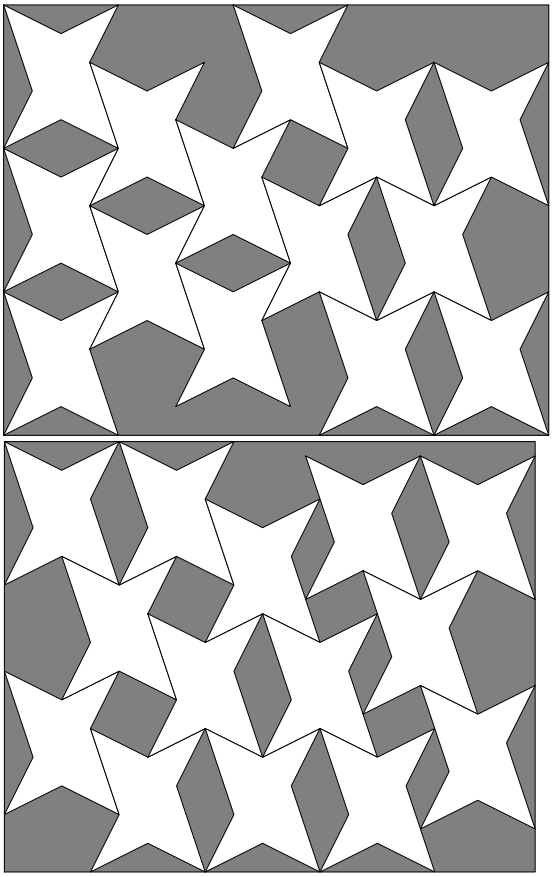

Figure 15: Optimal layouts for the instance BLAZP4_14, with the initial grid and the refined grid.

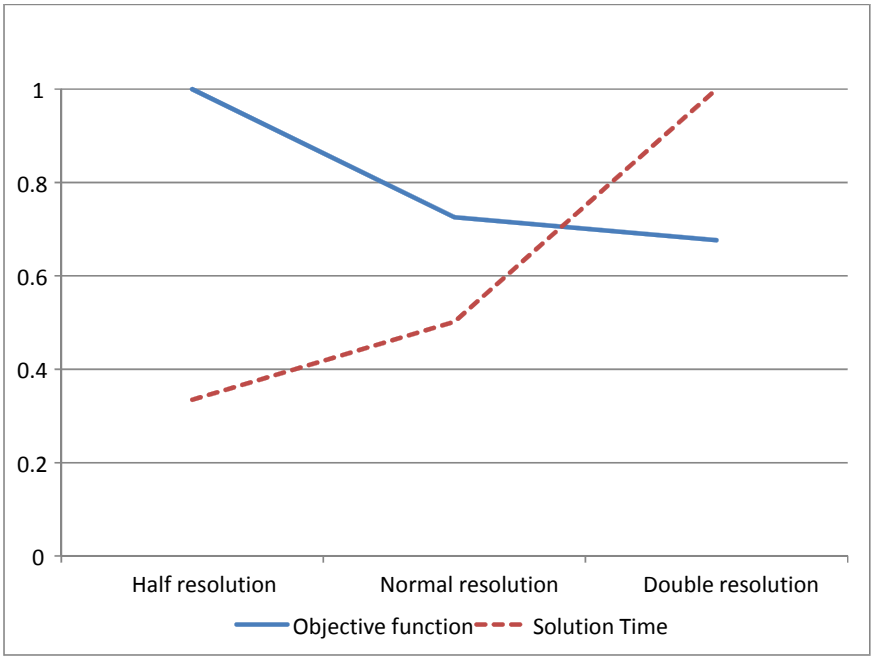

Figure 16: Evolution of the objective function and solution time values with grid resolution.

with more then 7 pieces. With this model we were able to solve until optimality problems with a total number of pieces ranging from 16 to 56, depending on the number of piece types, on the grid resolution and on the size of the board.

The strong points of this model are, first and foremost, its robustness with respect to the pieces and board geometries. From a conceptual analysis of the model it is clear that convexity does not play a role here, and the computational experiments support this conclusion. The model can, therefore, be quite easily applied to geometrically more complex instances such as non-convex boards, with holes or other defects, as represented in Figure 17. Another important advantage of the model is that the number of binary variables does not depend on the total number of pieces, but just on the number of piece types. This makes this model particularly suited for problems where there are few piece types.

The weaknesses of the *Dotted-Board Model* come from its discrete nature, due to the board discretization. The number of binary variables varies with the square of the grid resolution chosen to discretize the board and is also dependent on the size of the board itself. The computational experiments have also proven this dependence, as the SHAPES group of instances is harder to solve than the BLAZEWICZ group and the model's performance decreases when the grid resolution is doubled, for the BLAZP2 and BLAZP4 groups of instances.

As the length of the board is unknown and the model's performance depends on the size of the board, the use of good upper bounds can dramatically improve the model efficiency. These upper bounds on the length of the board can be generated by a first, short-time run of the model, as we have done in the computational experiments reported here, or can be provided by heuristic methods as long as they preserve the discrete characteristic of the solutions. For instance, a bottom-left heuristic using as feasible positioning points the dots would be a possible approach to generate

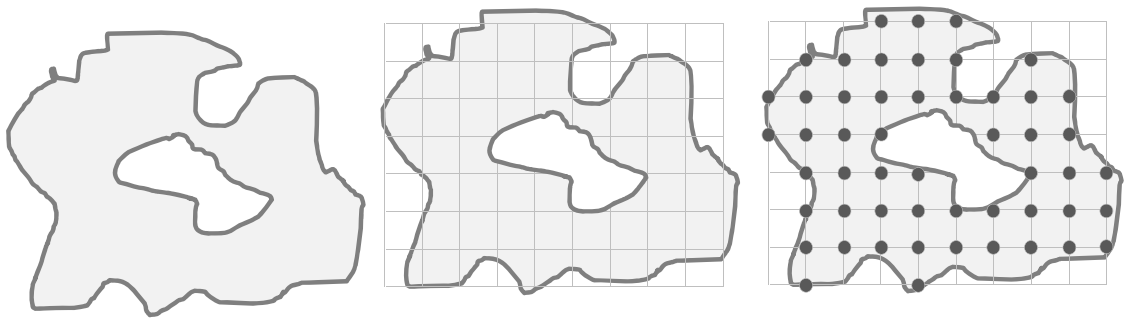

Figure 17: A non-convex board with a hole.

upper bounds.

As in all integer programming models, the availability of good lower bounds makes a difference. However, the linear relaxation of this model gives a poor lower bound, and the area lower bound and the longest piece lower bound are also not particularly good. This can be seen in the extremely large optimality gaps whenever the optimal solution is not reached. Following the evolution of the resolution process for some of the instances, even when optimal solutions were obtained it was common to have large gaps that suddenly converged to zero, a clear evidence of the poor quality of the lower bound. Further research should be carried on the development of tighter lower bounds.

Future research work will also address the generation of the dot pattern. In the current work the dots were provided by superimposing a regular grid over the board. However, the model itself imposes no constraint on the positions of the dots. There can be different grid resolutions along the x and the y axes, or different dot densities in different regions, allowing for a higher refinement on more complicated regions (e.g. holes, curves, narrow parts of the board). The additional complexity of a more sophisticated dot generation strategy puts an extra burden on the pre-processing phase only (model generation) and has no impact on the resolution.

In this work we have explored the modeling point of view where decisions are on the occupancy of (discretized) regions of the "container", rather than on the positioning point for each item. This approach can be successfully generalized to other cutting and packing problems. It is particularly adequate when accurately discretizing the placement region is not expensive (this is the case for items with a few congruent sizes) and when the assortment of items is very homogeneous (few piece types but many pieces in total). But this approach shows all its strength and flexibility when the placement region has complex placement constraints, as it happens in two-dimensional rectangular cutting problems with irregular non-convex boards and defects. This situation also arises in three-dimensional packing problems with obstructions, such as packing objects in satellites, where complex obstructed regions have to be taken into account, and packing boxes inside trucks, where boxes may be pre-positioned because of stability and weight balance constraints. For these reasons it is the authors' belief that the *Dotted-Board Model* represents a rather flexible framework to deal with cutting and packing problems that include complex geometric features which have been hard to address by existing approaches.

Acknowledgments This research was funded by FAPESP (2012/00464-4, 2010/10133-0), CNPq (300713/2010-0) and CAPES (CAPES BEX-1545/11-6) from Brazil, CAPES/FCT (no.273/2010)

from Brazil and Portugal, FP7-PEOPLE-2009-IRSES project no. 246881 from the European Commission, and ERDF through the Programme COMPETE and by the Portuguese Government through FCT – Foundation for Science and Technology (PTDC/EME-GIN/105163/2008).

We would like to express our deep gratitude to Marcos Arenales for his incentive and for his very valuable comments and suggestions. Our thanks also to the three anonymous reviewers whose constructive comments helped improving this paper.

References

[Alvarez-Valdes et al., 2009] Alvarez-Valdes, R., Parreño, F., and Tamarit, J. M. (2009). A branch and bound algorithm for the strip packing problem. *OR Spectrum*, 31(2):431–459.

[Beasley, 1985] Beasley, J. E. (1985). An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33(1):49–64.

[Bennell and Dowsland, 2001] Bennell, J. A. and Dowsland, K. A. (2001). Hybridising tabu search with optimization techniques for irregular stock cutting. *Management Science*, 47(5):1160–1172.

[Bennell and Song, 2008] Bennell, J. A. and Song, X. (2008). A comprehensive and robust procedure for obtaining the nofit polygon using Minkowski sums. *Computers & Operations Research*, 35(1):267–281.

[Bennell and Oliveira, 2008] Bennell, J. A. and Oliveira, J. F. (2008). The geometry of nesting problems: a tutorial. *European Journal of Operational Research*, 184(2):397–415.

[Bennell and Oliveira, 2009] Bennell, J. A. and Oliveira, J. F. (2009). A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60:S93–S105.

[Bennell and Song, 2010] Bennell, J. A. and Song, X. (2010). A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 10:167–188.

[Carravilla and Ribeiro, 2005] Carravilla, M. A. and Ribeiro, C. (2005). CP and MIP in the resolution of hard combinatorial problems: a case study with nesting problems. In *CSCLP 2005 Joint ERCIM/CoLogNET International Workshop on Constraint Solving and Constraint Logic Programming*:113–127.

[Carravilla et al., 2003] Carravilla, M. A., Ribeiro, C., and Oliveira, J. F. (2003). Solving nesting problems with non-convex polygons by constraint logic programming. *International Transactions in Operational Research*, 10(6):651–663.

[Castro and Oliveira, 2011] Castro, P. M. and Oliveira, J. F. (2011). Scheduling inspired models for two-dimensional packing problems. *European Journal of Operational Research*, 215(1):45–56.

[Egeblad et al., 2007] Egeblad, J., Nielsen, B. K., and Odgaard, A. (2007). Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, 183(3):1249–1266.

[Gomes and Oliveira, 2002] Gomes, A. M. and Oliveira, J. F. (2002). A 2-exchange heuristic for nesting problems. *European Jornal of Operational Research*, 141(2):359–370.

[Gomes and Oliveira, 2006] Gomes, A. M. and Oliveira, J. F. (2006). Solving irregular strip packing problems by hybridising simulated anealing and linear programming. *European Journal of Operational Research*, 171(3):811–829.

[Imamichi et al., 2009] Imamichi, T., Yagiura, M., and Nagamochi, H. (2009). An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6(4):345–361.

[Leung et al., 2012] Leung, S.C.H. and Lin, Y. and Zhang, D. (2012). Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, 3:678–686.

[Li and Milenkovic, 1995] Li, Z. and Milenkovic, V. (1995). Compaction and separation algorithms for non-convex polygons and their applications. *European Jornal of Operations Research*, 84:539–561.

[Martello et al., 2003] Martello, S., Monaci, M., and Vigo, D. (2003). An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319.

[Oliveira et al., 2000] Oliveira, J. F., Gomes, A. M., and Ferreira, J. A. S. (2000). TOPOS - A new constructive algorithm for nesting problems. *OR Spectrum*, 22(2):263–284.

[Ribeiro et al., 1999] Ribeiro, C., Carravilla, M. A., and Oliveira, J. F. (1999). Applying constraint logic programming to the resolution of nesting problems. *Pesquisa Operacional*, 19(2):239–247.

[Stoyan et al., 1996] Stoyan, Y. G., Novozhilova, M. V., and Kartashov, A. V. (1996). Mathematical model and method of searching for a local extremum for the non-convex oriented polygons allocation problem. *European Journal of Operational Research*, 92(1):193–210.

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