Nesting problems

Luiz Henrique Cherri

Luiz Henrique Cherri

Nesting problems

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação - ICMC-USP and to the Faculdade de Engenharia da Universidade do Porto - FEUP, in partial fulfillment of the requirements for the degrees of the Doctorate Program in Computer Science and Computational Mathematics (ICMC-USP) and of PhD (FEUP), in accordance with the international academic agreement for PhD double degree signed between ICMC-USP and FEUP. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics / Industial engenering

Advisor: Profa. Dra. Franklina Maria Bragion deToledo (ICMC-USP, Brasil)Advisor: Profa. Dra. Maria Antónia da Silva Lopes deCarravilla (FEUP, Portugal)

USP – São Carlos
June 2016

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

C521n

Cherri, Luiz Henrique
Nesting problems / Luiz Henrique Cherri; orientador Franklina Maria Bragion Toledo; co-orientador Maria Antónia da Silva Lopes Carravilla. -- São Carlos, 2016.
147 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2016.

1. Nesting problems. 2. Mixed-integer programming models. 3. Constraint programming models. 4. Heuristics. 5. Geometric tools. I. Toledo, Franklina Maria Bragion, orient. II. Carravilla, Maria Antónia da Silva Lopes, co-orient. III. Título.

Luiz Henrique Cherri

O problema de corte de peças irregulares

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP e àFaculdade de Engenharia da Universidade do Porto- FEUP, como parte dos requisitos para obtenção dos títulos de Doutor em Ciências - Ciências deComputação e Matemática Computacional (ICMC-USP) e PhD (FEUP), de acordo com o convênioacadêmico internacional para dupla titulação dedoutorado assinado entre o ICMC-USP e a FEUP. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Área de Concentração: Ciências de Computação e Matemática Computacional / Engenharia indústrial

Orientadora: Profa. Dra. Franklina Maria Bragion de Toledo (ICMC-USP, Brasil)
Orientadora: Profa. Dra. Maria Antónia da Silva Lopes de Carravilla (FEUP, Portugal)

USP – São Carlos
Junho de 2016

"*Find something that you enjoy doing so much that you would be willing to do it for nothing and you will never work a day in your life.*"

Anonymous

Acknowledgements

This thesis marks the end of a path that started three years ago, and this path I did not walk alone. I am glad to have many individuals that supported, inspired and understood me even in the most difficult times.

For my family, that is responsible for my education and gave me the unconditional support and motivation since the beginning. Especially for my parents, Maria and Luiz, for my brothers, Adriana and Cesar, and for my brothers-in-law, Leda and Celso. Furthermore, I am grateful for my niece, Maria Alice, who inspired me in the last nine months.

It would not be possible to explore all the content of this thesis without the orientation and support of my Brazilian advisor Franklina and my Portuguese advisor Maria Antónia. Thank you, Fran and Maria Antónia, you have been wonderful academic advisors and friends to me. I also would like to thank the Professor José Fernando and Cristina Ribeiro for the collaboration, ideas and friendship. Their ideas deeply contributed for the quality of this thesis.

I am grateful for all my friends of LOt - Laboratório de Otimização which aided me on this journey. In particular for Aline for the supporting in the writing of this thesis and for the great friendship and for Artur, Marcos and Leandro, whose friendship goes beyond the laboratory. I also thank my friends of laboratory at FEUP for the excellent reception, that made my period in Portugal so pleasurable.

I thank FAPESP - Fundação de Amparo a Pesquisa do Estado de São Paulo (grants BEPE/2014/10740-4 and 2012/18653-8) - for the financial support. I also must express my gratitude to the professors and staff of ICMC and FEUP.

Finally, I would like to thank everyone who helped me to finish one of the most important chapters of my life.

Abstract

The two-dimensional irregular cutting and packing problems (aka nesting problems) have been studied over the past six decades and consist in cutting (packing) convex and non-convex small pieces from (in) large boards without overlapping. There are several variants of this problem that are defined according to the board shapes and the objective of each problem. There are a number of heuristics proposed in the literature to solve irregular cutting and packing problems, but only few mixed-integer programming models. Specifically, these models were developed for the irregular strip packing problem, that consists in packing pieces into a single board with fixed width and length to be minimized. For the other problem variants, there is no exact methods presented in the literature. The main difficulty in solving irregular cutting and packing problems is how to handle with the geometric constraints. These constraints depend on the type of placement of the pieces on the board that can be continuous or discrete. In this thesis, we present two mixed-integer programming models for the irregular strip packing problem in which the pieces can be continuously placed on the board. These models do not demand complex structures to be built. We also present a new dot data structure to store the information on the placement of the pieces and overlapping positions bringing flexibility and efficiency to discrete approaches. Using this structure, a matheuristic is proposed, combining the advantages of the models with discrete and continuous placement positions for the pieces on the board. Furthermore, constraint programming models for several variants of irregular cutting and packing problems are exploited. For some variants, these models are the first modelling representation. A new global constraint is developed to eliminate the overlap among pieces. Computational experiments were conducted to evaluate the developed approaches.

Keywords: Irregular cutting and packing, mixed-integer programming models, constraint programming models, heuristics, geometric tools.

Resumo

Os problemas de corte e empacotamento de peças irregulares bidimensionais vêm sendo estudados há décadas e consistem em cortar (empacotar) peças menores, convexas e não convexas, a partir de (em) placas maiores de forma a não se sobreporem. Existem diversas variantes deste problema, definidas de acordo com o formato da placa e objetivo de cada problema. Na literatura, muitas heurísticas foram propostas para a resolução dos problemas de corte e empacotamento de peças irregulares, porém, poucos modelos de programação inteira mista podem ser encontrados. Especificamente, estes modelos foram desenvolvidos para o problema de empacotamento em faixa, que consiste em empacotar as peças em uma placa de largura fixa e comprimento a ser minimizado. Para as demais variantes do problema, não existem métodos exatos propostos na literatura. A principal dificuldade na resolução dos problemas de corte e empacotamento de peças irregulares está na manipulação das restrições geométricas. Estas restrições dependem do tipo de posicionamento das peças na placa, que pode ser discreto ou contínuo. Nesta tese, apresentamos dois modelos de programação inteira mista para o problema de empacotamento de peças em faixa, no qual cada peça pode ser alocada de forma contínua na placa. Estes modelos não demandam estruturas complexas para serem construídos. Também apresentamos uma nova estrutura de dados para armazenar informações sobre o posicionamento das peças e as posições de sobreposição, trazendo flexibilidade e eficiência para abordagens discretas. Utilizando esta estrutura, uma matheuristica foi proposta, combinando as vantagens dos modelos com alocação discreta e contínua das peças na placa. Além disso, modelos de programação por restrições para diversas variantes dos problemas de corte e empacotamento de peças irregulares foram explorados. Para algumas variantes, estes modelos são a primeira representação via modelagem. Uma nova restrição global foi desenvolvida para eliminar a sobreposição entre as peças. Experimentos computacionais foram realizados para avaliar as abordagens propostas.

Palavras chave: Corte e empacotamento de peças irregulares, modelos de programação inteira mista, modelos de programação por restrições, heuristicas, ferramentas geométricas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Irregular cutting and packing problems are hard combinatorial optimization problems (Fowler et al., 1981) that consist in cutting small, convex or non-convex items (pieces) from larger objects (boards). The items are placed on the object - with no overlapping among them - and completely contained in the boards. The objective may vary according to the application, however, it generally invokes waste (the boards' area not occupied by pieces) reductions or/and increase in the profits.

The irregular cutting and packing problems are not only scientifically relevant, but also economically and environmentally important, given their many industrial applications and the reductions in the use of raw materials they involve. From an economic point-of-view, a solution to the problem reduces the amount of material necessary for the production of the pieces, hence, the production costs, and contributes to waste reduction, as industries tend to discard less raw-material, which provides environmental benefits.

Industries of garment, furniture and shoe manufacture, sheet metal cutting and others are faced with such a problem. Figure 1.1 illustrates a solution to an instance of the problem.

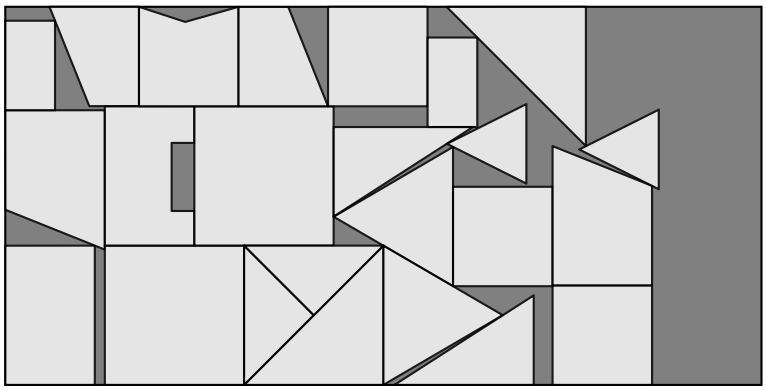


Figure 1.1: Example of a solution to the two-dimensional irregular cutting and packing problem.

The irregular cutting and packing problems comprehend several variants classified by Wäscher et al. (2007). For many of these variants no exact method or mathematical model has been developed to solve or represent them. Specifically, the exact methods proposed focus on a single variant, i.e., the irregular strip packing problem (or one open dimensional problem), as only heuristics have been proposed for the other variants.

The most studied variant is the irregular strip packing problem, in which all pieces must be placed into a board of fixed height and variable length, so that the minimum length is used. Some exact methods have been developed to solve this problem. A mixed-integer programming model in which pieces can be freely placed on the board was proposed by Fischetti and Luzzi (2009). Some structures of this model were formalized by Alvarez-Valdes et al. (2013), who also developed a branch and bound algorithm to solve the problem and extended a linear compaction model designed by Gomes and Oliveira (2006) for a mixed-integer programming model that represents the whole problem. Although all such models accurately represent the problem, they demand complex geometric structures not easy to be obtained.

Considering a finite set of positions for placing each piece, Carravilla et al. (2003) and Ribeiro and Carravilla (2004) proposed constraint programming methods to solve the problem. Toledo et al. (2013) developed a mixed-integer programming model to represent the problem also considering a discrete approximation. This geometric representation is also used by heuristics (Carravilla and Ribeiro, 2005; Bennell and Dowsland, 2001; Dowsland et al., 1998). Using a predetermined set of positions for the placement of the pieces, some geometric features can be determined prior to the application of the solution method. Despite the simplicity introduced by discrete approaches, some valuable solutions may be lost because of the discretization.

Several authors have combined heuristics with linear and non-linear programming to obtain more compact layouts. Examples can be found in the simulated annealing algorithm (Gomes and Oliveira, 2006), the hybrid tabu search (Bennell and Dowsland, 2001), and the iterated local search (Imamichi et al., 2009). The methods combine the efficiency of the heuristics with the compactness of the layouts generated by linear and non-linear programming. Although in the last decade several mixed-integer programming models were proposed to represent the irregular strip packing problem, they were never used for the development of a heuristic for the irregular strip packing problem.

1.1 Contributions of this thesis

The contributions of this thesis regard the development of innovative exact and heuristic methods to solve the irregular cutting and packing problems. We propose two mixed-integer programming models for the strip packing problem; a new dot structure

to handle the geometry of cutting and packing problems; pieces rotations were included to the dotted board model (Toledo et al., 2013) and a matheuristic was build using it; constraint programming methods were proposed to all variants of cutting and packing problems classified by Wäscher et al. (2007). Also, a global constraint is proposed to eliminate the overlap between pieces. In the following the contributions of each subject studied are described.

Two mixed-integer programming models to solve the irregular strip packing problem are proposed. In both models, the placement of the pieces is continuous inside the board. They differ on how the non-overlapping among pieces is ensured. One of them guarantees that the pieces do not overlap using only the information of the piece vertices, i.e., complex structures, as nofit polygons or phi-functions are not necessary for the construction of the model. The other model uses the nofit polygon covering to avoid overlapping among pieces and outperformed the best results from the literature for exact methods. In both approaches, the geometric concepts used for the creation of the model are simpler than those of previous models proposed in the literature and easily deal with non convex pieces and pieces with holes. They are also the first continuous models that enable the rotation of the pieces.

An innovative dot structure is proposed to deal with the geometry of the problem according to a discrete placement of the pieces inside the board. It converts the geometric analyses over polygons into information of the dots. Each piece type can be placed in its own set of dots that can be different from the set of dots for other types of pieces. The use of different dots for the placement of different piece types has never been explored in the literature. Therefore, the structure simplifies and improves the efficiency of the creation of solution methods in which pieces can only be placed over specific positions.

The proposed dot structure enables the dotted board model to be reformulated, allowing it to be constructed using a specific set of dots for each piece type. Therefore, a model that represents the problem more precisely is generated with fewer variables and solutions with better quality. Rotations for the pieces with a finite number of angles are considered in the reformulation of the dotted board model.

As the dot structure simplifies the management of the dots on the board, a model-based heuristic has been developed with this new structure. The matheuristic uses the dotted board model in two phases and a linear compaction model in the last phase. In the first and second phases, a constructive heuristic based on relax and fix obtains an initial feasible solution that is improved through the addition of more dots, so that the pieces can be placed and then in the second phase local searches are performed using the dotted board model. In the last phase, a linear compaction model eliminates the gaps among pieces.

We also propose new constraint programming models to solve all variants of the irregular cutting and packing problems classified by Wäscher et al. (2007), as the literature lacks an exact method that solves many of such problems. Although each problem has specific constraints, some general characteristics, as non-overlap among pieces, are required in feasible solutions by all problem variants. Therefore, we propose new constraint programming models to represent all these variants. Three solution methods were developed for each problem variant and they differ only in the way they represent the domains of the variables and deal with the core constraints of the problem. A global constraint to avoid the overlapping between pieces is also proposed. It promotes quick search with less memory usage in comparison with the built-in constraints of the constraints solver.

1.2 Thesis outline

After this introductory chapter, Chapter 2 addresses the definition of irregular cutting and packing problems and provides a review of the geometry of the heuristic methods and the mixed-integer programming models developed for the irregular strip packing problems. Two new mixed-integer programming models for the irregular strip packing problem are presented in Chapter 3. An innovative dot structure that represents the geometry of irregular cutting and packing problems and simplifies the construction of heuristics and models based on discrete placement positions for the pieces is described in Chapter 4. A matheuristic that uses the structure combined with the dotted board model is proposed in Chapter 5. Chapter 6 presents new constraint programming models to solve the irregular cutting and packing problem variants classified by Wäscher et al. (2007). Finally, Chapter 7 provides the conclusions and future research directions.

Chapter 2

Irregular cutting stock problem

In the literature, the cutting and packing problems are classified into different variants. The first classification scheme was proposed by Dyckhoff and Finke (1992) and was later extended by Wäscher et al. (2007). This classification is based on five criteria: dimensionality, kind of assignment, assortment of pieces, assortment of containers (or boards in the two-dimensional case) and shape of the pieces (for problems with more than one dimension).

The dimensionality concerns the number of relevant dimensions of the problems that can have one, two, three or more dimensions. Output value maximization and input value minimization are two kinds of assignment considered. The pieces can be classified as strongly heterogeneous (many pieces of many types), weakly heterogeneous (many pieces of few types) and identical (many pieces of a single type). The assortment of the boards can be classified into several large boards with fixed dimensions and a single board with fixed or variable dimensions. If the problem has two dimensions or more, the shapes of the pieces can be classified into regular (e.g., rectangle, circles, boxes, spheres, etc.) and irregular (non-regular).

In this thesis, we study the irregular two-dimensional cutting and packing problems, that, according to the typology of Wäscher et al. (2007), can be classified into six basic types: Identical Item Packing Problem (*IIPP*), Placement Problem (*PP*), Knapsack Problem (*KP*), Cutting Stock Problem (*CSP*), Bin Packing Problem (*BPP*) and Open Dimension Problem (*ODP*). *IIPP*, *PP* and *KP* are output maximization problems and *ODP*, *BPP* and *CSP* are input minimization problems. Figure 2.1 shows a diagram that classifies the basic two-dimensional cutting and packing problem types.

Looking at the output maximization problems, in the *IIPP* the problem consists in placing many copies of the same item type on the board. In *PP*, many piece types with several copies of each one must be placed on the board. The number of copies of each piece type that need to be cut can be finite (*PPr*) or large enough to be considered infinite (*PP*). If exactly one copy of many piece types must be placed on the board, the

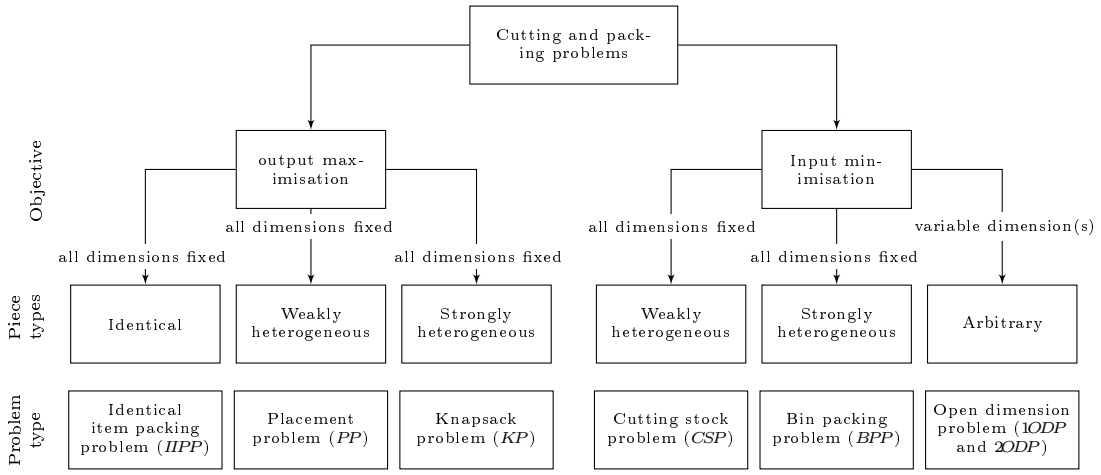


Figure 2.1: Basic cutting and packing problem types (adapted from Wäscher et al. (2007)).

problem is called *KP*. In these problems the board has finite dimensions and, usually, there is no space on the board to cut all the demanded pieces. The objective is to extract the maximum value performing the cut (pack) of the pieces.

For input minimization problems, there are sufficient resources to cut all pieces and the objective is to minimize the used resource. In *CSP*, a minimum number of boards must be used to place many piece types with several copies of each. When only one piece of each type must be placed on the boards, the problem is called *BPP*. In *ODP*, many piece types with several copies of each one must be placed on the boards with one (1*ODP*, also known as strip packing problem) or two (2*ODP*) variable dimensions. The objective for *CSP*, *BPP* and 1*ODP* is typically to minimize the amount of resources used to cut all the demanded pieces. This objective can be reached by minimizing the number of boards used or the length of the board used to perform the cut. Solving 2*ODP*, several objectives can be used, for example, the area or perimeter of the bounding box of the packing or other relations between the length and the width of the board.

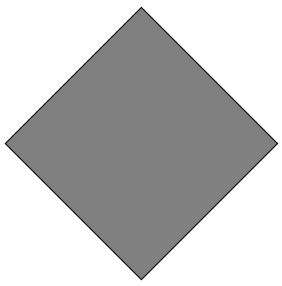
The focus of this thesis is the irregular strip packing problem. However, other variants of cutting and packing problems were investigated. In the reminder of this chapter, Section 2.1 describes the possible geometric representations for pieces and how to use them to tackle the geometric constraints of the problem. A review of exact methods and mixed-integer programming models is presented in Section 2.2. Some heuristics for the irregular strip packing problem are reviewed in Section 2.3. Heuristics for the other variants of irregular cutting and packing problems are presented in Section 2.4.

2.1 Geometry representation

The irregular cutting and packing problems have two common constraints: i) ensure that the pieces do not overlap; and ii) the guarantee of the pieces are completely on the board. The solution methods for these problems are directly related with the geometry used to represent the pieces and, consequently, to handle these constraints. A complete review about the problem geometry can be found in Bennell and Oliveira (2008).

In the literature, different approaches were used to represent the geometry of the problems. The most common are: raster points, D-functions, nofit polygons and phi-functions.

Approaching the problem by raster points, the pieces and the board are represented by a set of matrices. These matrices contain the positions occupied by each piece on a grid, as exemplified in Figure 2.2. The representation of the board and the pieces is similar. Furthermore, the matrices that represent the board contain all the positions that are free or occupied by the pieces. Using this representation, the pieces intersection analysis is reduced to evaluate if the matrix that composes the pieces, when placed on the board, overlap in non-zero positions. In addition, the piece is inside of the board if all the non-zero positions that compose its matrix are inside of the board. Although it is a simpler approach, the raster points cannot represent the pieces precisely generating some gaps among the pieces and the refinement of the representation of the pieces can demand huge computational resources.



0	0	0	0	0	1	1	0	0	0	0	
0	0	0	0	1	1	1	1	0	0	0	
0	0	0	1	1	1	1	1	1	0	0	
0	0	1	1	1	1	1	1	1	1	0	
0	1	1	1	1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	
0	1	1	1	1	1	1	1	1	1	0	
0	0	1	1	1	1	1	1	1	0	0	
0	0	0	1	1	1	1	1	0	0	0	
0	0	0	0	1	1	1	1	0	0	0	
0	0	0	0	0	1	1	0	0	0	0	

Figure 2.2: A piece (on the left) and its matrix representation (on the right).

The gap among the pieces can be reduced using polygons to represent the pieces. Each piece is represented by a set of vertices and a reference point used to control where the piece will be placed on the board. To ensure that the piece is entirely contained on the board, the innerfit polygon (*IFP*) is used. The *IFP* of a piece t (IFP_t) represents all the positions where the reference point of piece t can be placed keeping the piece entirely inside of the board. Note that, if the board is a rectangle with length L and width W, the *IFP* is also a rectangle. Consider l^{left} the horizontal distance from the leftmost piece vertex to the reference point, w^{top} the vertical distance from the highest piece vertex to the reference point, w^{bottom} the vertical distance from the lowest

piece vertex to the reference point, and l^{right} the distance of the piece reference point to its rightmost vertex. The piece dimensions are illustrated in Figure 2.3a where its reference point is highlighted in a darker circle. The *IFP* is the rectangle with left bottom vertex (l^{left}, w^{bottom}) and the top right vertex $(L - l^{right}, W - w^{bottom})$ as illustrated in Figure 2.3b.

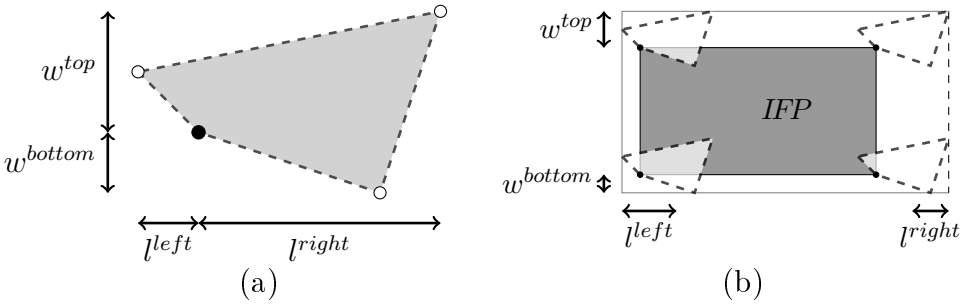


Figure 2.3: Building an innerfit polygon. (a) shows how l^{left}, w^{top}, l^{right} and w^{bottom} are obtained. In (b) the *IFP* is built using the constants found in (a).

Performing the overlapping analysis between the pieces using this geometric representation is a complex task. One of the techniques used for this analysis is the D-function, that determines whether a point is to the left or to the right side of an oriented line. Therefore, the overlap between pieces can be verified using the information of vertices and edges of polygon that compose the piece. The use of D-functions is exemplified in Figure 2.4 where the point B is over the D-function, C is on the right side of the D-function and A is on the left side of the D-function.

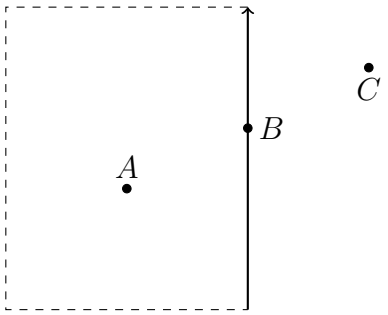


Figure 2.4: A D-function and three points in different positions.

To simplify the verification of non-overlap among pieces, the nofit polygon can be used. The nofit polygon represents the contact area between two pieces reducing the problem of checking if two pieces overlap is reduced to the verification if a point is strictly inside of this polygon. The nofit polygon between two pieces is illustrated in Figure 2.5.

Aiming to combine the simplicity of raster points with the precision of the polygonal representation, some authors used discrete positions (dots) to place the pieces, while the

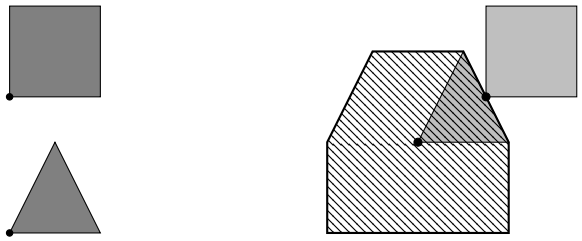


Figure 2.5: The nofit polygon composed by the pieces on the left is presented hatched on the right.

non-overlap analysis is performed using nofit polygons. As the placement positions for the pieces are known, it is possible to predict which dots of the board cause intersection of pieces if both representations are used simultaneously. Figure 2.6 illustrates an example using the pieces in Figure 2.5, where if the triangle is placed over a grid dot, the square reference point cannot be placed at the dots inside of the nofit polygon of these pieces (that are highlighted by larger circles).

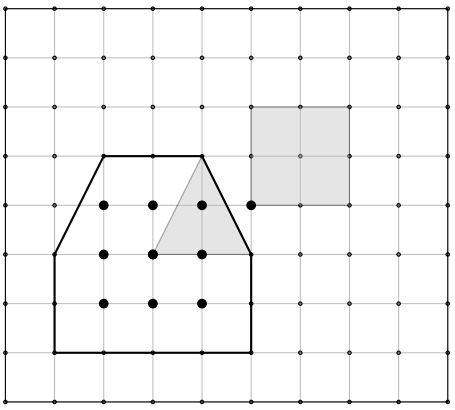


Figure 2.6: Example of a board discretization and a feasible placement position for a triangle and a square.

To represent the pieces with better precision, linear or non-linear functions (phi-functions) can be used. The phi-function of two pieces is the function that gives the distance between them. Although this representation is more general and accurate, the task of define functions for all pieces is difficult. Generally, the phi-functions are obtained combining primary objects, i.e, objects that the phi-functions are already known. Figure 2.7 illustrates the representation of pieces using phi-functions.

It is possible to represent the pieces using circles to cover its entire surface. This representation is simpler than representing the pieces by any functions, however, it is not precise and can lead to small intersections between the pieces. In addition, define the a set of circles to cover the pieces can be a difficult task. Figure 2.8 shows a representation of a rectangle using circles.

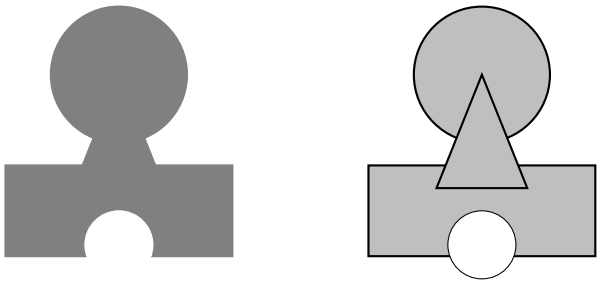


Figure 2.7: Representing phi-functions (based on Chernov et al. (2010)). The polygon is on the left and a possible representations using phi-functions is on the right.

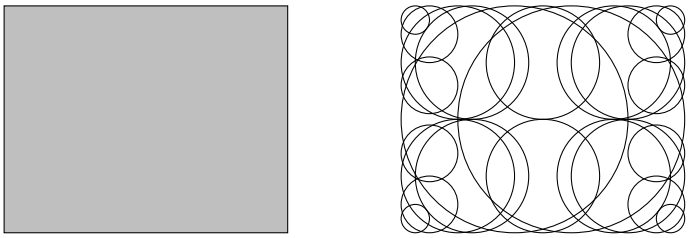


Figure 2.8: Representing a piece by circles. The piece is on the left and a possible representations is on the right.

2.2 Exact approaches for the irregular strip packing problem

The irregular strip packing problem consists in cutting a number of convex and concave pieces from a rectangular board of fixed width (W) and infinite length. The objective is to minimize the board length (L) used to perform the cuts. This problem is NP-complete (Nielsen and Odgaard, 2003) and because of its difficulty, only few exact methods were proposed to solve it. By solving this problem by exact solution methods, a search over the solution space of the problem is made, ensuring the solution optimality. Besides solving the problem to optimality, these methods are also a great tool to analyse the quality of heuristic methods. All exact methods proposed to the irregular cutting and packing problem deal with the irregular strip packing problem variant. We present here an exact constraint programming method and the mixed-integer approaches proposed in the literature to solve the irregular strip packing problem.

2.2.1 An exact method using constraint programming

Based on formal logic, logical programming can represent combinatorial optimization problems by a set of logical sentences, expressing some rules to build the problem domain. Constraint logical programming extends logical programming by allowing the problem to be also represented by constraints instead of only by logical statements

(Jaffar and Maher, 1994).

Carravilla et al. (2003) developed an exact method to solve the irregular strip packing problem based on constraint logical programming, a finite domain variables and an implicit enumeration scheme. The method deals with problems with non-convex pieces through the decomposition of the nofit polygon in convex parts. The placement position of each piece i is represented by discrete (x_i, y_i) coordinates of its reference point. The initial domain for the (x_i, y_i) coordinate is built using the width, the upper bound of the board length and the piece i dimensions as depicted in Figure 2.3. The nofit polygons are used to avoid the overlap among pieces. When the position of one piece is fixed, the domain of the variables related to all other pieces is reduced based on their NFP's.

On the enumeration phase, the choice of the placement points follows the order of the pieces. Along the search, the best solution found so far is memorized and every piece positioned beyond the best solution is not considered.

Ribeiro and Carravilla (2004) proposed a global constraint called "outside", the first global constraint designed for the irregular strip packing problem. With this constraint, expressing the problem is easier and the authors claim that its resolution become more efficient. They also proposed an improved pruning procedure of the search that is based on the concept of the piece "responsible" for the total length.

2.2.2 A linear model based on compaction strategies

A mixed-integer programming model to represent the irregular strip packing problem can be derived from the compaction models that were used in literature as part of heuristic methods (Li and Milenkovic, 1995; Stoyan et al., 1996; Bennell and Dowsland, 2001; Gomes and Oliveira, 2006). Alvarez-Valdes et al. (2013) presented the complete formulation for the problem based on this model.

Different from Carravilla et al.'s method, in this mathematical model the placement coordinates of each piece i are defined by continuous coordinates (x_i, y_i). The total number of pieces is denoted by $\mathcal{N}$ and the reference point of piece i, p_i, has coordinates denoted by (x_i, y_i), $i = 1, ..., \mathcal{N}$. To ensure that piece i is entirely on the board, IFP_i is represented by constraints (2.1)-(2.3), that guarantee that the piece reference point is always in a feasible position in relation to the board.

$$y_i \leq W - w_i^{bottom}, \quad i = 1, ..., \mathcal{N}, \tag{2.1}$$

$$y_i \geq w_i^{top}, \quad i = 1, ..., \mathcal{N}, \tag{2.2}$$

$$x_i \geq l_i^{left}, \quad i = 1, ..., \mathcal{N}. \tag{2.3}$$

where w_i^{top}, l_i^{left} and w_i^{bottom} are constants associated with the piece as shown in Figure 2.3.

To avoid the overlap between pieces i and j, consider the NFP_{ij}. Define E_{ij} as the set of lines that correspond to edges of the NFP_{ij} as illustrated in Figure 2.9b. The equation of the line $e \in E_{ij}$ corresponds to an edge of NFP_{ij} and is given by $\alpha_{ije}(x_j - x_i) + \beta_{ije}(y_j - y_i) = \gamma_{ije}$, where α_{ije}, β_{ije} and γ_{ije} are the coefficients of a line $e \in E_{ij}$.

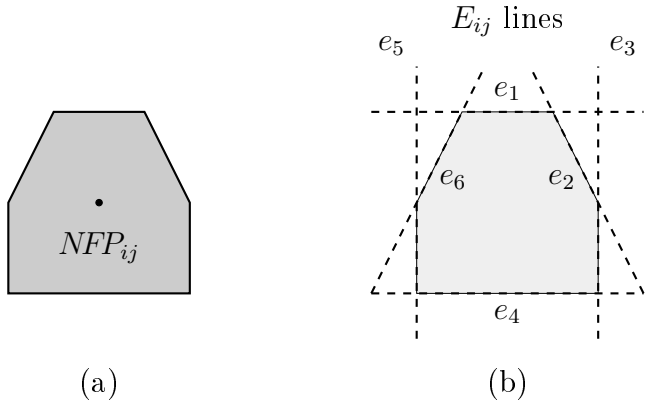


Figure 2.9: In (a) a *NFP* is presented and in (b) the lines associated with the respective edges are shown.

Suppose that piece i is fixed. To avoid overlap the reference point of piece j must be over the edge of NFP_{ij} or outside NFP_{ij} (the reference points of i and j must be on opposite sides of at least one edge of NFP_{ij}). Figure 2.10 illustrates the cases that the pieces are separated (Figure 2.10a), touching (Figure 2.10b) and overlapping (Figure 2.10c). Since it is not possible for two pieces to be at different sides of all lines defined by *NFP* edges, new variables are necessary to relax some of these constraints. For each line $e \in E_{ij}$, a binary variable v_{ije} is defined, which is 1 if pieces i and j are on different sides or touching line e, and 0 otherwise. The general form of the constraints that prevent overlap is defined by:

$$\alpha_{ije}(x_j - x_i) + \beta_{ije}(y_j - y_i) \leq \gamma_{ije} + M(1 - v_{ije}), \qquad 1 \leq i < j \leq \mathcal{N}, \forall e \in E_{ij}, \quad (2.4)$$

where M is a real constant that is large enough to relax this constraint if $v_{ije} = 0$. To ensure that at least one variable v_{ije} associated with lines $e \in E_{ij}$ is active, constraints (2.5) are imposed.

$$\sum_{e \in E_{ij}} v_{ije} \geq 1, \qquad 1 \leq i < j \leq \mathcal{N}. \quad (2.5)$$

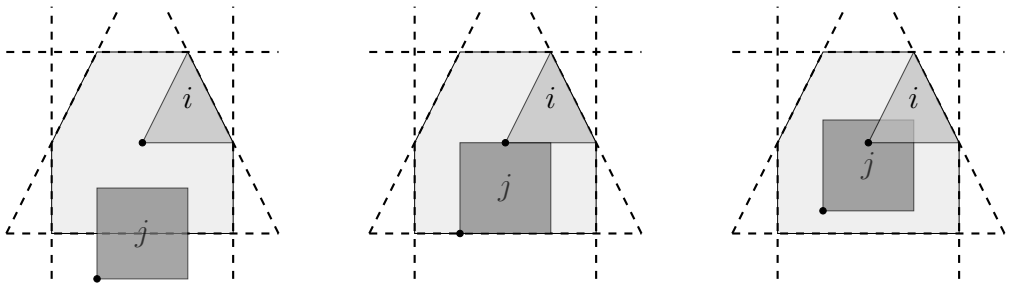


Figure 2.10: In (a) the reference points of the pieces are at different sides of a line, then the pieces are separated. In (b), the reference point of piece j touches one line, then the pieces touch. In (c), the reference point of both pieces are at the same side of all lines, then the pieces overlap.

The objective is to minimize the used board length L. Based on constraints (2.6), the variable L is always larger than the used board length, consequently, the objective is to find the smallest L that satisfies all these constraints.

$$L - x_i \geq l_i^{right}, \qquad i = 1, ..., \mathcal{N}, \tag{2.6}$$

where l_i^{right} is a constant of piece i as presented in Figure 2.3.

The mixed-integer programming model presented in Alvarez-Valdes et al. (2013) is given as follows.

$$\text{minimize} \quad L \tag{2.7}$$

subject to:

$$L - x_i \geq l_i^{right}, \qquad i = 1, ..., \mathcal{N}, \tag{2.8}$$

$$x_i \geq l_i^{left}, \qquad i = 1, ..., \mathcal{N}, \tag{2.9}$$

$$y_i \leq W - w_i^{bottom}, \qquad i = 1, ..., \mathcal{N}, \tag{2.10}$$

$$y_i \geq w_i^{top}, \qquad i = 1, ..., \mathcal{N}, \tag{2.11}$$

$$\alpha_{ije}(x_j - x_i) + \beta_{ije}(y_j - y_i) \leq \gamma_{ije} + M(1 - v_{ije}), \quad 1 \leq i < j \leq \mathcal{N}, \forall e \in E_{ij}, \tag{2.12}$$

$$\sum_{e \in E_{ij}} v_{ije} \geq 1, \qquad 1 \leq i < j \leq \mathcal{N}, \tag{2.13}$$

$$v_{ije} \in \{0, 1\}, \quad i, j = 1, ..., \mathcal{N}, \forall e \in E_{ij}, \tag{2.14}$$

$$x_i, y_i \geq 0, \qquad i = 1, ..., N. \tag{2.15}$$

Despite the simplicity of this model, it is difficult to develop constraints for non convex polygons or polygons with holes. Precisely, in these cases two problems as-

sociated with the nofit polygons are found: (i) the generation, that can be difficult considering any piece shapes; and (ii) the usage, since the authors did not detail how to handle holes and non convexities in the model.

2.2.3 The Fischetti and Luzzi (2009) model

Fischetti and Luzzi (2009) proposed the first mixed-integer programming model to represent the irregular strip packing problem. The authors performed a lifting over some coefficients of the model, reducing the solution space and then improving the performance of solution methods. To tackle the geometric constraints, the nofit polygon and the innerfit polygon were used. The difference between this model and the one presented in Section 2.2.2 is the way of preventing the overlap.

Given two pieces i and j and the nofit polygon between them (NFP_{ij}), the locus of point (x_j, y_j) of piece j must not be inside of NFP_{ij} translated by (x_i, y_i) from the origin, i.e., $(x_j, y_j) - (x_i, y_i)$ must be contained in $\overline{NFP}_{ij}$, where $\overline{NFP}_{ij}$ is the complementary set of NFP_{ij}. In order to derive linear constraints preventing overlaps, the authors divided the $\overline{NFP}_{ij}$ in m_{ij} convex and disjoint sub-regions (slices) whose union set must be equal to the $\overline{NFP}_{ij}$. These regions are created by drawing lines from convex NFP_{ij} vertices to its exterior side. Only the vertices which are formed by segments with external angle bigger than 180 degrees are chosen to trace these lines. The authors did not explain clearly how to choose the direction of each line. Figure 2.11 illustrates one possible $\overline{NFP}_{ij}$ division.

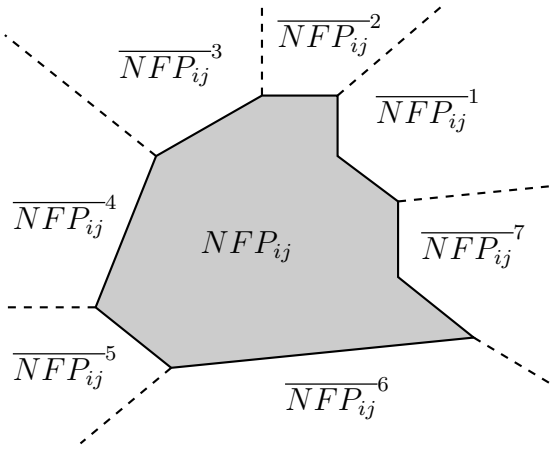


Figure 2.11: Partitioning the $\overline{NFP}_{ij}$ region in seven convex regions.

To ensure that piece j does not overlap piece i, the reference point of piece j must be in one of the slices $\overline{NFP}^k_{ij}$, $k = 1, ..., m_{ij}$. Consider the variable u_{ijk} that is equal to one if the reference point of piece j is allocated on the slice $\overline{NFP}^k_{ij}$, and 0 otherwise. Any $\overline{NFP}^k_{ij}$ slice can be expressed by a set of s^k_{ij} linear inequalities. Then, a natural

way to ensure that the pieces do not overlap is the use of inequalities similar to those presented in (2.4). These constraints are:

$$\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \gamma_{ij}^{kf} + M(1 - u_{ijk}), \quad 1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, f = 1, ..., s_{ij}^{k}, \tag{2.16}$$

where M is large enough to ensure that constraints (2.16) are valid.

The authors lifted the M term of constraints (2.16) by calculating a specific coefficient for each combination of constraint indices. In order to define these coefficients, the M term in expressions (2.16) is replaced by a new set of terms related to each pair of pieces i and j resulting in:

$$\begin{aligned} \alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \gamma_{ij}^{kf} + \sum_{h=1}^{m_{ij}} \theta_{ij}^{kfh} u_{ijh}, \\ 1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, f = 1, ..., s_{ij}^{k}. \end{aligned} \tag{2.17}$$

Note that $\sum_{h=1}^{m_{ij}} \theta_{ij}^{kfh} = 1$, and then γ_{ij}^{kf} can be expressed as $\gamma_{ij}^{kf} \sum_{h=1}^{m_{ij}} \theta_{ij}^{kfh}$. Consequently, constraint (2.17) can be rewritten as:

$$\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \phi_{ij}^{kfh} u_{ijh}, \quad 1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, f = 1, ..., s_{ij}^{k}, \tag{2.18}$$

where $\phi_{ij}^{kfh} = \theta_{ij}^{kfh} + \gamma_{ij}^{kf}$.

In order to keep the inequality valid, each coefficient ϕ_{ij}^{kfh} is defined as the maximum value of the left-hand side of the inequality if the variable $u_{ijh} = 1$. These values are computed by:

$$\begin{aligned} \phi_{ij}^{kfh} = \max_{((x_j, y_j) - (x_i, y_i)) \in \overline{NFP}_{ij}^{h} \bigcap B} \left(\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \right), \\ 1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, f = 1, ..., s_{ij}^{k}, \end{aligned}$$

where B is a rectangle large enough to contain the possible placements of pieces i and j, for example, a rectangle with width $2W$ and length $2\overline{L}$ (where $\overline{L}$ is an upper bound of the board length).

The model must ensure that exactly one slice of $\overline{NFP}_{ij}$ is active. Then, constraints (2.19) are defined.

$$\sum_{k=1}^{m_{ij}} u_{ijk} = 1, \qquad 1 \leq i < j \leq \mathcal{N}. \tag{2.19}$$

Additional constraints are defined as in (2.8) - (2.11) and the objective is also to minimize the used length of the board L. A second objective imposed by the authors is that the pieces must be positioned as low and to the left as possible. Fischetti and Luzzi's model is presented as follows.

$$\text{minimize} \quad L - \epsilon \textstyle\sum_{i=1}^{\mathcal{N}}(x_i + y_i) \tag{2.20}$$

subject to:

$$L - x_i \geq l_i^{right}, \qquad i = 1, ..., \mathcal{N}, \tag{2.21}$$

$$x_i \geq l_i^{left}, \qquad i = 1, ..., \mathcal{N}, \tag{2.22}$$

$$y_i \leq W - w_i^{bottom}, \qquad i = 1, ..., \mathcal{N}, \tag{2.23}$$

$$y_i \geq w_i^{top}, \qquad i = 1, ..., \mathcal{N}, \tag{2.24}$$

$$\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \phi_{ij}^{kfh} u_{ijh},$$
$$1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, f = 1, ..., s_{ij}^k, \tag{2.25}$$

$$\sum_{k=1}^{m_{ij}} u_{ijk} \geq 1, \qquad 1 \leq i < j \leq \mathcal{N}, \tag{2.26}$$

$$u_{ijk} \in \{0, 1\}, \quad i, j = 1, ..., \mathcal{N}, k = 1, ..., m_{ij}, \tag{2.27}$$

$$x_i, y_i \geq 0, \qquad i = 1, ..., N. \tag{2.28}$$

With this model, the authors were able to solve to optimality problems with up to seven pieces. As in the model presented in Section 2.2.2, it is not clearly stated how to solve problems when the nofit polygons have holes or more difficult convexities. Furthermore, the authors did not state clearly how to generate the slices of the nofit polygon complement used to avoid the overlap among pieces.

2.2.4 Alvarez-Valdes et al. (2013) approach

Alvarez-Valdes et al. (2013) reviewed the mixed-integer models presented in Sections 2.2.2 and 2.2.3 and defined a procedure to design the slices of Fischetti and Luzzi (2009) model. Furthermore, they lifted the bound constraints (2.21)-(2.23) and developed a branch-and-cut method to solve the model proposed by Fischetti and Luzzi (2009).

The procedure to define the slices is performed in three steps. In order to explain

these steps, consider pieces i and j as illustrated in Figure 2.12 and the nofit polygon between them (Figure 2.13a). The complementary region of the polygon in Figure 2.13a must be divided into slices. First, the authors define slices to the holes and dots inside of the NFP_{ij}, associating them with variable u_{ijk}, i.e., each hole becomes one of the slices where piece j can be placed. If a hole is not convex, it is divided into convex polygons and one variable is associated with each polygon. In Figure 2.13b, the first step is exemplified. The second step is iterative and consists in filling the concavities of the nofit polygon. A variable u_{ijk} is associated with each concavity of NFP_{ij} as illustrated in Figures 2.13c and 2.13d. Finally, in the third step, the exterior regions of a convex polygon are defined by cutting the region in horizontal slices as exemplified in Figure 2.13e.

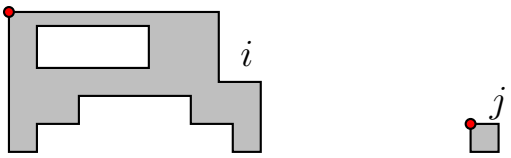


Figure 2.12: Pieces i and j and their reference points.

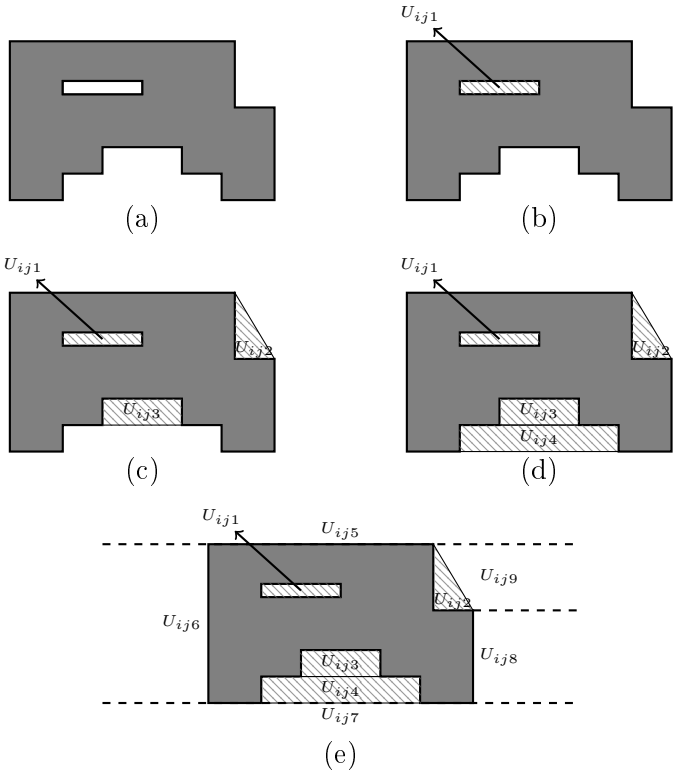


Figure 2.13: Creating the slices to Fischetti and Luzzi (2009) model. (a) Shows the nofit polygon between the piece i and piece j. In (b) the holes are identified as slices. (c) and (d) fill the the holes by setting them as slices. The division of the polygon complementary area in convex horizontal slices is shown in (e).

Having defined the slices, the authors lifted the bound constraints (2.21)-(2.23). Consider the slices of $\overline{NFP}_{ij}$ between pieces i and j as illustrated in Figure 2.14. For each slice, constants $\underline{x}_{ijk}$ and $\overline{x}_{ijk}$ represent the minimum and maximum values respectively that x_j can assume if $u_{ijk} = 1$. Likewise, constants $\underline{y}_{ijk}$ and $\overline{y}_{ijk}$ represent the minimum and maximum value that y_j can take if $u_{ijk} = 1$. Moreover, the constant $\underline{X}^{ij}$ ($\underline{Y}^{ij}$) represents the rightmost (lowest) point of the NFP_{ij}. These values are illustrated in Figure 2.14 for slice 2 (u^2_{ij} depicted in darkgray in Figure 2.14).

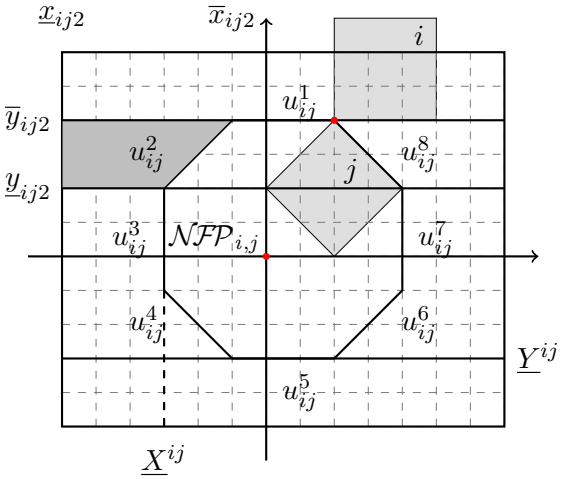


Figure 2.14: Slices generated in the exterior region of a nofit polygon defined by pieces i and j (based on Alvarez-Valdes et al. (2013)).

In order to represent the relative position of the reference point of pieces i and j, four subsets of variables are defined. Consider DN_{ij} (DS_{ij}) as the subset of variables that contain the reference point of piece j on its respective slice and ensure that it is above (below) the reference point of piece i. Similarly, subsets DE_{ij} (DR_{ij}) contain the variables associated with the slices that the reference point of piece j is to the right (left) of the reference point of piece i. These sets are described below.

$$DN_{ij} = \{u_{ijk} \in \overline{NFP}^k_{ij} | k = 1, ..., m_{ij}, \underline{y}_{ijk} \geq 0\}$$
$$DS_{ij} = \{u_{ijk} \in \overline{NFP}^k_{ij} | k = 1, ..., m_{ij}, \overline{y}_{ijk} \leq 0\}$$
$$DG_{ij} = \{u_{ijk} \in \overline{NFP}^k_{ij} | k = 1, ..., m_{ij}, \underline{x}_{ijk} \geq 0\}$$
$$DR_{ij} = \{u_{ijk} \in \overline{NFP}^k_{ij} | k = 1, ..., m_{ij}, \overline{x}_{ijk} \leq 0\}$$

In the example of Figure 2.14, $DN_{ij} = \{u_{ij1}, u_{ij2}, u_{ij8}\}$, $DS_{ij} = \{u_{ij4}, u_{ij5}, u_{ij6}\}$, $DE_{ij} = \{u_{ij6}, u_{ij7}, u_{ij8}\}$ and $DR_{ij} = \{u_{ij2}, u_{ij3}, u_{ij4}\}$.

Based on these four sets, the authors defined the lifted bound constraints. If the reference point of piece j is on the slices of DE_{ij} set, the distance between pieces i and

j must be at least $\underline{x}_{ijk}$. The new constraints are given by (2.29).

$$x_j \geq l_i^{left} + \sum_{k \in DE_{ij}} \underline{x}_{ijk} u_{ijk}, \quad 1 \leq i < j \leq N. \tag{2.29}$$

A similar idea is applied when the reference point of piece j is placed above the reference point of piece i, activating a slice of N_{ij} and resulting in constraints (2.30).

$$y_j \geq w_i^{top} + \sum_{k \in N_{ij}} \underline{y}_{ijk} u_{ijk}, \quad 1 \leq i < j \leq N. \tag{2.30}$$

If the reference point of piece j is on the slices of the DR_{ij} set, the minimum distance from the reference point of this piece to the rightmost point of the board is $l_i^{right} - (\overline{x}_{ij} + \underline{X}^{ij})$. Likewise, if the reference point of piece j is on the slices of DS_{ij} set, the minimum distance from the reference point of this piece to topmost point of the board must be $w_i^{top} - (\overline{y}_{ij} + \underline{Y}^{ij})$. The lifted left bound constraints are given by 2.31 and 2.32.

$$x_j \leq L - l_j^{right} - \sum_{k \in R'_{ij}} l_i^{right} - (\overline{x}_{ijk} - \underline{X}^{ij}) u_{ijk}, \quad 1 \leq i < j \leq N, \tag{2.31}$$

$$y_j \leq W - w_j^{bottom} - \sum_{k \in S'_{ij}} w_i^{bottom} - (\overline{y}_{ijk} - \underline{Y}^{ij}) u_{ijk}, \quad 1 \leq i < j \leq N. \tag{2.32}$$

The revised model of Fischetti and Luzzi (2009) with the lifted bound constraints is presented next.

$$\text{minimize} \quad L - \epsilon \textstyle\sum_{i=1}^{\mathcal{N}} (x_i + y_i) \tag{2.33}$$

subject to:

$$x_j \geq l_i^{left} + \sum_{k \in DE_{ij}} \underline{x}_{ijk} u_{ijk}, \qquad 1 \leq i < j \leq N, \tag{2.34}$$

$$x_j \leq L - l_j^{right} - \sum_{k \in DR'_{ij}} l_i^{right} - (\overline{x}_{ijk} - \underline{X}^{ij}) u_{ijk}, \qquad 1 \leq i < j \leq N, \tag{2.35}$$

$$y_j \geq w_i^{top} + \sum_{k \in DN_{ij}} \underline{y}_{ijk} u_{ijk}, \qquad 1 \leq i < j \leq N, \tag{2.36}$$

$$y_j \leq W - w_j^{top} - \sum_{k \in DS'_{ij}} w_i^{top} - (\overline{y}_{ijk} - \underline{Y}^{ij}) u_{ijk}, \qquad 1 \leq i < j \leq N, \tag{2.37}$$

$$\alpha_{ij}^{kf}(x_j - x_i) + \beta_{ij}^{kf}(y_j - y_i) \leq \sum_{h=1}^{m_{ij}} \phi_{ij}^{kfh} u_{ijh},$$
$$1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, f = 1, ..., s_{ij}^k, \tag{2.38}$$

$$\sum_{k=1}^{m_{ij}} u_{ijk} \geq 1, \qquad 1 \leq i < j \leq \mathcal{N}, \tag{2.39}$$

$$u_{ijk} \in \{0, 1\}, \quad 1 \leq i < j \leq \mathcal{N}, k = 1, ..., m_{ij}, \tag{2.40}$$

$$x_i, y_i \geq 0, \qquad i = 1, ..., N. \tag{2.41}$$

The authors also developed a branch and cut method that uses specialized branching strategies for the irregular strip packing problem. The lower bounds are calculated by a mixed-integer programming model based on the 1-contiguous bin packing problem. The results of this branch and cut method outperformed the ones found by the commercial solver.

Although the authors defined clearly how to generate the slices for the Fischetti and Luzzi (2009) model, identify holes and non convexities of the nofit polygon can still be a difficult task. Furthermore, obtain the nofit polygons of pieces with holes or narrow entries is tough, making difficult to generate an instance of this problem.

2.2.5 The dotted board model

The mathematical models presented in Sections 2.2.2-2.2.4 consider the continuous placement of pieces over the board, i.e., each piece can be allocated in any feasible position of the board. Toledo et al. (2013) proposed a model where the reference point of the pieces can only be positioned at the dots of a given grid on the board. The grid used was regular, i.e., the horizontal distance between any two different points of the grid must be multiple of constant g_x. Likewise, the vertical distance between any two different points of the grid must be multiple of constant g_y. The authors emphasized that these constants must be carefully defined because they impact directly on the number of variables and constraints of the model and also on the solution quality. An

example of a grid is illustrated in Figure 2.15.

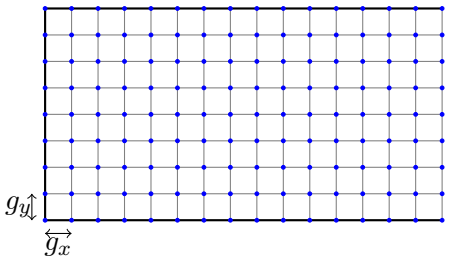


Figure 2.15: Example of grid used in Toledo et al.'s model where $g_x = g_y$.

Consider the binary variable δ_t^d which is equal to 1, if the reference point of the piece of type $t \in \mathcal{T}$ is placed at dot $d \in \mathcal{D}$, and 0 otherwise. Note that, unlike the previous models, the dotted board model is based on types of pieces and not on each particular piece of a certain type. Therefore, the number of pieces of the same type to be packed does not increase the number of variables in the model. The feasible placement positions for each piece of type $t \in \mathcal{T}$ are the dots $d \in \mathcal{D}$ belonging to IFP_t, which defines the $DIFP_t$ set. Constraints (2.42) ensure that demand q_t for each piece of type t is met.

$$\sum_{d \in DIFP_t} \delta_t^d = q_t, \qquad t \in \mathcal{T}. \tag{2.42}$$

An important condition for a solution to be feasible is the non overlap between pieces. Consider that for each pair of pieces of types t and u, with the reference point of t positioned at dot d, the intersection points are represented by the dots $DIFP_u$ inside NFP_{tu}, defining the $DNFP_{tu}^d$ set. If reference point of piece t is at dot d and the reference point of piece u is placed at a dot $d' \in DNFP_{tu}^d$ then the pieces overlap. Constraints (2.43) ensure that the pieces do not overlap.

$$\delta_t^d + \delta_u^{d'} \leq 1, \qquad d' \in DNFP_{t,u}^d, t, u \in \mathcal{T}, d \in DIFP_t. \tag{2.43}$$

The objective is to minimize the used board length L. Consider that the reference point of piece t is placed at dot d with coordinates (d_x, d_y). Constraints (2.44) ensure that L is equal or greater to the used board length.

$$(d_x + l_t^{right}) \times \delta_t^d \leq L, \qquad t \in \mathcal{T}, d \in DIFP_t. \tag{2.44}$$

The dotted board model is presented in the following.

$$\text{minimize} \quad L \tag{2.45}$$

subject to:

$$(d_x + l_t^{right}) \times \delta_t^d \leq \quad L, \quad d \in \mathit{DIFP}_t, t \in \mathcal{T}, \tag{2.46}$$

$$\delta_u^{d'} + \delta_t^d \leq \quad 1, \quad d \in \mathit{DNFP}_{t,u}^d, t, u \in \mathcal{T}, d \in \mathit{DIFP}_t, \tag{2.47}$$

$$\sum_{d \in \mathit{DLFP}_t} \delta_t^d = \quad q_t, \quad t \in \mathcal{T}, \tag{2.48}$$

$$\delta_t^d \in \quad \{0,1\}, \quad d \in \mathit{DIFP}_t, t \in \mathcal{T}, \tag{2.49}$$

$$L \geq \quad 0. \tag{2.50}$$

Note that the optimality of this model is associated with the used grid, i.e., a refined grid allows the pieces to fit better leading to solutions of a better quality. On the other hand, the grid should not be too refined because the number of variables is proportional to the number of dots on the board.

Using this model, the optimal solution for instances with up to 56 pieces of two different types and instances with up to 21 pieces of seven different types were found.

Recently, Rodrigues (2015) studied the non overlap constraints (2.47) and proposed a reformulation of these constraints. The new constraints are based on clique covering and, with this reformulation, the number of constraints of the problem is reduced and the bounds of the model are improved.

2.2.6 The semi-continuous model

Aiming to merge the compactness of the linear models and the flexibility of the dotted board model, Leao et al. (2016) proposed a semi-continuous model. The pieces can be placed inside of the board along the x-axis and only on discretized positions on y-axis, i.e., the reference point of the pieces can be placed only on horizontal stripe lines in the board. The stripes are parallel and they are equally distributed, i.e., the distance among the stripes on the y-axis is multiple of a constant g_y. This distance must be carefully defined since the solution quality depends on it. Figure 2.16 illustrates a board discretized by stripes used in the semi-continuous model.

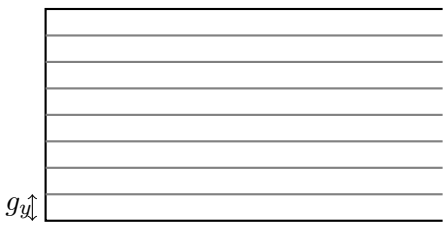


Figure 2.16: Example of board discretized by lines used in Leao et al.'s model.

In this model, the locus of piece i is defined by the continuous variable x_i and the binary variable $\vartheta_i^{p_i}$ which is 1 if the piece i is placed on stripe p_i and 0 otherwise, for

$i = 1, ..., \mathcal{N}$, $p_i = 0, ..., W$. Using these variables, the authors ensure that the pieces are entirely inside the board using constraint (2.3) and restricting p_i in the interval $[w_i^{top}, W - w_i^{bottom}]$.

To ensure that the pieces do not overlap, the authors used the nofit polygon. Consider that NFP_{ij} is divided in horizontal stripes, with the lowest stripe n_{ij}^{min} and the highest stripe n_{ij}^{max}. Consider also a_{ij}^{oc} (a'^{oc}_{ij}) the x coordinate of the leftmost (rightmost) point of stripe o related to concavity c of NFP_{ij}. Figure 2.17 illustrates the division of the nofit polygon by horizontal stripes and parameters n_{ij}^{min}, n_{ij}^{max}, a_{ij}^{oc} and a'^{oc}_{ij}.

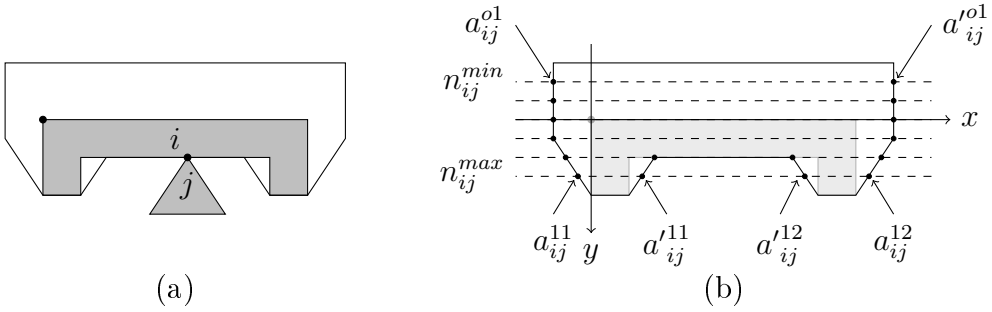


Figure 2.17: (a) shows pieces i and j and the resulting NFP_{ij}. The division of NFP_{ij} in horizontal stripes and the discretization parameters are presented in (b).

Consider C_{ij}^{o} the number of concavities of stripe o of the NFP_{ij}. The variable ϖ_{ij}^{c} which is 1 if the reference point of piece i is placed on the right side of concavity c of piece j and 0 otherwise, $c = 1, ..., C_{ij}^{p_j - p_i}$. Note that when the reference points of pieces i and j are respectively placed on p_i and p_j board stripes, $p_j - p_i$ is the vertical distance between pieces i and j reference points.

To avoid the overlap among pieces the authors presented inequalities (2.51) and (2.52).

$$\begin{aligned} x_i \leq x_j - a'^{(p_i - p_j)c}_{ij} + \varpi_{ij}^{c} M + (1 - \vartheta_i^{p_i})M + (1 - \vartheta_j^{p_j})M, \quad 1 \leq i < j \leq \mathcal{N}, \\ c = 1, ..., C_{ij}^{p_j - p_i}, w_i^{top} \leq p_i \leq W - w_i^{bottom}, \\ w_j^{top} \leq p_j \leq W - w_j^{bottom}, n_{ij}^{max} \leq p_j - p_i \leq n_{ij}^{max}, \end{aligned} \tag{2.51}$$

$$\begin{aligned} x_i \geq x_j - a_{ij}^{(p_i - p_j)c} + (1 - \varpi_{ij}^{c}) M + (1 - \vartheta_i^{p_i})M + (1 - \vartheta_j^{p_j})M, \quad 1 \leq i < j \leq \mathcal{N}, \\ c = 1, ..., C_{ij}^{p_j - p_i}, w_i^{top} \leq p_i \leq W - w_i^{bottom}, \\ w_j^{top} \leq p_j \leq W - w_j^{bottom}, n_{ij}^{max} \leq p_j - p_i \leq n_{ij}^{max}. \end{aligned} \tag{2.52}$$

In inequality (2.51) if the piece i is on stripe p_i, piece j is on stripe p_j and the reference point of piece i is on the left of concavity c then x_j must be greater or equal to $x_i + a'^{(p_i-p_j)c}_{ij}$. The inequality (2.52) ensures that if the pieces i and j are receptively on stripes p_i and p_j and the reference point of piece i is on the right side of concavity c then x_j must be smaller than or equal to $x_i + a'^{(p_i-p_j)c}_{ij}$.

To ensure that all the pieces are placed on the board, the authors impose constraints (2.53) which assign the piece reference point of each to a line.

$$\sum_{p=w_i^{top}}^{W-w_i^{bottom}} \vartheta_i^{p_i} = 1, \quad 1 \leq i \leq \mathcal{N}. \tag{2.53}$$

The semi-continuous model proposed by Leao et al. (2016) is given as follows.

$$\text{minimize} \quad L \tag{2.54}$$

subject to:

$$L - x_i \geq l_i^{right}, \qquad i = 1, ..., \mathcal{N}, \tag{2.55}$$

$$x_i \geq l_i^{left}, \qquad i = 1, ..., \mathcal{N}, \tag{2.56}$$

$$x_i \leq x_j - a'^{(p_i-p_j)c}_{ij} + \varpi_{ij}^c M + (1 - \vartheta_i^{p_i})M + (1 - \vartheta_j^{p_j})M, \quad 1 \leq i < j \leq \mathcal{N}, \\ c = 1, ..., C_{ij}^{p_j-p_i}, w_i^{top} \leq p_i \leq W - w_i^{bottom}, \\ w_j^{top} \leq p_j \leq W - w_j^{bottom}, n_{ij}^{max} \leq p_j - p_i \leq n_{ij}^{max}, \tag{2.57}$$

$$x_i \geq x_j - a_{ij}^{(p_i-p_j)c} + (1 - \varpi_{ij}^c)M + (1 - \vartheta_i^{p_i})M + (1 - \vartheta_j^{p_j})M, \quad 1 \leq i < j \leq \mathcal{N}, \\ c = 1, ..., C_{ij}^{p_j-p_i}, w_i^{top} \leq p_i \leq W - w_i^{bottom}, \\ w_j^{top} \leq p_j \leq W - w_j^{bottom}, n_{ij}^{max} \leq p_j - p_i \leq n_{ij}^{max}, \tag{2.58}$$

$$\sum_{p=w_i^{top}}^{W-w_i^{bottom}} \vartheta_i^{p_i} = 1, \qquad i = 1, ..., \mathcal{N}, \tag{2.59}$$

$$\vartheta_i^{p_i} \in \{0, 1\}, \qquad i = 1, ..., \mathcal{N}, w_i^{top} \leq p_i \leq W - w_i^{bottom}, \tag{2.60}$$

$$\varpi_{ij}^c \in \{0, 1\}, \qquad 1 \leq i < j \leq \mathcal{N}, \tag{2.61}$$

$$x_i \in \mathbb{R}^+, \qquad i = 1, ..., \mathcal{N}. \tag{2.62}$$

As in the dotted board model, the optimality of this model depends on how sparse are the stripe lines that represent the board. In addition, the number of variables and constraints depends on this discretization. Note that the discretization has less impact on the semi-discrete model than on the dotted board model since it discretizes only

the y-coordinates where the pieces can be placed.

2.3 Heuristic methods

In contrast to the number of exact approaches, many heuristics were proposed to solve the irregular strip packing problem. A review of heuristics was presented in Bennell and Oliveira (2009). These heuristics can be classified into constructive and improvement heuristics.

Constructive heuristics aim to build a solution to the problem using a particular strategy. The most popular strategy is the bottom-left heuristic introduced by Art (1966). This heuristic is performed in steps and in each step one piece is placed on the layout. Every time a piece is placed, it is moved horizontally from the top right corner of the board to its leftmost position (Figure 2.18(a)). Then, the piece is moved vertically to the bottom until it can be moved to the left again (Figure 2.18(b)). The process ends when the piece can not be moved to the left or downwards (Figure 2.18(c) and 2.18(d)). Figure 2.18(e) illustrates the place where piece 5 is inserted in the layout using the bottom-left heuristic.

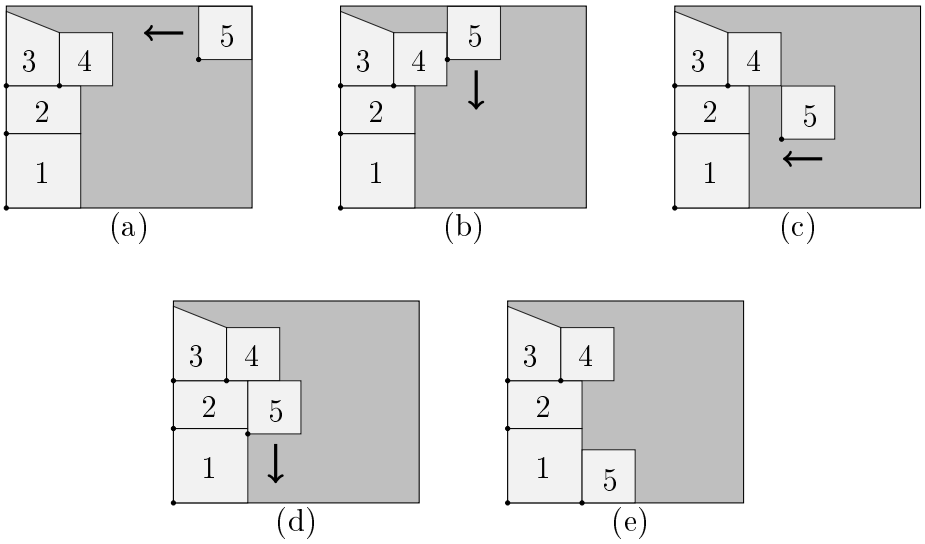


Figure 2.18: Placing a piece in the board using the bottom-left heuristic.

The solution found by the bottom-left heuristic depends on the sequence of the pieces. Similar strategies can be derived from the bottom-left heuristic by moving pieces to different directions.

Some authors investigated sequences for the placement of the pieces in the bottom-left heuristic. Oliveira et al. (2000) presented five rules to arrange the pieces to be inserted in the board. Based on these rules, 126 variations of the bottom-left heuristic were derived. Dowsland et al. (2002) proposed nine rules for ordering the pieces to

perform the heuristic and an efficient implementation of the bottom-left heuristic is described. A search over the sequence of input pieces in the bottom-left heuristic was performed by Gomes and Oliveira (2002). In each iteration, the authors changed the position of two pieces in the input sequence and then performed the bottom-left heuristic again. Albano and Sapuppo (1980) proposed a heuristic similar to the bottom-left where each piece is placed in the position that least increases the length of the layout.

Given a feasible solution, improvement heuristics search for solutions with better quality. A solution method based on simulated annealing was presented by Gomes and Oliveira (2006). Specifically, the bottom-left heuristic is used to find the initial layout. Then simulated annealing is used to perform a search over the sequence of pieces. To obtain better layouts, the authors used linear models for compaction (Li and Milenkovic, 1995) obtaining a local optimum.

Jakobs (1996) presented a genetic algorithm to search over the input sequence of the pieces in the bottom-left heuristic. Another genetic algorithm that also searches over the input sequence was proposed by Babu and Babu (2001). The procedure can handle predefined rotations of the pieces.

By also using bottom-left heuristic to find the initial solution, Egeblad et al. (2007) proposed a fast neighbourhood search to solve the problem. The method used local search to move the pieces horizontally and vertically over the layout aiming to reduce overlaps. A guided local search was used to escape the local minimum. If a solution with no overlap is found, the layout length is reduced and the process is applied again. The heuristic was extended for three-dimensional packing problems.

Burke et al. (2006) applied tabu search and hill climbing heuristics over the input sequence of the bottom-left heuristic. In their method, the pieces can be packed in holes formed by other pieces and the polygons can also be represented by arcs.

Rather complex and sophisticated heuristic methods have been developed and recently published. Umetani et al. (2009) presented an overlap minimization algorithm based on translations of pieces on vertical and horizontal directions. This algorithm is incorporated into a guided local search in order to solve the strip packing problem. Imamichi et al. (2009) proposed an iterated local search heuristic to solve the problem. The local search swaps the position of a pair of pieces in the solution and a non linear programming separation algorithm ensures the non-overlap between the pieces. An extended local search heuristic to solve the problem was proposed by Leung et al. (2012). Two neighbourhoods are used to change the piece positions during the local search and the feasibility of a solution is reached by non-linear programming separation and compaction models. Sato et al. (2012) proposed two constructive heuristics using the concept of collision free regions and identifying pieces that fit well. These

heuristics are combined with a simulated annealing algorithm in order to obtain better solutions. A compaction model is used to reduce the length of the solutions obtained in each iteration of the algorithm. Elkeran (2013) proposed a heuristic where first the pieces are clustered in pairs, then a guided cuckoo search heuristic is used to pack the pieces into the board. This heuristic reached the best results in the literature for the irregular strip packing problem.

2.4 Solution methods for other cutting and packing problem variants

Heuristics have been proposed for other cutting and packing problem variants. To solve the irregular two-dimensional knapsack problem, Dalalah et al. (2014) proposed a heuristic where the pieces are placed inside of the board by an iterative process. First the pieces are ordered by area, then they are placed in the board with a rule based on their area and on the area of the convex hull of the solution layout. Alves et al. (2012) proposed several placement heuristics trying to quickly obtain good quality solutions for the problem. Their placement method uses information of the contact region among pieces in order to obtain a compact layout. The authors use this method to solve the leather cutting problem of automotive industry. A genetic algorithm to solve the problem was presented by Crispin et al. (2005). The algorithm was developed to solve the problem of cutting shoe components. At each iteration of their algorithm, the placement heuristic is based on the contact region among the pieces and the genetic algorithm is responsible to improve the solution quality.

Valle et al. (2012) proposed heuristics for the irregular binary knapsack problem and the irregular unconstrained knapsack problem. To solve the irregular binary knapsack problem, a GRASP heuristic is combined with a constructive heuristic in order to obtain solutions for the problem. For the irregular unconstrained knapsack problem the pieces are first clustered in small rectangles that are packed in the board using an exact method that solves the rectangular knapsack problem with guillotine constraints. Using this last heuristic, the authors also proposed a column generation procedure to solve the irregular cutting stock problem.

Song and Bennell (2014) proposed the first method to solve the irregular cutting stock problem. The authors consider that all boards have the same size. First the problems are decomposed into master problem and sub-problem, then a column generation procedure is used to solve it. The sub-problem defines a cutting pattern to be used in cutting stock problem which represents a column in the master problem. These columns are heuristically generated and thus the procedure is a heuristic.

To solve the irregular two-dimensional bin packing problem considering a single board size, Lamousin and Waggenspack (1997) proposed a constructive heuristic. The heuristic tries to mimic the work of a human, which consists of placing one piece on the board at each iteration. The piece chosen to be placed is the one that generates less waste. Okano (2002) proposed a heuristic to solve the problem by approximating the bins and pieces by lines. Using this representation, the authors proposed a placement heuristic to build the solution. López-Camacho et al. (2013) proposed a constructive heuristic to solve this problem where the order by which the pieces will be placed is first defined, then a placement heuristic is used to build the solution. The authors use seven criteria to define the piece sequence. Two placement heuristics were used: an adaptation of the bottom-left heuristic, and a constructive heuristic proposed by the authors. A heuristic to solve this problem considering that the pieces are convex polygons were proposed in Terashima-Marín et al. (2010). The authors proposed a genetic algorithm to decide the piece sequences and several constructive heuristics to build the solution layout.

To solve irregular bin packing problem where the board is irregular with defects, Babu and Babu (2001) proposed a genetic algorithm. In their method, the genetic algorithm chose a sequence of pieces and specific angles to rotate them. This sequence is used to place the pieces on the board using the same strategy of the bottom-left heuristic and considering the irregular parts and board defects. Baldacci et al. (2014) proposed a heuristic to solve the irregular bin packing problem with an irregular board with defects and quality regions where only a subset of pieces can be placed. In their method, solutions for each single irregular board are built using different constructive heuristics, that generate a set of cutting patterns. This algorithm to generate cutting patterns is embedded in a heuristic for solving the bin packing problem.

2.5 Research gaps

After this review over different aspects of the two-dimensional irregular cutting and packing problems, some gaps in the literature can be pointed:

1. The mixed-integer programming models for the strip packing problem demand geometric structures that generally are tough to be obtained to be formulated. It suggests as research direction the proposal of new models considering simpler structures to be built and with competitive results compared with the literature.

2. Looking to the geometry, there is no structure to handle the case where discrete placement for the pieces in the board is considered and the nofit polygon is

used to evaluate the overlap. Such structure could make easy and flexible the development methods based on this geometry.

3. Among the proposed heuristic methods, only a few combine exact approaches and heuristics. In addition, there is no method in the literature that combines the recently mixed-integer programming models with heuristics strategies.

4. Despite the number of mathematical models and heuristics for the irregular strip packing problem, for the other variants of the irregular cutting and packing problems there are only a few heuristics and no exact method. Furthermore, the development of constraint programming methods to solve cutting and packing problems seem promising and not fully explored.

The next chapters of this thesis are dedicate to investigate these gaps.

Chapter 3

New mixed-integer programming models to the irregular strip packing problem[1]

The geometric representation of the pieces and the board is the main problem in the development of models and heuristics for the irregular strip packing problem. The models in the literature handle with pieces and board geometry by discretising the pieces and the board or by not considering non-convex pieces with holes, pushing the models away from the real-world needs. Part of the models' limitations are due to the algorithms available to build the nofit polygons (Bennell and Oliveira, 2008). While the construction of the nofit polygon of convex pieces is just a matter of ordering the edge angles, for non-convex pieces, this process is harder. Moreover, pieces with narrow crevices, consecutive edges with similar slopes and holes lead to complex and numerically unstable algorithms.

In this chapter, mixed-integer programming models to solve the irregular strip packing problem are proposed. The direct trigonometry model (DTM) is built using only geometric information about the pieces and the board, that is, more advanced structures as nofit polygons are avoided making the model implementation easier. The nofit polygon covering model ($NFP-CM$) uses the nofit polygon to avoid overlaps among pieces. Both approaches are robust in terms of the geometry of the pieces they can address, considering convex and non-convex polygons with or without holes. They are also simpler to implement than the previous literature models. This simplicity allowed to consider for the first time a variant of the linear models that deals with piece rotations. Computational experiments with benchmark instances show that $NFP-CM$ outperforms both DTM and the best mixed-integer programming models from the literature. In addition, new real-world based instances with more complex geometries are

[1] This chapter is strongly based on Cherri et al. (2016)

proposed and used to verify the robustness of the new models.

The remainder of this chapter is strongly based on the paper "Mixed-integer linear programming models for the irregular strip packing problem" (Cherri et al., 2016) and is organized as follows. Section 3.1 presents the specific geometric definitions used along the chapter. In Section 3.2, two new mathematical programming models are proposed, including some variants based on valid inequalities and variable reduction strategies and, in Section 3.3, computational experiments on benchmark instances from the literature are presented. The performance of the models is evaluated by comparing their resuts with the best results previously published for exact methods. Furthermore, computational experiments including piece rotations and pieces with holes are presented.

3.1 Description and geometric definitions of the irregular strip packing problems

This section presents the geometric definitions and the notations that will be used along this chapter. Some basic definitions have already appeared in chapter 2 and are presented with more details.

3.1.1 Problem definition

The two-dimensional irregular strip packing problem can be formally defined by a set of pieces of m distinct types, each one described by a polygon P_i, $i = 1, \ldots, m$, that have to be placed, in an amount of d_i units, on a large rectangle board characterized by a width W and a length L. For the sake of legibility and comparability the models will be presented and tested considering that piece rotations are not allowed, as in all previous exact approaches to this problem. However, an extension of the models allowing different orientations for each piece will be proposed and discussed in Section 3.2.3. As the length L is not fixed, the board can be considered as having "infinite" length and the problem's goal is to minimise L, i.e. the length of the board that is necessary to cut all demanded pieces. In practice, this corresponds to minimising the amount of raw-material used to satisfy a given order of irregularly shaped pieces.

The problem constraints are of three types:

1. each piece type i has to be cut in the demanded quantities d_i, in a total of $N = \sum_{i=1}^{m} d_i$ pieces;

2. pieces must not overlap, i.e. $\text{int}(P_j) \bigcap \text{int}(P_l) = \emptyset$, $\forall j, l = 1, \ldots, N; j \neq l$;

3. pieces must be completely contained inside the board, i.e.:

$$\text{int}(P_j) \bigcap \text{int}(\text{rect}(L, W)) = \text{int}(P_j),\ \forall j = 1, \ldots, N;$$

where int() stands for the topological interior and rect(L, W) for the rectangle of length L and width W.

3.1.2 Basic concepts

In the new models, each piece i is not described by a single polygon but it is decomposed and represented by a set of convex polygons, the parts of piece i, that may or may not overlap, allowing to represent convex and non-convex pieces, as well as pieces with holes.

Each part of piece i is represented by a list of (clockwise) ordered points in the plane that represents its vertices. One of the vertices of one of the parts of piece i is chosen to be its reference point, denoted by (x_i, y_i). l_i^{left} and l_i^{right} are defined as the distances in the x-axis from x_i to the leftmost and rightmost vertex in piece i, respectively. Analogously, w_i^{top} and w_i^{bottom} are defined as the distance in the y-axis from y_i to the vertices of piece i that are closest and farthest from the origin, respectively. Figure 3.1 illustrates a piece decomposed in two convex polygons, its reference point and the corresponding measures for l_i^{left}, l_i^{right}, w_i^{top} and w_i^{bottom} as well as the coordinate system and its origin.

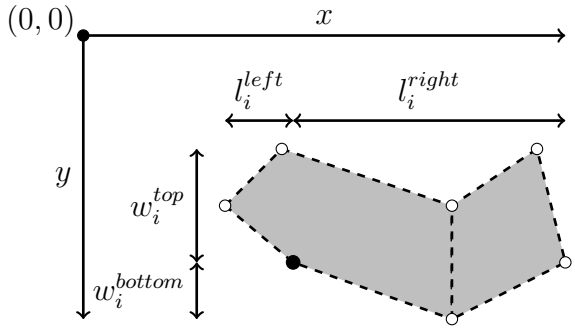


Figure 3.1: Decomposition of a piece in two convex polygons and distances in the x and y-axis from the positioning point to the borders.

3.1.3 Detecting overlaps between two pieces

To ensure that pieces i and j do not overlap, each one of the parts of piece i must not overlap each one of the parts of piece j and vice-versa. Therefore, it is only necessary to detect if a part of a piece overlaps a part of another piece. As stated in Bennell and Oliveira (2008) two methods for the detection of the overlaps between two pieces are the D-function, that uses direct trigonometry and the nofit polygon.

The D-function

Given an oriented edge $\overline{ab}$, where $a = (a_x, a_y)$ and $b = (b_x, b_y)$ and a point $r = (r_x, r_y)$, the D-function gives the relative position of the point with respect to the oriented edge and is defined by equation (3.1).

$$D_{abr} = (a_x - b_x)(a_y - r_y) - (a_y - b_y)(a_x - r_x). \tag{3.1}$$

Figure 3.2 shows the different values that D_{abr} can have and its implications. If $D_{abr} = 0$ (Figure 3.2a), point r is over the line defined by $\overline{ab}$; if $D_{abr} < 0$ (Figure 3.2b), point r is on the right side of line $\overline{ab}$; and if $D_{abr} > 0$ (Figure 3.2c), point r is on the left side of line $\overline{ab}$. This particular relationship between the sign of the D-function and the right-left position of point r is dependent on where the origin of the coordinate system is defined (Figure 3.1). In our implementation, as other authors (in the literature), for historical reasons we consider that the origin of the coordinate system is on the left-top corner (as it happens with screens and pixel numbering), i.e. x-coordinates grow to the right and y-coordinates grow downwards. This has an impact on the interpretation of the result of the D-function, e.g. a negative value of the D-function means that the point is on the right side of the edge if and only if the top-left origin for the coordinate system is considered.

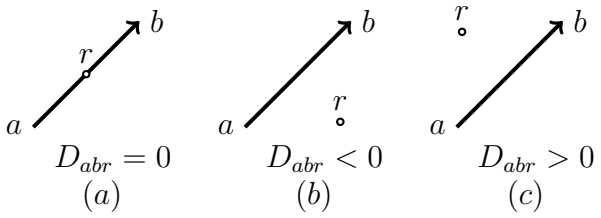


Figure 3.2: Values of D_{abr} and its implications.

The D-function is used to analise the intersection of two polygons that represent two pieces by evaluating the relative position between the edges of one polygon and the vertices of the other one. If pieces i and j are convex, consider K_i to be the set of edges of piece i. Given an edge $e \in K_i$, the D-function can be used to verify if the vertices of piece j are on the right side of e. If it is true for at least one edge $e \in K_i$, then piece i does not overlap piece j.

The nofit polygon

The nofit polygon of two pieces i and j, denoted by NFP_{ij}, is the locus of all the points where the reference point of piece j cannot be placed without overlapping piece i. Since

the pieces are defined as sets of convex parts, the NFP_{ij} can be defined as the set of all nofit polygons of parts of i and parts of j, that is, each part of NFP_{ij} is the nofit polygon of p and q, where $p \in P_i$ and $q \in P_j$. Note that as all the parts of pieces i and j are convex polygons all parts of NFP_{ij} are also convex polygons and they might overlap.

To build the nofit polygon of convex pieces, the Cuninghame-Green (1989) algorithm can be used as exemplified in Figure 3.3. In the method, given two pieces i and j, the NFP_{ij} can be built ordering the edges of the fixed polygon (that represented the piece i) on clockwise orientation and the orbital polygon (that represent piece j) edges on counter-clockwise orientation (Figure 3.3a). Then, the edges are translated in order to start at the same point (Figure 3.3b). Finally, the edges are concatenated in a increasing angle order (Figure 3.3c).

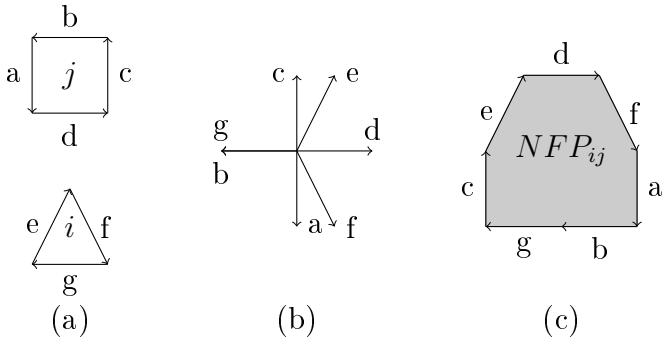


Figure 3.3: Building the nofit polygon using Cuninghame-Green (1989) method.

By using the NFP_{ij} it is possible to verify if piece i does not overlap piece j just by checking if the reference point of j is on the border or outside the NFP_{ij}.

3.2 Mathematical models

A mathematical formulation for an irregular cutting problem must ensure that the pieces are entirely inside the board and they do not overlap. The first condition can be easily satisfied by using the pieces geometric information presented in Figure 3.1. To guarantee that the pieces do not overlap two different approaches are proposed. The first one uses only the information provided by the geometry of the pieces leading to a *Direct Trigonometry Model* presented in Subsection 3.2.1. The second approach is based on the *NFP* covering structure presented in Section 3.1, resulting in the *NFP Covering Model* presented in Subsection 3.2.2.

Some valid inequalities have been developed for each model and are described in the respective section. Subsection 3.2.4 presents the bounds used in the models and includes a short discussion about their importance.

3.2.1 Direct Trigonometry Model - *DTM*

This section presents a mixed-integer programming model to solve the irregular strip packing problem based only on direct trigonometry to infer the set of feasible solutions. Some variable reductions and valid inequalities are introduced in order to eliminate redundant constraints and part of the symmetric solutions.

Basic model

The first set of constraints must ensure that the pieces do not overlap. Consider two pieces i and j. Piece i does not overlap piece j if either all the vertices of i are at the right side of an oriented edge of piece j or the vertices of piece j are at the right side of an oriented edge of piece i. Note that two pieces composed by several parts do not overlap if all the pairs of parts of these pieces do not overlap. The D-function in equation (3.1) is used to build these constraints.

Consider that the points $a^k = (a_x^k, a_y^k)$ and $b^k = (b_x^k, b_y^k)$ represent respectively the initial and final vertices of edge $k \in K_{ip}$, where K_{ip} is the set of edges of part p of piece i. Consider also g_x^{irqj} and g_y^{irqj} to be the horizontal and vertical distances between (x_i, y_i), the positioning point of piece i, and the vertex r of part q of piece j. Using the D-function (3.1) we can write inequality (3.2).

$$D_{abg} \;=\; (a_x^k - b_x^k)(a_y^k - g_y^{irqj}) - (a_y^k - b_y^k)(a_x^k - g_x^{irqj}) \leq 0. \tag{3.2}$$

If this inequality is satisfied, the pieces i and j are either separated or touching. Moreover the distance between each vertex of part q of piece j and the reference point of piece j must be taken into account. Consider g_x^{jrqj} and g_y^{jrqj} the vertical and horizontal distances between the reference point of piece j (x_j, y_j) and vertex r of part q of piece j: $g_x^{jrqj} = x_r - x_j$ and $g_y^{jrqj} = y_r - y_j$. Then, $g_x^{irqj} = x_j + g_x^{jrqj} - x_i$ and $g_y^{irqj} = y_j + g_y^{jrqj} - y_i$. The distances g_x^{jrqj}, g_y^{jrqj}, g_x^{irqj} and g_y^{irqj} are illustrated in Figure 3.4. Using this information on inequality (3.2), inequality (3.3) is obtained.

$$\begin{aligned}(a_x^k - b_x^k)(a_y^k + y_i - y_j - g_y^{jrqj}) - (a_y^k - b_y^k)(a_x^k + x_i - x_j - g_x^{jrqj}) \;&\leq\; 0 \Rightarrow \\ C_{ij}^{pqkr} + (a_x^k - b_x^k)(y_i - y_j) + (a_y^k - b_y^k)(x_i - x_j) \;&\leq\; 0,\end{aligned} \tag{3.3}$$

where C_{ij}^{pqkr} is defined as $(a_x^k - b_x^k)(a_y^k - g_y^{jrqj}) - (a_y^k - b_y^k)(a_x^k - g_x^{jrqj})$.

Note that it is not necessary to create constraints invoking that all the vertices of q (or p) are on the right side of one line of p (or q). Indeed, given a set of points and a specific line, constraint (3.3) associated with them differs only on the constant

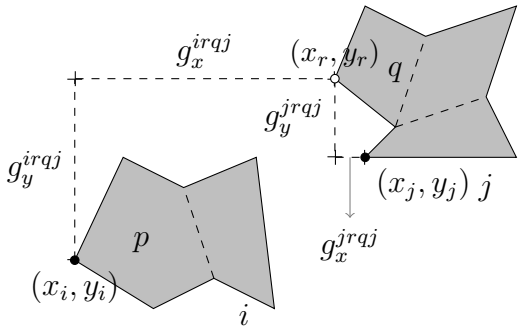


Figure 3.4: Obtaining the constants g_x^{jrqj}, g_y^{jrqj}, g_x^{irqj} and g_y^{irqj}.

C_{ij}^{pqkr}. By this fact, only the constraint with the largest constant C_{ij}^{pqkr} needs to be inserted in the model. This rule reduces substantially the number of constrains on the model, leading to a faster search. Then the following constraints prevent two parts from overlapping.

$$C_{ij}^{pqk} + (a_x^k - b_x^k)(y_i - y_j) + (a_y^k - b_y^k)(x_i - x_j) \quad \leq \quad 0, \tag{3.4}$$

where C_{ij}^{pqk} is defined as $\max_r\{C_{ij}^{pqkr}\}$.

However, only one inequality needs to be satisfied for each part p to ensure that the pieces do not overlap. Consider the variable v_{ij}^{pqk} that is 1 if the inequality for edge k of part p of piece i is satisfied with respect to part q of piece j, and 0 otherwise. These line constraints (3.4) are formulated as follows:

$$\begin{aligned} C_{ij}^{pqk} + (a_x^k - b_x^k)(y_i - y_j) + (a_y^k - b_y^k)(x_i - x_j) \leq (1 - v_{ij}^{pqk})M_{ij}^{pqk}, \\ i = 1, ..., N,\ j = 1, ..., N,\ i \neq j,\ p \in P_i,\ q \in P_j,\ k \in K_{ip}, \end{aligned}$$

where M_{ij}^{pqk} is a number large enough to make the inequality always valid when v_{ij}^{pqk} is equal to zero. This number is estimated in Section 3.2.1.

As it is not acceptable that the pieces overlap, a constraint to ensure that exactly one constraint related to a given pair of parts of different pieces is satisfied must be created. It is important to reinforce that it is guaranteed that if all the parts of different pieces do not overlap, the pieces do not overlap. The following constraint ensures that one inequality of edge k, part p of piece i or of part q of piece j is satisfied. Clearly, feasible solutions with more than one edge active for the same pair of parts can exist, but these solutions are not precluded by activating only one of these edges.

$$\sum_{k \in K_{ip}} v_{ij}^{pqk} + \sum_{k \in K_{jq}} v_{ji}^{qpk} = 1, \quad 1 \leq i < j \leq N,\ p \in P_i,\ q \in P_j.$$

The constraint that the pieces must be entirely contained inside the board, can be ensured by the innerfit polygon represented by the following constraints.

$$\begin{aligned} l_i^{left} &\leq x_i, \quad i = 1, ..., N, \\ w_i^{top} \leq y_i &\leq W - w_i^{bottom}, \quad i = 1, ..., N. \end{aligned}$$

The next constraints are imposed to ensure that the used length L of the board will be minimised.

$$x_i \ \leq L - l_i^{right},\ i = 1, ..., N.$$

The Direct Trigonometry Model is presented in (3.5)-(3.11).

$$\min L \tag{3.5}$$

$$\text{s.t. } l_i^{left} \leq x_i \leq L - l_i^{right}, \qquad i = 1, ..., N, \tag{3.6}$$

$$w_i^{top} \leq y_i \leq W - w_i^{bottom}, \qquad i = 1, ..., N, \tag{3.7}$$

$$\begin{aligned} &C_{ij}^{pqk} + (a_x^k - b_x^k)(y_i - y_j) + \\ &(a_y^k - b_y^k)(x_i - x_j) \leq (1 - v_{ij}^{pqk}) M_{ij}^{pqk}, \qquad i, j = 1, ..., N,\ i \neq j, \\ &\qquad\qquad k \in K_{ip}, \\ &\qquad\qquad p \in P_i,\ q \in P_j, \end{aligned} \tag{3.8}$$

$$\sum_{k \in K_{ip}} v_{ij}^{pqk} + \sum_{k \in K_{jq}} v_{ji}^{qpk} = 1, \qquad 1 \leq i < j \leq N,$$

$$p \in P_i,\ q \in P_j, \tag{3.9}$$

$$(x_i, y_i) \in \mathcal{R}^2, \qquad i = 1, ..., N, \tag{3.10}$$

$$\begin{aligned} &v_{ij}^{pqk} \in \{0, 1\}, \qquad i, j = 1, ..., N,\ i \neq j, \\ &\qquad\qquad k \in K_{ip},\ p \in P_i,\ q \in P_j. \end{aligned} \tag{3.11}$$

This model does not need special geometric structures, as the nofit polygons, to be built. This characteristic is interesting since by using these simpler structures it can be easier, compared to more complex models, to add new constraints to the model or

to change the existing ones.

Eliminating redundant constraints and variables

It is possible to eliminate redundant constraints by finding the sets of points and lines that lead to the same solution space. Consider part p of piece i and part q of piece j as in Figure 3.5. Notice that the same solutions can be reached either if the constraint related to line α is active or if the constraint related to line β is active. In other words, if part p of piece i and part q of piece j have lines with the same orientation and opposite directions the line β can be removed from K_{jq} set (or α can be removed from K_{ip} set). Therefore, the binary variable $v_{ji}^{qp\beta}$ (or $v_{ij}^{pq\alpha}$) is eliminated reducing the number of elements in the sum of variables on Constraint (3.9) and one non-overlap constraint (3.8) is eliminated. The elimination of these redundant constraints keeps the model's optimality.

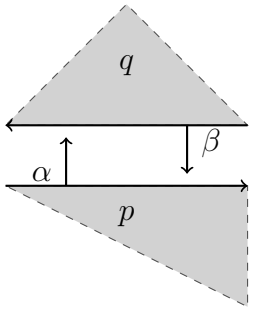


Figure 3.5: Parts p and q of different pieces that have lines with the same orientation and different directions.

Collinear lines of different parts of a piece can be represented by the same line in the model, since it leads to the same solution space. Consider a piece i that is composed by three parts, as illustrated in Figure 3.6. As the lines $\alpha \in K_{ip_1}$ and $\beta \in K_{ip_3}$ are collinear, the variable β can be removed from K_{ip_3} set and α can be included in the set. These modifications change constraints (3.8) and (3.9) eliminating a variable from the model for each pair of collinear lines. Note that line γ needs to be maintained.

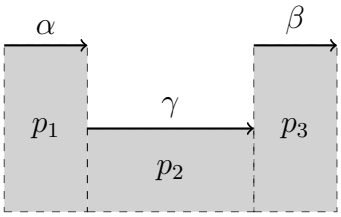


Figure 3.6: Piece i decomposed in three parts. An example of collinear lines is given by the lines α and β from parts p_1 and p_3.

Reducing the number of variables and constraints, the model size and complexity

are also reduced. These reductions lead the solution method to perform a faster search over the solution space.

Symmetry breaking

In the proposed formulation items of the same type are considered as different items, implying that a huge number of symmetric solutions are computed. This can be avoided by imposing that $x_i \leq x_j$ for all $i < j \in N$ if pieces i and j are of the same type. These constraints ensure that the pieces will respect a precedence order and significantly reduce the symmetry of the solution space.

Big-M estimation

The value of the big-M in constraint (3.8) must be estimated. If $v_{ij}^{pqk} = 1$, the value of M_{ij}^{pqk} is not important since it will be multiplied by zero. However, if $v_{ij}^{pqk} = 0$, then the following equation holds:

$$M_{ij}^{pqk} \geq C_{ij}^{pqk} + (a_x^k - b_x^k)(y_i - y_j) + (a_y^k - b_y^k)(x_i - x_j).$$

The value of M_{ij}^{pqk} depends on the placement positions of pieces i and j, the variables x_i, x_j, y_i and y_j. To eliminate the variables from the equation we take the board wigth W and an upper bound for the board length $\overline{L}$ as upper bounds for $(y_i - y_j)$ and $(x_i - x_j)$, respectively. The coefficients $(a_x^k - b_x^k)$ and $(a_y^k - b_y^k)$ are also replaced by their absolute value. By making these substitutions we obtain the following equation for big-M that guarantees that M_{ij}^{pqk} is large enough to make constraint (3.8) valid.

$$M_{ij}^{pqk} \geq C_{ij}^{pqk} + |a_x^k - b_x^k|H + |a_y^k - b_y^k|\overline{L}.$$

The upper bound on the board length ($\overline{L}$) is defined as the sum of the lengths of all the pieces. This value is clearly too high compared to the best solution, leading to doubts about the importance of the estimation of the big-M term. However, two facts must be taken into account. The first is that with the estimation of the big-M, it is possible to apply the model to instances of any size, avoiding numerical errors. The second is that as we had verified in our experiments, a tight M in the formulation generally makes it hard to the solver to find good quality solutions at the beginning of the search. A further discussion about upper bounds is provided in Section 3.2.4.

Not using advanced geometric structures in *DTM* also bring some drawbacks. The number of variables and constraints in the model is directly correlated with the number

of vertices of each pair of convex polygons that represented the piece that compose the different pieces that must be placed. This conclusion led us to use advanced structures, as the nofit polygon, as described in the next section.

3.2.2 NoFit Polygon Covering Model - *NFP−CM*

The *DTM* is relatively easy to built and uses only basic geometric informations. However, for the cases where several pieces are decomposed in several convex parts, the model may become difficult to solve because of the number of constraints and variables.

In this section, a model based on the nofit polygon is proposed. This approach has at most the same number of constraints and variables as the *DTM*. Some valid inequalities proposed for the *DTM* can also be used in this model. Moreover, a new set of valid inequalities can be introduced in the model, pruning symmetric solutions of the search space and leading therefore to a faster search.

Basic model

The main difference between the *NFP* Covering Model and the *DTM* is in how the non-overlapping constraints are tackled.

Consider the nofit polygon NFP_{ij} generated from pieces i and j as defined in Section 3.1.3.

$$NFP_{ij} = \bigcup_{p=1,...,Q_{ij}} NFP_{ij}^{p},$$

where NFP_{ij}^{p} is the part p of NFP_{ij} and Q_{ij} is the number of parts of NFP_{ij}. To ensure that the pieces do not overlap, the reference point of piece j must be outside NFP_{ij}^{p}, for all $p = 1, ..., Q_{ij}$.

Considering that K_{ij}^{p} is the set of directed edges of NFP_{ij}^{p}, these constraints are ensured by imposing, for each part of NFP_{ij}, that the reference point of piece j is at the right side of exactly one edge in K_{ij}^{p}. In order to build these constraints, the D-function (equation (3.1)) is used. Consider $a_{ij}^{p} = (a_{ij,x}^{p}, a_{ij,y}^{p})$ and $b_{ij}^{p} = (b_{ij,x}^{p}, b_{ij,y}^{p})$ as two consecutive vertices of the NFP_{ij}^{p}. Consider also x_i and y_i the variables which represent the point where piece i is placed on the board.

$$\begin{aligned} (a_{ij,x}^{p} - b_{ij,x}^{p})(a_{ij,y}^{p} - g_{ij,y}) - (a_{ij,y}^{p} - b_{ij,y}^{p})(a_{ij,x}^{p} - g_{ij,x}) &\leq 0 \Rightarrow \\ \overline{C}_{ij}^{pk} - (a_{ij,x}^{p} - b_{ij,x}^{p})g_{ij,y} + (a_{ij,y}^{p} - b_{ij,y}^{p})g_{ij,x} &\leq 0, \end{aligned} \tag{3.12}$$

where $g_{ij,x}$ and $g_{ij,y}$ are respectively the horizontal and vertical distance between the reference points of pieces i and j, i.e., $g_{ij,x} = x_i - x_j$ and $g_{ij,y} = y_i - y_j$, and

$$\overline{C}_{ij}^{pk} = (a_{ij,x}^p - b_{ij,x}^p)a_{ij,y}^p - (a_{ij,y}^p - b_{ij,y}^p)a_{ij,x}^p.$$

Inequality (3.12) ensures that the reference point of piece j is on the right side or over the line defined by the vertices a_{ij}^p and b_{ij}^p of the NFP_{ij}^p. Since only one line must be activated to avoid the overlap between pieces, the following constraints are imposed.

$$\overline{C}_{ij}^{pk} - (a_{ij,x}^p - b_{ij,x}^p)(y_j - y_i) + (a_{ij,y}^p - b_{ij,y}^p)(x_j - x_i) \leq (1 - v_{ij}^{pk})M_{ij}^{pk},$$
$$i = 1, ..., N-1,\ j = i+1, ..., N,\ p \in Q_{ij},\ k \in K_{ij}^p,$$

$$\sum_{k \in K_{ij}^p} v_{ij}^{pk} = 1,\ i = 1, ..., N-1,\ j = i+1, ..., N,\ p \in Q_{ij},$$

where the variable v_{ij}^{pk} is 1 if the reference point of piece j is on the right side or over the line k of NFP_{ij}^p and 0 otherwise. The same discussion presented in Section 3.2.1 can be applied to the estimation of the M_{ij}^{pk} term. These constraints are sufficient to ensure that the pieces do not overlap.

To ensure that the pieces are entirely inside the board, constraints (3.6) and (3.7) of the *DTM* are imposed in this model as constraints (3.14) and (3.15).

The complete *NFP* Covering Model formulation is presented in (3.13)-(3.19).

$$\min\ L \tag{3.13}$$

$$\text{s.t. } l_i^{left} \leq x_i \leq L - l_i^{right}, \qquad i = 1, ..., N, \tag{3.14}$$

$$w_i^{top} \leq y_i \leq W - w_i^{bottom}, \qquad i = 1, ..., N, \tag{3.15}$$

$$\begin{aligned} &\overline{C}_{pk}^{ij} - (a_{ij,x}^p - b_{ij,x}^p)(y_i - y_j) + \\ &(a_{ij,y}^p - b_{ij,y}^p)(x_i - x_j) \leq (1 - v_{ij}^{pk}) M_{ij}^{pk}, \qquad i = 1, ..., N-1, \\ &\qquad\qquad j = i+1, ..., N, \\ &\qquad\qquad p \in Q_{ij},\ k \in K_{ij}^p, \end{aligned} \tag{3.16}$$

$$\begin{aligned} &\sum_{k \in K_{ij}^p} v_{ij}^{pk} = 1, \qquad i = 1, ..., N-1, \\ &\qquad\qquad j = i+1, ..., N, \\ &\qquad\qquad p \in Q_{ij}, \end{aligned} \tag{3.17}$$

$$(x_i, y_i) \in \mathcal{R}^2, \qquad i = 1, ..., N, \tag{3.18}$$

$$\begin{aligned} &v_{ij}^{pk} \in \{0, 1\}, \qquad i, j = 1, ..., N, \\ &\qquad\qquad k \in K_{ij}^p,\ p \in Q_{ij}. \end{aligned} \tag{3.19}$$

It is clear that the symmetry breaking constraints presented in Section 3.2.1 are valid for this case, since they are related only with the feasible placement positions of the pieces reference points. Some other important inequalities are presented in the next section.

Valid inequalities and variable reductions

As in the *DTM* model, some valid inequalities and variable eliminations can be performed in order to reduce the model size.

Recalling that there is a binary variable assigned to each edge of each convex component p of each nofit polygon NFP_{ij}, the first variable reduction comes from assigning the same binary variable to two collinear edges belonging to components p and q of the same NFP_{ij}. The situation is similar to the one presented in Figure 3.6, considering now the nofit polygon parts instead of parts of the pieces.

Some valid inequalities are also proposed in order to reduce the search space. These inequalities are driven by the geometric characteristics of the different *NFP* parts. Consider two parts, p and q, of a given *NFP* as shown in Figure 3.7. Consider also a line k defined by two consecutive vertices of polygon p. Similarly, consider the line e defined by two consecutive vertices of polygon q. If these lines are parallel or if, for all feasible placements of polygons p and q on the board, these lines intersect outside the

board, then valid inequalities of three different types can be derived based on them.

The first set of valid inequalities comes from the observation that there are cases in which if a constraint, corresponding to the support line of a given edge, is active, then a constraint corresponding to the supporting line of an edge of another part of the *NFP* must also be active, otherwise the problem would be unfeasible. We call to the later a slave line and this happens whenever the domain of this second line constraint covers all the domain of the first line constraint. Figure 3.7b illustrates a case where the constraint corresponding to line β is activated (slaved) whenever the constraint corresponding to line associated to θ is active. Therefore, the constraint $v_{ij}^{p\beta} \geq v_{ij}^{q\theta}$ can be included in the model.

Two lines can be classified as covering lines (Figure 3.7c) if when merging the regions defined by the corresponding constraints the entire board is covered. This means that for any feasible solution at least one of these constraints must be active. Therefore, in the case depicted in Figure 3.7c the constraint $v_{ij}^{p\beta} + v_{ij}^{q\gamma} \geq 1$ must be added.

The last set of valid inequalities refers to disjoint lines (Figure 3.7d) and models the cases when two variables can not be active at the same time. Specifically, if two lines of different *NFP* parts define a disjoint region, i.e. if the regions do not intersect, then their corresponding constraints can not be activated at the same time (see α and θ in Figure 3.7d). In this case, the constraint $v_{ij}^{p\alpha} + v_{ij}^{q\theta} \leq 1$ must be added to the model.

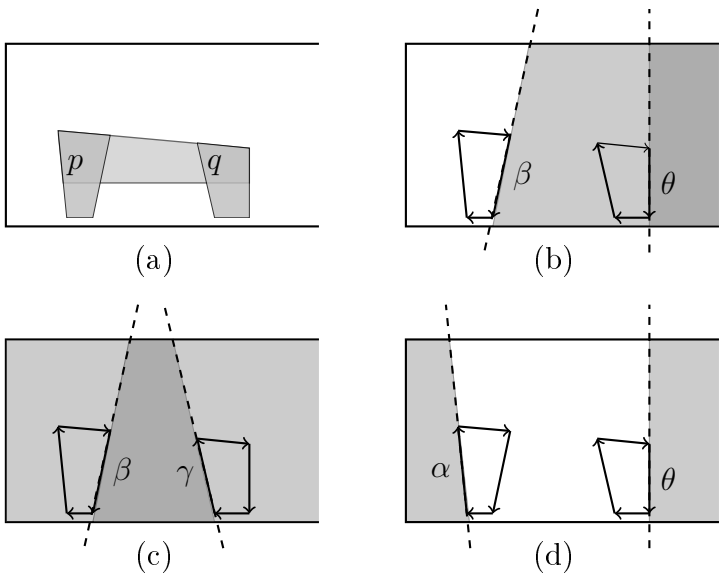


Figure 3.7: Illustrating the cuts over the solution space. (a) presents the board, the nofit polygon and its convex parts. Slave lines, covering lines and disjoint lines are presented in (b), (c) and (d), respectively.

The procedure to find the valid inequalities is presented in Algorithm 1.

Algorithm 1: VALID INEQUALITIES

Input: K_{ij}^p, K_{ij}^q.
Output: C set of cuts.

$C \leftarrow \emptyset$;
for $k \in K_{ij}^p$ *and* $e \in K_{ij}^q$ **do**
  if *(k and e do not intersect inside the board)* **then**
    Let a be a vertex of edge k, and b be a vertex of edge e;
    if *(b is on the left of k) and (a is on the right of e)* **then**
      Add cut ($v_{ij}^{qe} \geq v_{ij}^{pk}$) to set C; \\$e \equiv \beta$ and $k \equiv \theta$ in Figure 3.7(b)
    end
    if *(b is on the left of k) and (a is on the left of e)* **then**
      Add cut ($v_{ij}^{qe} + v_{ij}^{pk} \geq 1$) to set C; \\$e \equiv \beta$ and $k \equiv \gamma$ in Figure 3.7(c)
    end
    if *(b is on the right of k) and (a is on the right of e)* **then**
      Add cut ($v_{ij}^{qe} + v_{ij}^{pk} \leq 1$) to set C; \\$e \equiv \alpha$ and $k \equiv \theta$ in Figure 3.7(d)
    end
  end
end
return C;

3.2.3 Incorporating piece rotations

As previously stated, these models suppose that piece's orientation is fixed, i.e. no rotations are allowed. However, in many real-world applications pieces may have multiple orientations. In some applications any orientation is feasible, but considering this continuous rotation in the model would make it non-linear and out of the scope of our current research. In other applications a discrete and pre-defined set of orientations is possible for each piece. To consider this case each piece will be replicated as many times as the number of admissible orientations. Each one of the replicas will be treated in the models as a different piece, and therefore from now on i will stand for a piece in a given orientation. Variable $\delta_i \in \{0, 1\}$ will stand for the placement or not of piece i on the layout. Let also consider the existence of S sets Γ_s, each one of them containing indices of pieces that are mutually exclusive, as it happens when they represent different orientations of the same initial piece (Figure 3.8).

To impose that only one piece of each set Γ_s is placed on the layout constraints (3.20) have to be added to the models.

$$\sum_{i \in \Gamma_s} \delta_i = 1, \quad s = 1, \ldots, S \tag{3.20}$$

Additionally, the non-overlap constraints between two pieces in a particular orientation will only be activated if those orientations are the ones chosen by the model for those pieces. To achieve this, constraints (3.9) and (3.17) are replaced by constraints

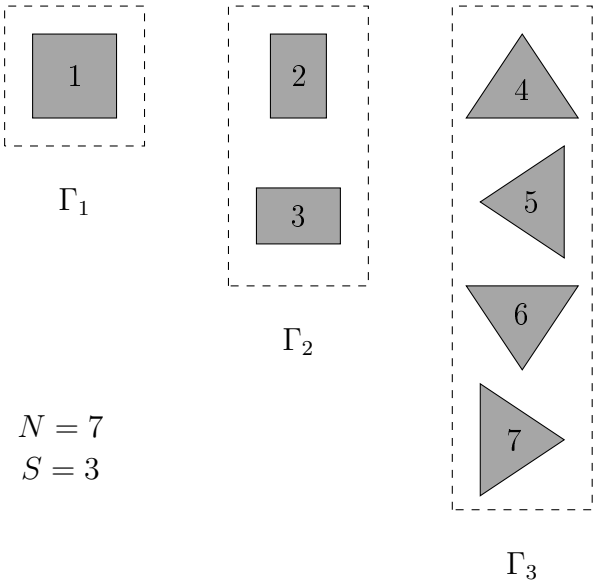


Figure 3.8: Example with $S = 3$ initial pieces which, after having rotations applied, originate a total of $N = 7$ pieces, together with the respective Γ_1, Γ_2 and Γ_3 sets definition.

(3.21)-(3.23) and (3.24)-(3.26), respectively.

$$\sum_{k \in K_{ip}} v_{ij}^{pqk} + \sum_{k \in K_{jq}} v_{ji}^{qpk} \geq \delta_i + \delta_j - 1, \quad 1 \leq i < j \leq N, \quad p \in P_i,\ q \in P_j, \tag{3.21}$$

$$\sum_{k \in K_{ip}} v_{ij}^{pqk} + \sum_{k \in K_{jq}} v_{ji}^{qpk} \leq \delta_i, \quad 1 \leq i < j \leq N, \quad p \in P_i,\ q \in P_j, \tag{3.22}$$

$$\sum_{k \in K_{ip}} v_{ij}^{pqk} + \sum_{k \in K_{jq}} v_{ji}^{qpk} \leq \delta_j, \quad 1 \leq i < j \leq N, \quad p \in P_i,\ q \in P_j. \tag{3.23}$$

$$\sum_{k \in K_{ij}^p} v_{ij}^{pk} \geq \delta_i + \delta_j - 1, \quad i = 1, ..., N-1, \quad j = i+1, ..., N, \quad p \in Q_{ij}, \tag{3.24}$$

$$\sum_{k \in K_{ij}^p} v_{ij}^{pk} \leq \delta_i, \quad i = 1, ..., N-1, \quad j = i+1, ..., N, \quad p \in Q_{ij}, \tag{3.25}$$

$$\sum_{k \in K_{ij}^p} v_{ij}^{pk} \leq \delta_j, \quad i = 1, ..., N-1, \quad j = i+1, ..., N, \quad p \in Q_{ij}. \tag{3.26}$$

Note that the Constraints (3.22) and (3.23) (or (3.25) and (3.26)) are not necessary to guarantee the feasibility of the solution, but they eliminate symmetries of the model, improving its performance. This modelling strategy does not only handle piece rotations but can also be used to address irregular packing problems in which the board is fully limited (both the length and the width are given). According to the typology of

Wäscher et al. (2007) it is the case of the irregular Knapsack Problem and the irregular Placement Problem. In these cases constraints (3.20) are satisfied as equal or less than one instead of equal to one.

3.2.4 Bounds

Defining better bounds for the problem may help the task of proving optimality. This section describes how the lower and upper bounds used in the models were calculated.

Lower bounds are defined with simple computations using the pieces and the board width. The length of a solution will always be greater then or equal to the length of the longest piece, and then this is assumed as a lower bound for the problem. Another lower bound that can be used is the length that would be used if the optimal solution had no waste of material to perform the cut. This value is obtained by dividing the total area of the pieces by the width of the board and is a simple generalization of the lower bound of Martello et al. (2003) for the two-dimensional rectangular strip packing problem. The maximum of these two lower bounds is used in the models aiming to give more information for the solution method and then leading to a faster convergence.

Unlike the lower bounds, estimating upper bounds as tight as possible does not lead to a faster convergence of the solution method. This happens because tighter upper bounds lead the branch-and-cut method to spend more time finding feasible solutions. Despite this, it is not useless to estimate upper bounds, since good upper bounds make the process of defining the variable domains and the big-M estimation automated for any instance dimensionality (see Section 3.2.1). Moreover, for some instance sizes, huge upper bounds for the board length could lead to numerical instability. In our experiments, a big-M large enough to solve all instances, including a large instance as *albano*, produces numerical instability to solve small instances as *three*, *fu5*, *shapes_4* For this reason, the used upper bound for the length, $\overline{L}$, is instance dependent, it is the sum of all pieces' length, i.e. $\overline{L} = \sum_{i=1,\dots,N}(l_i^{left} + l_i^{right})$. This upper bound on the length avoided the numerical instability for all the instances used in our computational experiments.

3.3 Computational results

Computational experiments were run to evaluate the performance of both *DTM* and *NFP−CM*. Variations of *NFP−CM* were tested in order to show the effect of using the proposed valid cuts. We compared the results of our approach with the results presented in Alvarez-Valdes et al. (2013). The authors presented results for the Fischetti and Luzzi (2009) model ($HS1$) and a variation of it with lifted bounds ($HS2$). They also presented the results of an extension of Gomes and Oliveira (2006) compaction model

(GO). In their article, the authors concluded that the $HS2$ model outperforms the $HS1$ and GO model and then only the results of $HS2$ will be compared with our best model. In addition, we present the results of our method for larger instances from the literature and new instances based on real world applications.

Three sets of instances were used in our experiments. The first set is the same used in Alvarez-Valdes et al. (2013) and was used to compare the performance of the proposed models and to compare the results of our best model against the $HS2$ model presented in Alvarez-Valdes et al. (2013). Classical instances of strip packing problems compose the second instance set. This instance set shows the effectiveness of the models for larger instances. Finally, the last instance set is composed by new instances based on a real world application, where small pieces can be positioned inside holes of larger pieces.

To build the proposed models, given the width of the board and the vertices of the pieces, we have a pre-processing phase. In this phase, the pieces were divided in the minimum number of convex parts by the Greene's partitioning algorithm proposed by Greene (1983) and implemented in CGAL. In order to build the nofit polygons, the ordering edges algorithm presented in Cuninghame-Green (1989) was used. Unlike other approaches, which also decompose non-convex pieces into convex components to generate the respective (convex) nofit polygons, these convex parts are not merged in a single nofit polygon but directly used in the model.

The computational experiments were performed on a Intel Core i7-2600 with 16 GB of memory using Ubuntu 12.04 operating system. To solve the proposed models we use the CPLEX 12.6 optimization tool with C/C++ programming language and default settings. All instances were run until optimality was proven or a time limit (denoted TL, defined as 3600 seconds) was reached. This computational environment and time limit are similar to the ones used in Alvarez-Valdes et al. (2013).

3.3.1 Evaluating the models performance

In this section the performances of the two proposed models, DTM and $NFP-CM$, are analysed and compared. For $NFP-CM$ we show the importance of the valid cuts by comparing the solutions obtained with the valid cuts ($NFP-CM$) and without them ($NFP-CMnc$). Only the set of instances present in Alvarez-Valdes et al. (2013) was used in this phase.

Table 3.1 shows the results obtained by DTM and $NFP–CM$ for this set of instances. In columns UB and LB we present the upper and lower bounds reported by CPLEX on termination. The solution gap, computed according to the formula $\frac{(UB-LB)}{UB}$ is given in column GAP. Column TIME gives the computational time, in seconds, reported by CPLEX.

Table 3.1: Computational results for the proposed models.

Instances	Pieces	*DTM*				*NFP−CMnc*				*NFP−CM*			
		UB	LB	GAP (%)	TIME	UB	LB	GAP (%)	TIME	UB	LB	GAP (%)	TIME
three	3	6.00	6.00	0.00	0.0	6.00	6.00	0.00	0.0	6.00	6.00	0.00	0.0
shapes_4	4	24.00	24.00	0.00	0.2	24.00	24.00	0.00	0.1	24.00	24.00	0.00	0.1
fu_5	5	17.89	17.89	0.00	0.1	17.89	17.89	0.00	0.1	17.89	17.89	0.00	0.0
glass1	5	45.00	45.00	0.00	0.1	45.00	45.00	0.00	0.1	45.00	45.00	0.00	0.2
fu_6	6	23.00	23.00	0.00	0.1	23.00	23.00	0.00	0.1	23.00	23.00	0.00	0.1
threep2	6	9.33	9.33	0.00	1.8	9.33	9.33	0.00	1.3	9.33	9.33	0.00	0.8
threep2w9	6	8.00	8.00	0.00	2.3	8.00	8.00	0.00	2.3	8.00	8.00	0.00	4.8
fu_7	7	24.00	24.00	0.00	0.1	24.00	24.00	0.00	0.1	24.00	24.00	0.00	0.1
glass2	7	45.00	45.00	0.00	6.6	45.00	45.00	0.00	5.7	45.00	45.00	0.00	64.3
fu_8	8	24.00	24.00	0.00	0.1	24.00	24.00	0.00	0.1	24.00	24.00	0.00	0.1
shapes_8	8	26.00	22.00	15.38	TL	26.00	26.00	0.00	1547.3	26.00	26.00	0.00	479.0
fu_9	9	25.00	25.00	0.00	7.4	25.00	25.00	0.00	5.3	25.00	25.00	0.00	5.9
threep3	9	13.53	11.20	17.24	TL	13.53	12.27	9.33	TL	13.53	11.33	16.26	TL
threep3w9	9	11.00	8.40	23.64	TL	11.00	9.00	18.18	TL	11.00	11.00	0.00	2144.5
glass3	9	100.00	100.00	0.00	2640.6	100.00	100.00	0.00	546.3	100.00	100.00	0.00	377.8
fu_10	10	28.69	28.68	0.03	575.6	28.68	28.68	0.00	379.2	28.68	28.68	0.00	278.6
dighe2	10	100.00	100.00	0.00	32.8	100.00	100.00	0.00	56.2	100.00	100.00	0.00	37.7
J1-10-20-0	10	18.00	18.00	0.00	55.5	18.00	18.00	0.00	10.4	18.00	18.00	0.00	7.7
J1-10-20-1	10	17.00	17.00	0.00	11.0	17.00	17.00	0.00	7.5	17.00	17.00	0.00	3.5
J1-10-20-2	10	20.00	20.00	0.00	22.3	20.00	20.00	0.00	7.6	20.00	20.00	0.00	5.2
J1-10-20-3	10	20.75	19.00	8.43	TL	20.75	20.00	3.61	TL	20.75	20.00	3.61	TL
J1-10-20-4	10	12.50	11.00	12.00	TL	12.50	12.50	0.00	290.4	12.50	12.50	0.00	149.6
J1-12-20-0	12	12.00	11.00	8.33	TL	12.00	12.00	0.00	92.8	12.00	12.00	0.00	39.9
J1-12-20-1	12	10.00	10.00	0.00	10.2	10.00	10.00	0.00	20.9	10.00	10.00	0.00	39.1
J1-12-20-2	12	12.00	12.00	0.00	125.9	12.00	12.00	0.00	80.6	12.00	12.00	0.00	50.6
J1-12-20-3	12	8.00	8.00	0.00	112.8	8.00	8.00	0.00	2120.9	8.00	8.00	0.00	160.4
J1-12-20-4	12	13.00	12.00	7.69	TL	13.00	12.00	7.69	TL	13.00	13.00	0.00	1618.7
fu	12	34.00	28.50	16.18	TL	33.14	28.50	14.00	TL	33.14	28.50	14.00	TL
J1-14-20-0	14	13.00	10.60	18.46	TL	12.00	12.00	0.00	2011.2	12.00	12.00	0.00	844.5
J1-14-20-1	14	12.00	9.85	17.92	TL	12.00	9.85	17.92	TL	12.00	9.92	17.33	TL
J1-14-20-2	14	14.00	12.00	14.29	TL	14.00	12.00	14.29	TL	14.00	12.00	14.29	TL
J1-14-20-3	14	11.00	8.60	21.82	TL	10.00	10.00	0.00	440.1	10.00	10.00	0.00	295.4
J1-14-20-4	14	15.00	11.90	20.67	TL	14.00	11.90	15.00	TL	14.00	11.90	15.00	TL
poly1a0	15	15.87	13.00	18.10	TL	16.38	13.00	20.61	TL	16.35	13.00	20.49	TL
dighe1	16	125.77	100.00	20.49	TL	116.45	100.00	14.13	TL	122.75	100.00	18.54	TL
#Optimal					20				25				27

Optimal: number of instances for which optimality was proven,
TL: time limit of 3600 sec.

To have a better overview of the models comparison, we build a performance profile graph (Dolan and Moré, 2002), using the computational time as a performance measure. In this graph, each model is represented by a curve. A point (x, y) in a curve of a model m means that the computational time model m took to solve $(100 \times y)\%$ of the instances is at most x times the computational time the fastest model took to solve them. To build the graphs in Figure 3.9 we used all the 35 instances, but, if a model could not find an optimal solution (i.e. gap greater than 0), we considered that the time it spend on solving the instance was "infinite". To build the graphs in Figure 3.10 we used only the 20 instances for which the models could compute the optimal solution. In Figure 3.10 and in Figure 3.9, the graph at the right is a zoom of the left portion of the graph at the left.

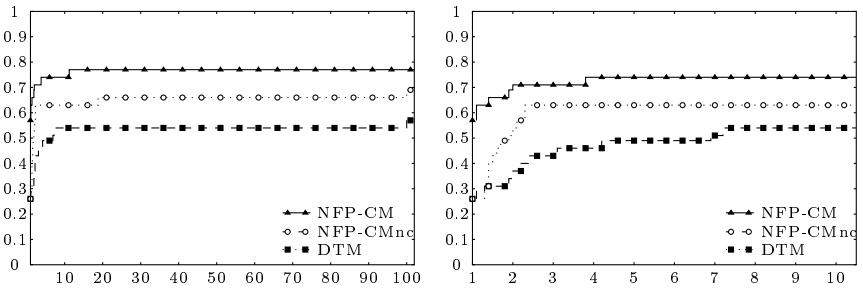


Figure 3.9: Performance profile using computational time as performance measure and considering all 35 instances. If a model cannot find an optimal solution, we consider it took "infinite" time. The graph at the right is a zoom of the left portion of the graph at the left.

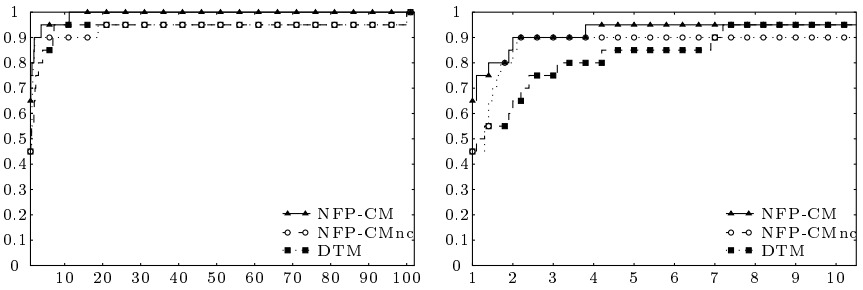


Figure 3.10: Performance profile using computational time as performance measure and considering only the 20 instances for which all models found the optimal solution. The graph at the right is a zoom of the left portion of the graph at the left.

In Table 3.1 and Figure 3.9, when we compare the computational time spent by the models to solve all 35 instances we can see that both *DTM* and *NFP−CMnc* are the fastest models for 26% of the instances, whereas *NFP–CM* is the fastest model for 57% of the instances. Besides that, *DTM* was able to solve to optimality 20 instances (57%), *NFP–CMnc* solved 25 instances (71%) and *NFP–CM* solved 27 instances (77%). When we compare the computational time spent to solve only the 20 instances for which all

models computed optimal solutions (see Figure 3.10), both *DTM* and *NFP−CMnc* are the fastest models for 45% of the instances, whereas *NFP−CM* is the fastest model for 65% of the instances.

In Table 3.1 we can also see that, with the exception of instances threep3, poly1a0 and dighe1, in all the other five instances for which *NFP–CM* could not find an optimal solution (J1-10-20-3, fu, J1-14-20-1, J1-14-20-2 and J1-14-20-4), it obtained either an equal or a smaller gap than the other two models.

Therefore, *NFP − CM* has a better performance than the other two models, at least for this set of instances. We can also see that the performance of *NFP − CMnc* is better than the performance of *DTM*. This behaviour was expected since in *DTM* only the geometric information about the pieces parts is used to build the model while the other proposed models use the information about the nofit polygon. Moreover, the performance of *NFP−CM* is better than the performance of *NFP−CMnc* because the latter does not have inequalities associating the parts of each nofit polygon used to build the model. As expected, the amount of information used to build the model is related with its performance. However, in situations where there are no geometric tools to generate the nofit polygon or the nofit polygon generator can not handle all the pieces characteristics, the *DTM* can be attractive to solve the problem. Furthermore, it is easier to include additional constraints in a model which depends on less information to be built, this fact also makes *DTM* attractive.

Since *NFP−CM* performed better, we will only use it on the remaining sections to make performance comparisons.

3.3.2 Comparing with the literature

To assess the performance of *NFP−CM* we will compare it with the $HS2$ model which is the best model presented in Alvarez-Valdes et al. (2013). To do this, we use only the instances reported in Alvarez-Valdes et al. (2013).

Table 3.2 shows the results obtained by *NFP−CM* and the results of $HS2$ reported in Alvarez-Valdes et al. (2013). In columns LB and GAP we have the lower bound and solution gap (given by $\frac{(UB-LB)}{UB}$) for each solution computed. Column TIME gives the computational time, in seconds, reported by CPLEX. It is important to notice that, although the computer environment used in Alvarez-Valdes et al. (2013) is similar to the one we used, they are not the same (the main difference being the CPLEX version - in Alvarez-Valdes et al. (2013) version 12.1 was used). Therefore, we must be cautious when comparing the models performance.

As can be seen in Table 3.2, the $HS2$ and *NFP−CM* approaches obtained almost the same number of optimal solutions within the allowed time limit (25 and 27, respectively). In general, for instances with up to nine pieces the results obtained with both

Table 3.2: Comparing the results of *NFP Covering Model* with the literature.

Instances	Pieces	HS2			*NFP−CM*		
		LB	GAP (%)	TIME	LB	GAP (%)	TIME
three	3	6.00	0	0.8	6.00	0	0.0
shapes_4	4	24.00	0	0.0	24.00	0	0.1
fu_5	5	17.89	0	0.1	17.89	0	0.0
glass1	5	45.00	0	0.1	45.00	0	0.2
fu_6	6	23.00	0	0.5	23.00	0	0.1
threep2	6	9.33	0	3.9	9.33	0	0.8
threep2w9	6	8.00	0	8.5	8.00	0	4.8
fu_7	7	24.00	0	1.0	24.00	0	0.1
glass2	7	45.00	0	2.8	45.00	0	64.3
fu_8	8	24.00	0	1.3	24.00	0	0.1
shapes_8	8	26.00	0	272.0	26.00	0	479.0
fu_9	9	25.00	0	70.0	25.00	0	5.9
threep3	9	13.53	0	3394.0	11.33	16	TL
threep3w9	9	10.00	9	TL	11.00	0	2144.5
glass3	9	100.00	0	324.0	100.00	0	377.8
fu_10	10	28.68	0	3064.0	28.68	0	278.6
dighe2	10	100.00	0	177.0	100.00	0	37.7
J1-10-20-0	10	18.00	0	45.0	18.00	0	7.7
J1-10-20-1	10	17.00	0	34.0	17.00	0	3.5
J1-10-20-2	10	20.00	0	304.0	20.00	0	5.2
J1-10-20-3	10	20.67	0	TL	20.00	4	TL
J1-10-20-4	10	12.50	0	628.0	12.50	0	149.6
J1-12-20-0	12	12.00	0	509.0	12.00	0	39.9
J1-12-20-1	12	10.00	0	2430.0	10.00	0	39.1
J1-12-20-2	12	12.00	0	2332.0	12.00	0	50.6
J1-12-20-3	12	8.00	0	214.0	8.00	0	160.4
J1-12-20-4	12	12.00	14	TL	13.00	0	1618.7
fu	12	24.00	29	TL	28.50	14	TL
J1-14-20-0	14	11.00	21	TL	12.00	0	844.5
J1-14-20-1	14	8.00	43	TL	9.92	17	TL
J1-14-20-2	14	10.00	36	TL	12.00	14	TL
J1-14-20-3	14	8.00	33	TL	10.00	0	295.4
J1-14-20-4	14	10.00	35	TL	11.90	15	TL
poly1a0	15	13.00	28	TL	13.00	20	TL
dighe1	16	71.00	54	TL	100.00	19	TL
#Optimal				24			27

Optimal: number of instances for which optimality was proven.

models are very similar, except for threep3 and threep3w9, instances where either $HS2$ or $NFP-CM$ were better. However, $NFP-CM$ is significantly faster than $HS2$ for instances with 10 or more pieces, even considering the difference between the CPLEX

version. According to IBM[2], CPLEX 12.6 (used to run $NFP-CM$) is on average 40% faster than CPLEX 12.1 (used to run $HS2$). For these instances, $HS2$ presented better results only for the instance J1-10-20-3. Furthermore, even for the instances in which the time limit was reached, the gaps were smaller for $NFP–CM$. The optimal solutions obtained by $NFP-CM$ that were not obtained by $HS2$ are displayed in Figure 3.11.

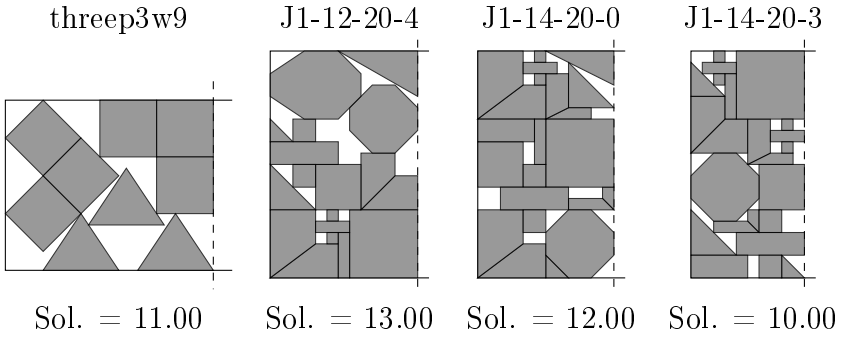


Figure 3.11: Optimal solutions proved by $NFP-CM$ and not proved for $HS2$ within the time limit.

3.3.3 Larger instances from literature

To assess the performance of $NFP-CM$ when solving larger instances, we first use 10 classical instances from the literature with 16 or more pieces. The results can be seen on Table 3.3.

From Tables 3.1 and 3.3 we can see that, given a time limit of 3600 seconds, $NFP-CM$ can find an optimal solution for most of the instances with up to 12 pieces (except instances threep3, J1-10-20-3 and fu). When the number of pieces if greater than 12, $NFP-CM$ can find a feasible solution for all instances with up to 30 pieces. For instances with more than 30 pieces, $NFP-CM$ cannot find a feasible solution. This indicates that, when the instance is small (at most 12 pieces), $NFP-CM$ has a good chance of solving it to optimality. On the other hand, if the instance is big (more than 30 pieces), it is probable $NFP-CM$ will not be able to even find a feasible solution.

3.3.4 New real world based instances

Since $NFP-CM$ (as well as $NFP-CMnc$ and DTM) can deal with pieces with holes without any special structure needed to built the model, we propose some instances based on a metal cutting layout from the industry. These instances are characterized by large pieces with holes where some small pieces from the instances can be placed.

[2]http://www-01.ibm.com/software/commerce/optimization/cplex-performance - accessed in December 12th, 2014.

Table 3.3: Computational results for instances with more than 16 pieces.

Instances	Pieces	*NFP−CM*			
		UB	LB	GAP (%)	TIME
blaze2	16	21.44	14.67	31.58	TL
mao	20	2180.97	1473.96	32.42	TL
albano	24	12876.60	8711.34	32.35	TL
marques	24	92.44	69.14	25.21	TL
jakobs1	25	12.00	9.73	18.96	TL
jakobs2	25	30.76	19.26	37.39	TL
blaze1	28	30.51	21.07	30.95	TL
dagli	30	71.04	50.60	28.77	TL
shapes0	48	x	39.68	x	TL
trousers	64	x	217.68	x	TL
Av. GAP				26.959	

x: no feasible solution found,
Av. GAP: average GAP.

The real pieces were approximated by polygonal forms as presented in Figure 3.12. The instances generated with these pieces were named *metal*.

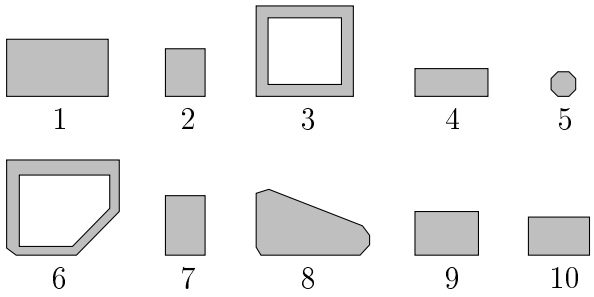


Figure 3.12: Pieces of the *metal* instances.

Four instance sets were designed using these pieces, called metal0, metal1, metal2 and metal3. The instance set metal1 is composed of five instances, identified as metal1$-i$, $i = 1, ..., 5$. The demand of pieces in metal1-1 was based on the real world instance and all the piece types presented in Figure 3.12 are in this instance. The demand of metal1$-i$, $i = 2, ..., 5$ is i times the demand of metal1-1.

The instances in sets metal2 and metal3 were derived from metal1 aiming to obtain layouts with holes where the pieces fit well (metal2) and where they do not fit well (metal3) in order to analyse these characteristics. Both sets metal2 and metal3 have four instances each and their names can be identified by the suffix $-i$, $i = 2, ..., 5$ on the set name. We obtain instance metal2-i taking pieces one to five from instance metal1-i, with the corresponding demand. The remaining pieces and corresponding demands give instance metal3-i. Sets metal2-1 and metal3-1 were omitted because

they had trivial solutions.

To try to identify the size of instances with pieces that have holes for which the model can find proven optimal solutions, instance set metal0 with a small number of pieces was proposed. Eleven instances compose metal0 set and can be identified by the suffix $-i$, $i = 3, ..., 13$ on the set name. The demands of the instances metal0-i, $i = 3, ..., 13$, were created using the i first pieces of Figure 3.12. When the ten different piece types are used, the pieces whose demand is greater than one on metal1-1 is increased by one. Clearly, for these instances the strip width is too high and then its size was reduced by 25%, 50% and 75% depending on the instance.

Table 3.4 shows the demand of the pieces for each instance on these sets.

Table 3.4: Pieces demand of metal instance set.

	Strip	Piece									
Instance	width	1	2	3	4	5	6	7	8	9	10
metal0-3	250	1	1	1	0	0	0	0	0	0	0
metal0-4	250	1	1	1	1	0	0	0	0	0	0
metal0-5	250	1	1	1	1	1	0	0	0	0	0
metal0-6	250	1	1	1	1	1	1	0	0	0	0
metal0-7	500	1	1	1	1	1	1	1	0	0	0
metal0-8	500	1	1	1	1	1	1	1	1	0	0
metal0-9	500	1	1	1	1	1	1	1	1	1	0
metal0-10	750	1	1	1	1	1	1	1	1	1	1
metal0-11	750	1	2	1	1	1	1	1	1	1	1
metal0-12	750	1	2	1	2	1	1	1	1	1	1
metal0-13	750	1	2	1	2	2	1	1	1	1	1
metal1-1	1000	1	2	1	2	3	1	1	1	1	1
metal1-2	1000	2	4	2	4	6	2	2	2	2	2
metal1-3	1000	3	6	3	6	9	3	3	3	3	3
metal1-4	1000	4	8	4	8	12	4	4	4	4	4
metal1-5	1000	5	10	5	10	15	5	5	5	5	5
metal2-2	1000	2	4	2	4	6	0	0	0	0	0
metal2-3	1000	3	6	3	6	9	0	0	0	0	0
metal2-4	1000	4	8	4	8	12	0	0	0	0	0
metal2-5	1000	5	10	5	10	15	0	0	0	0	0
metal3-2	1000	0	0	0	0	0	2	2	2	2	2
metal3-3	1000	0	0	0	0	0	3	3	3	3	3
metal3-4	1000	0	0	0	0	0	4	4	4	4	4
metal3-5	1000	0	0	0	0	0	5	5	5	5	5

The pieces' shapes are based on a metal cutting layout from a real world problem. The piece vertices are presented in Appendix A.

Table 3.5 shows the results obtained by $NFP-CM$ for the *metal* sets of instances. Figure 3.13 shows the pictures of the optimal solutions obtained by $NFP-CM$.

Table 3.5: Computational results for instances derived from the metal layout.

Instances	Pieces	*NFP−CM*			
		UB	LB	GAP (%)	TIME
metal0-3	3	501.00	501.00	0.00	0.0
metal0-4	4	501.00	501.00	0.00	0.0
metal0-5	5	501.00	501.00	0.00	0.0
metal0-6	6	785.00	785.00	0.00	0.2
metal0-7	7	501.00	501.00	0.00	3.4
metal0-8	8	529.00	529.00	0.00	9.9
metal0-9	9	529.00	529.00	0.00	11.2
metal0-10	10	356.00	356.00	0.00	3570.9
metal3-2	10	291.13	286.00	1.76	TL
metal0-11	11	364.32	345.00	5.30	TL
metal0-12	12	399.00	295.58	25.92	TL
metal0-13	13	490.00	332.98	32.05	TL
metal1-1	14	300.00	286.00	4.67	TL
metal3-3	15	518.12	321.05	38.04	TL
metal2-2	18	300.00	256.00	14.67	TL
metal3-4	20	668.00	428.06	35.92	TL
metal3-5	25	770.00	535.08	30.51	TL
metal2-3	27	490.00	363.61	25.79	TL
metal1-2	28	644.00	456.44	29.12	TL
metal2-4	36	601.00	484.81	19.33	TL
metal1-3	42	1154.00	684.65	40.67	TL
metal2-5	45	774.00	606.01	21.70	TL
metal1-4	56	x	912.87	x	TL
metal1-5	70	x	1141.09	x	TL

x: no feasible solution found.

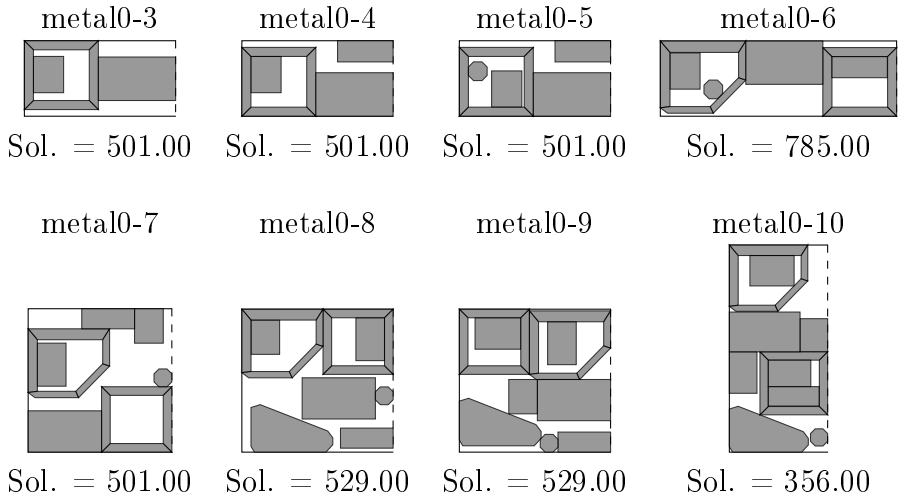


Figure 3.13: Optimal solutions of metal0 instances.

As we can see in Table 3.5, $NFP-CM$ was able to solve to optimality all *metal* instances with up to 10 pieces. For instances with a number of pieces between 10 and 45, $NFP-CM$ was always able to find a feasible solution. For instances with more than 45 pieces, it could not find a feasible solution. This is consistent with the results reported in Sections 3.3.1 and 3.3.3.

3.3.5 Instances with rotations

When considering piece rotations it is not possible to make comparisons with previously published exact methods because none of them allows piece rotations, i.e. that pieces may be placed with an orientation among a set of given feasible orientations. However, in order to provide grounds for future research in the field, experiments were run with the models "*NFP–CM* with rotations" and "DTM with rotations" using a set of smaller instances allowing two different orientations for each piece: the original (rotation of 0 degrees) and the orientation corresponding to a rotation of 180 degrees. It should be noticed that allowing piece rotations significantly increases the number of variables and the size of the model, and therefore only small instances can be tackled.

As we can see in Table 3.6, as expected, the solutions are always equal or better than the ones when no rotations are allowed, but the time needed to prove solution optimality increases quickly and quite soon it is not possible to prove it.

Table 3.6: Computational results with rotations of 0 and 180 degrees.

	$NFP-CM$ with rotations				DTM with rotations			
Instance	UB	LB	GAP (%)	TIME	UB	LB	GAP (%)	TIME
three	6,00	6,00	0,00	0,04	6,00	6,00	0,00	0,02
threep2	9,22	9,22	0,00	2,77	9,22	9,22	0,00	2,55
threep2w9	7,50	7,50	0,00	6,00	7,50	7,50	0,00	8,41
threep3	14,00	11,45	18,18	TL	13,22	11,00	16,81	TL
threep3w9	10,00	8,50	15,00	TL	10,00	8,00	20,00	TL
fu5	17,89	17,89	0,00	0,07	17,89	17,89	0,00	0,08
fu6	23,00	23,00	0,00	0,10	23,00	23,00	0,00	0,13
fu7	24,00	24,00	0,00	0,16	24,00	24,00	0,00	0,25
fu8	24,00	24,00	0,00	0,20	24,00	24,00	0,00	0,19
fu9	24,71	24,71	0,00	135,71	24,71	24,71	0,00	101,20
fu10	28,00	28,00	0,00	1052,27	28,00	28,00	0,00	698,84
fu	32,70	28,50	12,86	TL	32,18	24,00	25,41	TL

3.4 Final remarks

This chapter presented two new mathematical programming models for the two-dimensional irregular packing problem. These models aim to overcome the limitations of previous models in what concerns the geometry of the pieces they are able to deal with. The first model (*DTM*) states the piece non-overlapping constraints using direct trigonometry, while the second model ($NFP-CM$) firstly decomposes the pieces into convex parts and then states the non-overlapping constraints using the nofit polygons between these convex parts, as a covering of the actual non-convex nofit polygons. For the $NFP-CM$ valid inequalities were developed.

The computational experiments were divided in three phases: first the new models were tested over a set of 35 commonly used benchmark instances; secondly the models were compared against the best model known in the literature, *HS2* by Alvarez-Valdes et al. (2013), over the same classical set of instances; and finally the robustness of the new models was proven by running experiments over a set of new real-world based set of instances, incorporating geometric characteristics not dealt yet by previous models. From the first phase of experiments resulted the supremacy of the model $NFP-CM$ that, within the time limit of one hour, was able to solve until optimality 77% of the instances and, for the instances in which optimal solutions were achieved by more than one model or variant, it was the fastest model for 57% of the instances. For the 8 instances in which optimality was not proven, *NFP–CM* was able to generate a feasible solution and, in 5 of those instances, with a gap smaller than the other models. In the second phase, when comparing the new *NFP–CM* model against *HS2*, *NFP–CM* solves more problems until optimality (27 against 25), clearly faster for instances with 10 or more pieces (around 13 times faster, already taking into account the different versions of CPLEX used) and, when optimality is not guaranteed, with smaller gaps. In the third phase of experiments it was possible to successfully solve new instances where the pieces have holes (the only geometric characteristic not addressed in the previous experiments), solving until optimality instances until 10 pieces and generating feasible solutions for instances with up to 45 pieces. The gaps are high, but it is well-known that irregular strip packing problems have very poor lower bounds.

The new models presented can effectively address irregular strip packing problems with any kind of geometry that can be described by a polygon, without any approximation or simplification, being therefore robust in what concerns piece geometry. Equally important is the fact that they resort to much simpler geometric tools than previous models, easier to implement and without numerical stability problems. The downside of mathematical programming model based approaches is still the size of the instances that are possible to solve until optimality and, given the poor quality of the available

lower bounds, feasible solutions are not better than feasible solutions generated by sophisticated (meta)heuristics in the same amount of time. Lower bounds for irregular strip packing problems is clearly a very difficult but relevant topic of future research.

In many real-world applications pieces may have a pre-defined set of orientations. This possibility was considered as an extension of the proposed models and required the replication of each piece as many times as the number of admissible orientations and an additional binary variable for each of these replicas. The size of the instances that were possible to solve when considering the possibility was fairly small and efficiently tackling rotations in exact approaches is a challenge for future research.

Chapter 4

A dots data structure to handle the geometry of nesting problems[1]

The core constraints for the irregular cutting and packing problems arise from the geometry of the board and the pieces. As presented in Chapter 2, in the literature, the geometry is handled by different strategies. Trying to merge the simplicity of the raster points representation with the accuracy of the nofit polygon, some authors used a finite set of dots to place the pieces on the board while the overlap analysis among pieces is performed using the nofit polygon. Although it is a promising approach, there is no data structure in the literature to represent this type of geometry. In this chapter, we propose a new data structure that caries the information of the feasible placement positions for the pieces on the board and detects the overlap among pieces. This structure, simplifies the development of models and methods that use discrete placement positions for the pieces.

This chapter is strongly based on the paper "An innovative data structure to handle the geometry of nesting problems" and is organized as follows. The next section presents the problem definition. Section 4.2 details the dot data structure. A reformulation of the dotted-board model using the proposed data structure is presented in Section 4.3. Section 4.4 depicts examples of mesh generation rules. The complexity of the data structure is presented in Section 4.5. In Section 4.6, the influence of the mesh on the solution quality is evaluated for the dotted-board model. Some final conclusions are presented in Section 4.7.

[1]The text of this chapter is strongly based on the paper "An innovative data structure to handle the geometry of nesting problems" which is under review.

4.1 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 (W) 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.

In this chapter, each piece t at rotation r is represented by a set of dots ordered on the clockwise direction and to ensure that this piece is entirely inside the board, the innerfit polygon (IFP_{tr}) is used (Figure 2.3). To evaluates the overlap among pieces, the nofit polygon of each piece of type t at rotation r and piece of type u at rotation s ($NFP_{tr,us}$) is used (see Figure 2.5). Further details about the pieces representation and geometric structures can be found in Section 2.1.

4.2 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 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.

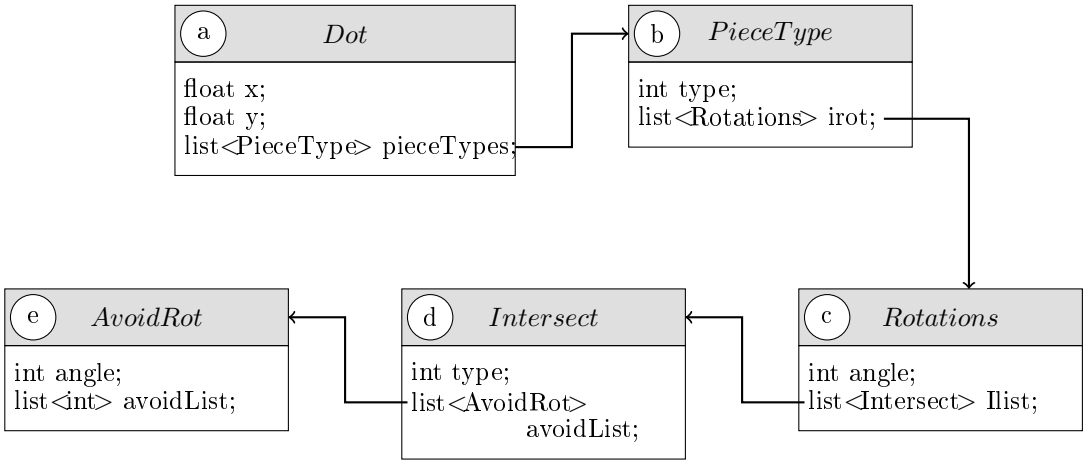


Figure 4.1: Data types used in the data structure.

Any dot (Figure 4.1a) 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 4.1b and 4.1c). The piece types at each dot of the structure are therefore always inside the board, i.e. the *IFP*s 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 4.1d). Each one of these elements points to the list of forbidden dots for piece type u at rotation s (Figure 4.1e). 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 4.2 shows an example of this structure.

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 4.2a, together with a list of admissible rotations at this dot (Figure 4.2b).

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

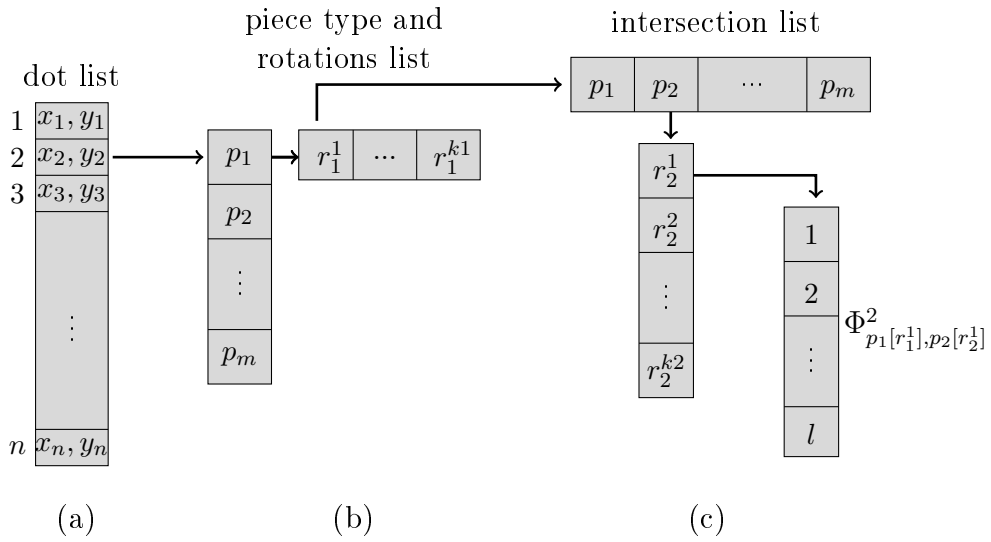


Figure 4.2: The dot data structure. (a) the dot list and (b) the piece list of dot 2 together with the admissible rotations for piece type 1. The intersection list presented in (c) contains $\Phi^2_{t_1^2[r_1^1],u_2^1[s_1^2]}$, the set of dots where a piece of type u_2^1 at rotation s_1^2 overlaps piece of type t_1^2 at rotation r_1^1, when the later is placed at dot 2.

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 4.2c). 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.

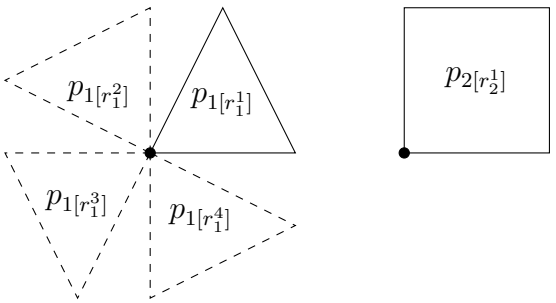


Figure 4.3: An application of the dot data structure – piece types and their admissible rotations.

By way of example, figures 4.3 – 4.5 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 4.3). Figure 4.4 shows an irregular board with a hole (shaded area). The dotted lines represent the nofit polygon between the

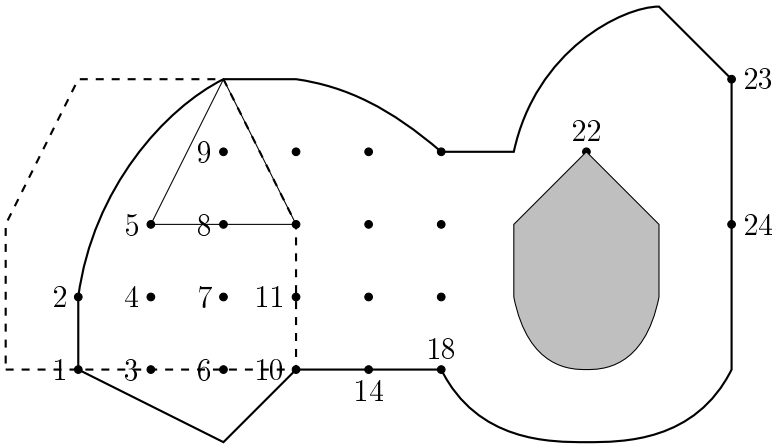


Figure 4.4: An application of the dot data structure – board and feasible dots.

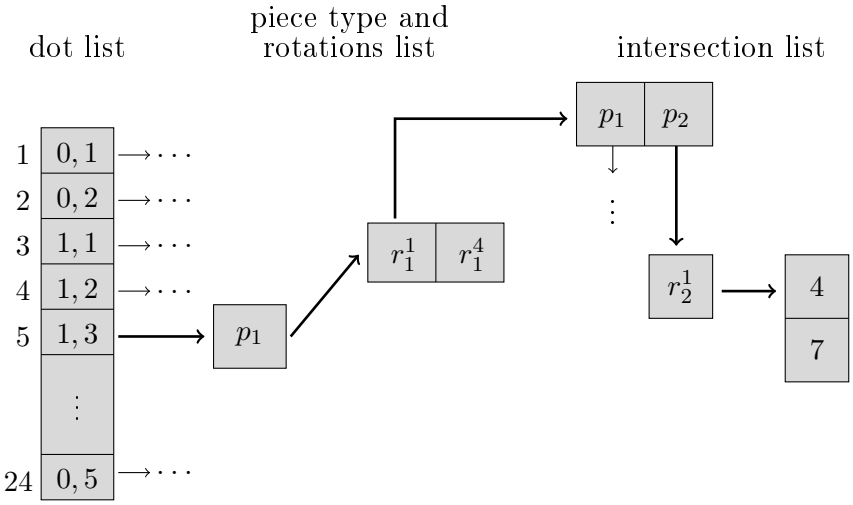


Figure 4.5: An application of the dot data structure – intersection list with a piece of type 2 when a piece of type 1 is placed at dot 5.

triangle, already placed on the board, and the square that we want to add to the same board. 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 inserted 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 $\Phi^5_{t_1^5[r_1^1],u_2^1[s_1^2]}$ ($\Phi^5_{triangle[0°],square[0°]}$) (Figure 4.5).

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.

4.3 The dotted-board model

In this section the dotted-board model proposed in Toledo et al. (2013) is adapted to the new dot data structure. As detailed in section 4.4, 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. (2013). However, since the dot data structure defined in Section 4.2 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_{tr}^{d} 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 (4.1) - (4.6).

$$\text{minimize} \quad L \tag{4.1}$$

$$\text{subject to:} \quad (d_x + l_{tr}^{right})\delta_{tr}^{d} \leq L \qquad \forall t \in T, r \in R_t, d \in D_{t[r]} \tag{4.2}$$

$$\delta_{tr}^{d} + \delta_{us}^{d'} \leq 1 \qquad \forall t \in T, u \in T, r \in R_t, s \in R_u, d \in D_{t[r]}, d' \in \Phi_{t[r],u[s]}^{d} \tag{4.3}$$

$$\sum_{r \in R_t} \sum_{d \in D_{t[r]}} \delta_{tr}^{d} = q_t \qquad \forall t \in T \tag{4.4}$$

$$\delta_{tr}^{d} \in \{0,1\} \qquad \forall t \in T, r \in R_t, d \in D_{t[r]} \tag{4.5}$$

$$L \in \mathbb{R}_{+} \tag{4.6}$$

The objective function (4.1) 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 2.3a. To ensure that auxiliary variable L is equal or greater than the used board length, constraint (4.2) must hold. Constraint (4.3) 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 $\Phi^d_{t[r],u[s]}$, *i.e.*, $d' \in \Phi^d_{t[r],u[s]}$. Constraint (4.4) ensures that, for each piece type t, the requested number of pieces (q_t) is placed. Constraints (4.5) and (4.6) define the domains of the variables.

The scope of the new model is broader than that of the model proposed in Toledo et al. (2013) for it allows for piece rotation. Moreover, due to the new dot data structure, the model is mesh-type independent.

4.4 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.

4.4.1 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 4.6(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 4.6(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 4.6(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 4.6(d)). In Figure 4.7 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.

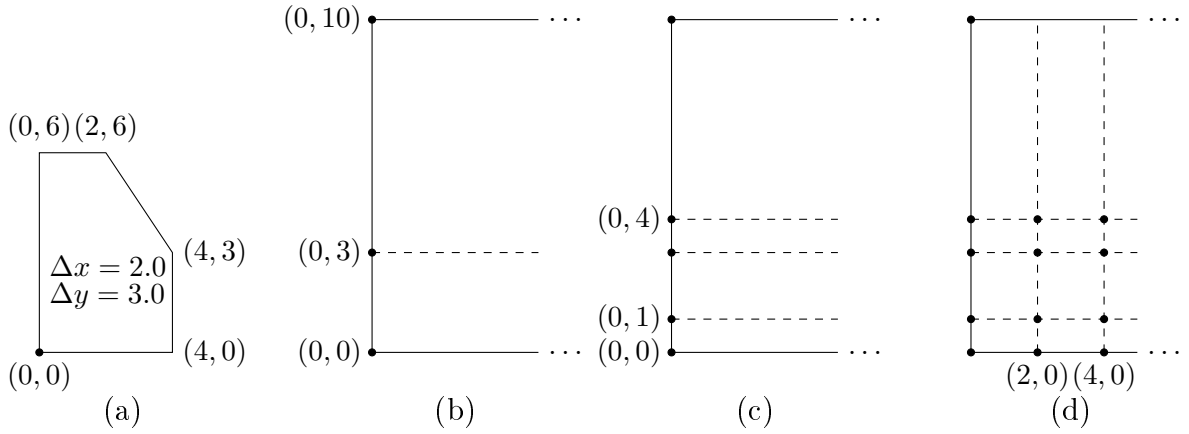


Figure 4.6: Example of a piece-based mesh for one piece type.

Algorithms 2 and 3 present the procedures to generate the dot list using the piece-based mesh. The procedure to build the intersection list is presented in Algorithm 4.

4.4.2 *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 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 4.8(a). Consider a piece of type t at rotation r (p_{tr}), placed at the lower left

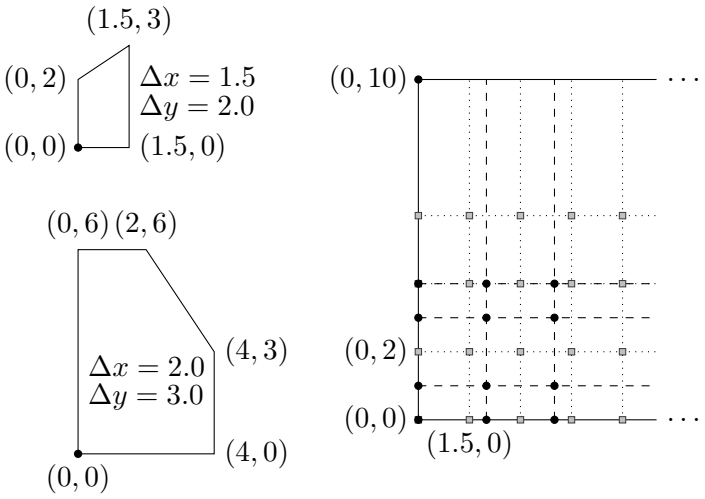


Figure 4.7: An example of a piece-based mesh for two piece types. The squares and circles represent feasible placement dots for the smaller and larger piece types, respectively.

Algorithm 2: Generation of a piece-based mesh

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

$D[\ ] \leftarrow \emptyset$;
for $t \in T$ **do**
  for $r \in R_t$ **do**
    calculate Δx_{tr} and Δy_{tr};
    $x = l_{tr}^{right}$;
    while $x \leq W - l_{tr}^{right}$ **do**
      $y = w_{tr}^{top}$;
      while $y \leq W - w_{tr}^{bottom}$ **do**
        $d = (x, y)$;
        $D[\ ] = \text{UPDATE\_DOT\_LIST}(D[\ ], d, t, r)$;
        $y = y + \Delta y_{tr}$;
      end
      $y = W - w_{tr}^{bottom}$;
      while $y \geq w_{tr}^{top}$ **do**
        $d = (x, y)$;
        $D[\ ] = \text{UPDATE\_DOT\_LIST}(D[\ ], d, t, r)$;
        $y = y - \Delta y_{tr}$;
      end
      $x = x + \Delta x_{tr}$;
    end
  end
end
return $D[\ ]$;

Algorithm 3: UPDATE_DOT_LIST.

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

```
if d ∉ D[ ] then
    D[ ] = D[ ] ⋃ d;
    D[d] ← ∅;
end
if t ∉ D[d] then
    D[d] = D[d] ⋃ t;
    D[d][t] ← ∅;
end
D[d][t] = D[d][t] ⋃ r;
D[d][t][r] ← ∅;
return D[ ];
```

Algorithm 4: INTERSECTION LIST GENERATION

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

```
Φ^d_{t[r],u[s]} ← ∅;
for d' ∈ D[ ] do
    for t, u ∈ T do
        for r ∈ R_t and s ∈ R_u do
            if d' ∈ NFP_{tr,us} then
                Φ^d_{t[r],u[s]} = Φ^d_{t[r],u[s]} ⋃ d';
            end
        end
    end
end
return Φ^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 4.8(b)). Piece p_{tr} is then translated vertically by $w^{top}_{tr} + w^{bottom}_{tr}$ and the same procedure is applied until p_{tr} reaches the uppermost admissible position on the board (Figure 4.8(c)). Piece p_{tr} is then translated horizontally by $l^{left}_{tr} + l^{right}_{tr}$ and the whole process is repeated as long as p_{tr} can be moved horizontally or vertically (Figure 4.8(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.

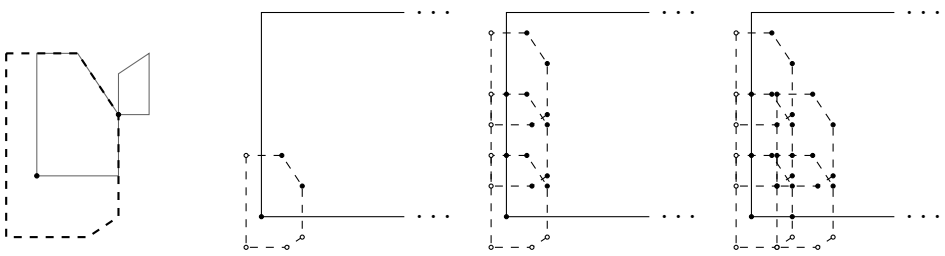


Figure 4.8: An example of an *NFP*-based mesh for the piece types presented in Figure 4.7.

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.

The procedure to generate the dot list using the *NFP*-based mesh is presented in Algorithm 5. Algorithm 4 can be used to generate the intersection list of each dot.

Algorithm 5: *NFP*-BASED MESH GENERATION

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

$D[\ ] \leftarrow \emptyset$;
for $t \in T$ *and* $u \in T$ **do**
  for $r \in R_t$ *and* $s \in R_u$ **do**
    $\Delta x_{tr} = l_{tr}^{left} + l_{tr}^{right}$;
    $\Delta y_{tr} = w_{tr}^{top} + w_{tr}^{bottom}$;
    $x = l_{tr}^{right}$;
    while $x \leq L - l_{tr}^{right}$ **do**
      $y = w_{tr}^{top}$;
      while $y \leq W - w_{tr}^{bottom}$ **do**
        $d = (x, y)$;
        $D[\ ] = \text{UPDATE\_DOT\_LIST}(D[\ ], d, t, r)$;
        for $d' \in NFP_{tr,us}$ **do**
          if $d' \in IFP_{us}$ **then**
            $D[\ ] = \text{UPDATE\_DOT\_LIST}(D[\ ], d', u, s)$;
          end
        end
        $y = y + \Delta y_{tr}$;
      end
      $x = x + \Delta x_{tr}$;
    end
  end
end
return $D[\ ]$;

4.5 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 pieces list, and (ii) the time needed to analyze the overlapping of the pieces 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, $t \in T$, $r \in R_t$. The number of dots generated on the board is approximately $\frac{W \times L}{\Delta x_{tr} \times \Delta y_{tr}}$, and therefore the maximum number of dots considered on the board for any given piece is $\frac{2W \times L}{\overline{\Delta}}$, 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 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 (4.7),

$$O\left(|T| \times \overline{R} \times \frac{2W \times L}{\overline{\Delta}} + |T| \times \overline{R} \times \frac{\overline{V}^2}{2}\right) \tag{4.7}$$

where $|.|$ is the number of elements of the set, and $\overline{R} = \max t \in T\{|R_t|\}$. Notice that the width of the board W 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 pieces t at rotation r and u at rotation s, 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 $\Delta x_{tr} = l_{tr}^{left} + l_{tr}^{right}$ and $\Delta y_{tr} = w_{tr}^{top} + w_{tr}^{bottom}$ 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 (4.8) shows the worst case complexity to generate the phase (i) dots.

$$O\left(|T|^2 \times \overline{R}^2 \times \frac{W \times L}{\overline{\Delta}} \times \overline{V}_{NFP}\right) \tag{4.8}$$

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 (4.9).

$$O\left(\overline{T} \times \overline{R} \times \frac{W \times L}{\Delta^2}\right) \tag{4.9}$$

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 pieces placed at different dots. In other words, for each piece $t \in T$ at rotation $r \in R_t$ and piece $u \in T$ at rotation $s \in R_u$ the overlapping needs to be analyzed for each possible combination of dots where these pieces 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 (4.10).

$$O\left(|D|^2 \times |T|^2 \times \overline{R^2} \times \overline{V}_{NFP}\right) \tag{4.10}$$

Hence, regarding the worst case, the time complexity to build the structure is given by the sum of Formula (4.10) with either Formula (4.7), Formula (4.8) or Formula (4.9), 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 4.1b and 4.1c, a piece-rotation list is associated with each dot in this dot list, which includes all pieces 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-rotation list, which includes all those dots where the placement of any given second piece 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 (4.11) represents thus the amount of memory used by the proposed structure.

$$O\left(|D| \times |T|^2 \times \overline{R}^2 \times \overline{\Phi}\right) \tag{4.11}$$

4.6 Computational experiments

This section presents the computational results obtained with the dotted-board model using the different meshes proposed in Section 4.4, and compares the results with those obtained with the regular mesh used in Toledo et al. (2013). The adopted instances aim

at demonstrating the advantages of the proposed meshes in the resolution of problems with specific characteristics.

4.6.1 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 4.4, 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 the MF (Metalworking/Furniture) instance set is proposed. The instances in this set are composed of five piece types as presented in Figure 4.9.

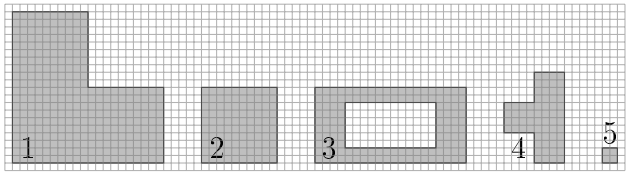


Figure 4.9: The MF instance set of pieces.

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 requirements by i. For all these instances, the board width 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 4.10, compose the instances of this set, and each piece type is allowed four rotations.

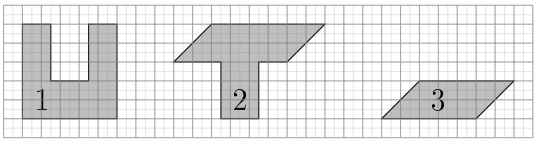


Figure 4.10: The CS instance set of pieces.

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 4.10. 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 4.10 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 4.1. 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 4.1: 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	Hopper (2000)
shapes0	43	4	0	4	40	73	Oliveira and Ferreira (1993)
shapes1	43	4	180	6	40	70	Oliveira and Ferreira (1993)
shirts	99	8	180	14	40	78	Dowsland et al. (1998)
blaz1	28	7	180	10	15	34	Oliveira et al. (2000)
blaz2	16	4	180	5	15	26	Oliveira et al. (2000)
jakobs1	25	22	90	62	35	15	Jakobs (1996)
jakobs2	25	21	90	63	70	29	Jakobs (1996)
fu	12	11	90	34	38	39	Fujita et al. (1993)

4.6.2 Piece-based mesh computational results

The piece-based mesh presented in Section 4.4.1 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 $\frac{1}{8} \times (l_{tr}^{left} + l_{tr}^{right})$ and $\frac{1}{8} \times (w_{tr}^{top} + w_{tr}^{bottom})$ respectively.

The computational results for the dotted-board model using the piece-based mesh are presented in Table 4.2. 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 $\Delta = 2$.

Table 4.2: Computational experiments using the piece-based mesh generation rule.

	Piece-based mesh					Regular mesh ($\Delta = 2$)				
	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	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.

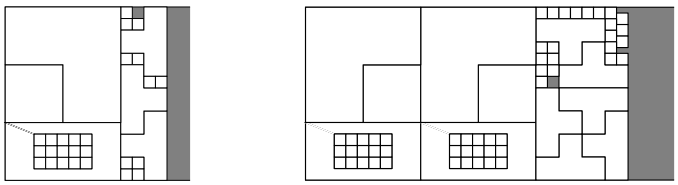
Using the piece-based mesh, optimality was proven for all instances except MF10 and MF15, whereas with the regular mesh with $\Delta = 2$, optimality could only be proven for the MF1 instance. Moreover, in the case of the regular mesh, only for the smallest instances, MF1 to MF5, was there a solution 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.

Using a regular mesh of granularity $\Delta = 1$, 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 mesh generation rule could not produce a feasible solution within the 3600 seconds time limit. 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 solutions - 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.

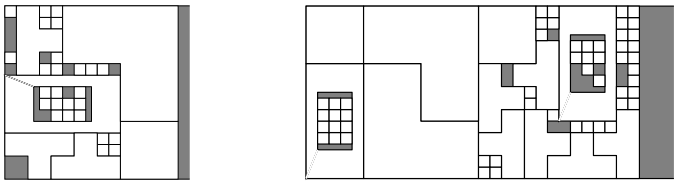
For the instances used, the model built using the piece-based mesh found solutions for all instances, and for problems with almost the same number of variables, optimality was proven quicker. For example, for instance MF5, the mesh by pieces generates a model with 6594 variables, equivalent to the 6541 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 optimal solutions for the MF1 and MF2 instances obtained using the piece-based mesh are represented in Figures 4.11(a) and 4.11(b), respectively. Figures 4.12(a) and 4.12(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 4.13 represents the optimal solution for the MF8 instance obtained with the piece-based mesh.



(a) MF1. (b) MF2.

Figure 4.11: Optimal solutions for the MF1 and MF2 instances, obtained with the piece-based mesh.



(a) MF1. (b) MF2.

Figure 4.12: Optimal solutions for the MF1 and MF2 instances, obtained with the regular mesh.

Figure 4.13: Solution for the MF8 instance obtained with the piece-based mesh.

4.6.3 *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 4.4.2) 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 the most appropriate regular mesh, with $\Delta = 0.5$.

Table 4.3 presents the results obtained with the models based on the *NFP*-based mesh, and on a regular mesh of granularity $\Delta = 0.5$. The columns have the same type of content as described in Table 4.2.

Table 4.3: 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%	3600.0	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.

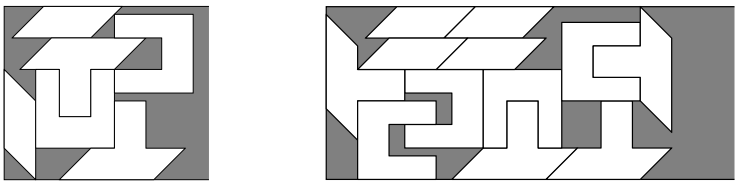
Both mesh generation rules, the piece-based mesh one and the *NFP*-based mesh rule, use information about the pieces to build the mesh. However, the *NFP*-based mesh uses information that relates to the interaction between pairs of pieces rather than to individual pieces, as happens with the piece-based 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 pieces are used to generate the dots in the mesh, the intersection between these two pieces is naturally avoided. This reinforces the statement that the right choice of placement dots for each piece can speed up the convergence time of the model.

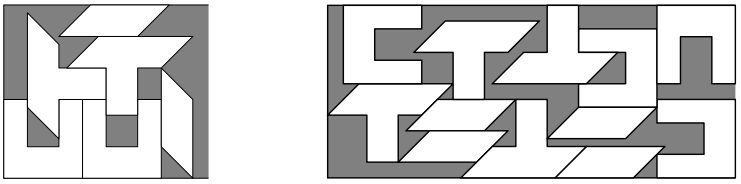
By using the regular mesh with $\Delta = 0.5$, an optimal solution for the CS1 instance with the same quality as the one found with the *NFP*-based mesh model is reached. Still, the computational time to find and prove the optimality of this solution is more than five hundred times faster in the case 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.

Figures 4.14(a) and 4.14(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 4.15(a)) and a feasible solution was found for the CS2 instance (Figure 4.15(b)). Figure 4.16 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.

Figure 4.14: Optimal solutions for the CS1 and CS2 instances obtained with the *NFP*-based mesh.



(a) CS1. (b) CS2.

Figure 4.15: Optimal solution for the CS1 instance and a feasible solution for CS2 instance obtained with the regular mesh.

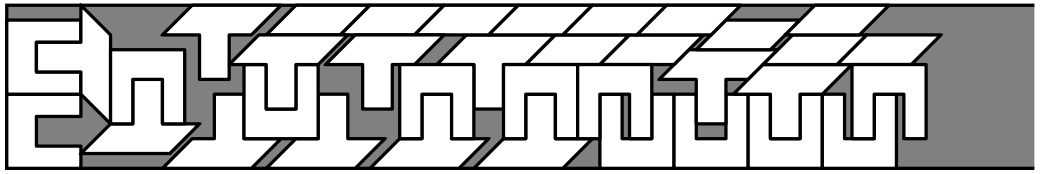


Figure 4.16: Solution for the CS6 instance obtained by the *NFP*-based mesh.

4.6.4 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 4.4 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 4.2. 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 4.4: 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.

4.6.5 Computational experiments with instances from the literature

This section presents the computational results obtained for the dotted board model using the meshes proposed in Section 4.4 and those achieved with a regular mesh as used in Toledo et al. (2013). The experiments were performed with different instances, showing the performance of each mesh generation rule with different instances.

Table 4.5 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 4.5: Computational experiments using various mesh generation rules.

	Regular mesh			*NFP*-based mesh			Piece-based mesh		
Instance	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**	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.

4.7 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.

Chapter 5

A model based heuristic for the irregular strip packing problem[1]

In the last decade, sophisticated heuristics have been proposed to solve the irregular strip packing problem. Examples of these heuristics are found in Elkeran (2013), Sato et al. (2012), Umetani et al. (2009), Imamichi et al. (2009). Notwithstanding the number of heuristics proposed to solve this problem, none of them combine mixed-integer programming models (Toledo et al., 2013; Alvarez-Valdes et al., 2013) with heuristic procedures.

Heuristics and exact methods have been successfully combined to solve combinatorial optimization problems (Maniezzo et al., 2009). To solve the generalized assignment problem, the method by Woodcock and Wilson (2010) combines tabu search, linear programming and branch and cut. Flisberg et al. (2009) used tabu search, linear programming and branch and cut to solve the vehicle routing problem. Bennell and Dowsland (2001) solved the irregular strip packing problem combining tabu search and linear programming. An approach that uses a branch and cut method to explore several large neighbourhood structures in order to solve the lot-sizing problem was presented by Muller et al. (2012).

In this chapter, a model based heuristic is presented to solve the irregular strip packing problem. The heuristic combines the dotted-board model (Toledo et al., 2013) with a new compaction model inspired on the model by Alvarez-Valdes et al. (2013). The data structure presented in Chapter 4 is used to handle the geometry of the problem when the dotted board model is used.

In Section 5.1, the model based heuristic is detailed. Computational experiments were performed in order to evaluate the performance of the method and are presented in Section 5.2. Section 5.3 presents some conclusions of the chapter.

[1]The text of this chapter is strongly based on the paper "A model based heuristic for the irregular strip packing problem" which is under review.

5.1 3–Phase Matheuristic (3PM)

The proposed matheuristic is based on two mathematical models, the dotted board model (Toledo et al., 2013) and a linear model developed from the compaction ideas used by Gomes and Oliveira (2006). Our objective is to obtain good-quality solutions to the problem in a short computational time. The solution method can be summarized into three phases:

- Constructive phase: finding an initial feasible solution to the problem using the dotted board model;
- Improvement phase: using uses the dotted board model to improve the initial solution;
- Compaction phase: improving the best solution found so far using the linear model.

In the subsections (5.1.1)-(5.1.3), these three phases are described in detail.

5.1.1 3PM – Constructive phase

The objective of the constructive phase is to iteratively build an initial feasible solution for the problem. The grid used has a minimum resolution, large enough to ensure a good trade-off between the computational time and the solution quality. Instead of using the regular grid as used in Toledo et al. (2013), the piece-based mesh is used (section 4.4.1). In this chapter we assume that $bx_{tr} = by_{tr} = g_{min}$. The g_{min} value was defined through preliminary computational experiments.

This phase is based on the relax-and-fix strategy. Consider the decision variables δ_{tr}^{d} which are 1 (one) if a piece of type t is assigned to dot d and 0 (zero) otherwise. These variables are split into four sets:

Γ) set of variables associated with fixed pieces, i.e., pieces that are already fixed on the board;

Δ) set of variables associated with positioned pieces, i.e., pieces that are previously positioned on the board, but can perform some movements;

Θ) set of variables associated with free pieces, i.e., pieces that can be freely positioned on the board;

Ω) set of variables associated with waiting pieces, i.e., pieces that are not considered in the current step.

Figure 5.1 illustrates the constructive phase steps. Initially, sets Γ, Δ and Θ are empty, and therefore, a small number of pieces associated with the set Ω are assigned to set Θ. In each step, a sub-problem defined according to the Γ, Δ and Θ sets is solved. In order to build each sub-problem, consider μ' and μ, the upper bounds of the last two sub-problems. Initially, these parameters are defined as zero. At the end of each step, sets Γ and Δ are redefined based on the parameters μ' and μ. The pieces with the reference point positioned in the interval $[0, \mu')$ define set Γ. The pieces with the reference point positioned in the interval $[\mu', \mu]$ define set Δ. Finally, a small number of pieces from set Ω are moved to set Θ. In the final step, set Θ is empty. By solving the associated sub-problem, a feasible solution to the problem is obtained. In Figure 5.1, the pieces associated with sets Γ and Δ are represented in black and dark gray, respectively. Pieces above the board comprise set Ω, and the number of times that pieces must be inserted in the board is indicated below each piece. The pieces at the right-hand side of the board represent set Θ. Figure 5.1a shows an example that consists of seven piece types. In Figure 5.1b, some pieces from the set Ω are selected to form set Θ. The solution of the Figure 5.1b sub-problem and the new set Θ is presented in Figure 5.1c. The pieces positioned on the board in Figure 5.1c comprise set Δ. Figure 5.1d shows the solution of the previous sub-problem where the pieces with the reference point in the interval $[0, \mu)$ are fixed and new pieces are positioned on the board. The complete problem solution is presented in Figure 5.1e.

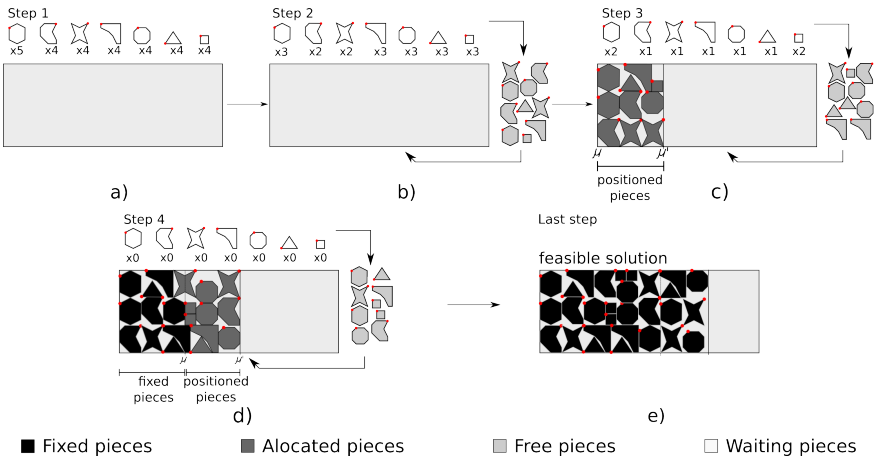


Figure 5.1: Steps of the constructive phase.

In each step, a subset of elements from set Ω is selected for set Θ. The size of these subsets is σ, a number small enough to provide a fast solution and big enough for the pieces to fit well. Furthermore, the size is calculated so as to reduce the difference between the sizes of the subsets, and details are provided in Section 5.2.1. To form each set Θ, the pieces are included one by one in the subset. The piece type selected is the one with the largest rate:

$$\frac{\text{number of pieces of type } t \text{ in the set } \Theta}{\text{number of pieces of type } t \text{ in the problem } (q_t)}, \qquad \forall t \in T.$$

This criterion was used in order to homogeneously distribute the different piece types in the solution.

To define each sub-problem model, consider subsets $\mathcal{M} \subset \mathcal{D}$ and $\mathcal{W} \subset \mathcal{D}$ containing the board dots in the intervals $[0, \mu')$ and $[\mu', \mu]$, respectively. The previous step solution is defined by $\overline{\delta_{tr}^d}$, $d \in \mathcal{D}$, $t \in T, r \in R_t$. Note that $\Gamma = \{(d,t), \overline{\delta_{tr}^d} = 1, d \in \mathcal{M}, t \in T, r \in R_t\}$ and $\Delta = \{(d,t,r), \overline{\delta_{tr}^d} = 1, d \in \mathcal{W}, t \in T, r \in R_t\}$. The partial demand is represented by $\overline{q_t}$, $t \in T$, that is, the number of pieces of type t in the sets Γ, Δ and Θ. Finally, α_t is the number of pieces of type t with the reference point in subset $\mathcal{W}$. The sub-problem model is given by (5.1)-(5.5):

$$\text{min: } L \tag{5.1}$$

$$\text{s. t.: } (4.2), (4.3), (4.5), (4.6), \tag{5.2}$$

$$\sum_{d \in D_{t[r]}} \delta_{tr}^d = \overline{q_t}, \qquad t \in T, \tag{5.3}$$

$$\sum_{r \in R_t} \left(\sum_{\bar{\delta}_{tr}^d = 1, d \in \mathcal{W}} (1 - \delta_{tr}^d) + \sum_{\bar{\delta}_{tr}^d = 0, d \in \mathcal{W}} \delta_{tr}^d \leq \alpha_t \right), \qquad t \in T, \tag{5.4}$$

$$\delta_{tr}^d = 1, \qquad t \in T, r \in R_t, \bar{\delta_{tr}^d} = 1, \; d \in D_{t[r]} \cap \mathcal{M}. \tag{5.5}$$

In the model (5.1)-(5.5), constraints (5.3) ensure that the partial demand will be met. Constraints (5.4) restrict the movements over the variables of set $\mathcal{W}$. Specifically, one move is counted when a piece previously allocated in set $\mathcal{W}$ is moved outside set $\mathcal{W}$ or when a piece from set Θ is allocated into set $\mathcal{W}$. Two moves are counted when a piece previously allocated in set $\mathcal{W}$ is moved into set $\mathcal{W}$. The upper bound for the moves is α_t. Constraints (5.5) fix to the board the pieces with the reference point positioned on a dot from the set $\mathcal{M}$. Algorithm 6 summarizes the constructive phase.

5.1.2 3PM – Improvement phase

The Improvement Phase starts with the solution of the Constructive Phase and it is also performed in steps. In the first step, g_{min} is equal to that used in the constructive phase, and after each step, g_{min} is divided by two. Note that g_{min} is only a lower bound of the grid resolution value. At the end of each step, the dots that contain reference points of pieces allocated are included into the grid of the next step. This ensures that

Algorithm 6: Constructive phase

Input: Sets D, T and Ω;
Output: A feasible solution $\overline{\boldsymbol{\delta}} = \{\overline{\delta_{tr}^{d}} | d \in D_{t[r]},\ t \in T, r \in R_t\}$;
Initialize:
  Calculate σ (number of pieces to compose Θ);
  Do $\overline{\boldsymbol{\delta}} = \mathbf{0}$, $\mu' = \mu = 0$;
Constructive phase:
  While ($\Omega \neq \emptyset$)
    Define the subsets $\mathcal{M}$ and $\mathcal{W}$;
    Do $\Theta = \emptyset$;
    Remove $\min\{\sigma, |\Omega|\}$ pieces from the set Ω and insert them into the set Θ;
    Solve the sub-problem (5.1)-(5.5) obtaining the solution $\overline{\boldsymbol{\delta}}$ with value $\overline{L}$;
    Do $\mu' = \mu$ and $\mu = \overline{L}$;
  Return $\overline{\boldsymbol{\delta}}$ as solution.

the best solution found so far is feasible for the next step and leads to a good initial solution for the search. The search ends when g_{min} is smaller than a threshold mr. In each step, a variable neighborhood descent heuristic (VND) is applied to improve the quality of the best solution found so far.

The VND heuristic is defined by applying successive local search procedures over K different neighborhoods. The choice of a neighborhood is performed in a deterministic way. A final solution is a local optimum with respect to all K neighborhoods. The neighborhoods are defined allowing the pieces to move in the dots that are inside a small board region around their position in the previous step solution $\overline{\boldsymbol{\delta}}$. The shape of these regions defines the neighborhood that will be explored during the search. The first neighborhood is a small square with its center in the dot where the piece was positioned. The second neighborhood is a rectangle with the same width as that of the board and length chosen so that the number of dots in each region is limited by md. Finally, the third neighborhood is a rectangle with the board length and the width calculated to be similar to the length of the second region. Figure 5.2 illustrates these three neighborhoods, where the dot represents the piece reference point and the highlighted rectangle, the region where this reference point can move to another dot.

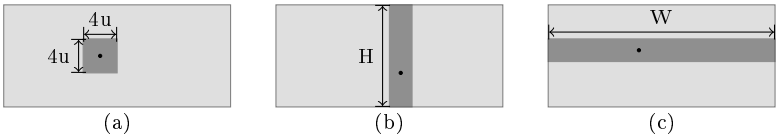


Figure 5.2: The first (a), second (b) and third (c) neighborhood for a piece reference point.

These three neighborhoods were chosen in order to explore a diversified set of dots and then find better solutions. The first region is a small region and hence results in

a fast search. To make the pieces fit better, the second region is created by reaching all the vertical contact areas between two pieces that are near each other. Finally, the third neighborhood aims to change the piece's position over the layout length.

The order of neighborhoods is obtained using a deterministic procedure. We start by choosing the first neighborhood to restrict the feasible piece placement and then solve the problem. If the solution is not better than the best solution found so far, then the second neighborhood structure is applied. If the search over the second neighborhood structure does not improve the solution quality, then the third neighborhood structure is applied. If the third neighborhood does not improve the solution quality, then the step is terminated. During the process, if any of the three neighborhoods yields a solution better than the best solution found so far, then the search process returns to the first neighborhood and the process is restarted.

Each neighborhood can be represented by a model. Consider Λ_{tr}^{d} as the set of dots inside a given board region around the dot d where the reference point of piece t at rotation r is allocated. The size of these regions is defined as described above. The neighborhood model is given as follows.

$$\text{min: } L \tag{5.6}$$

$$\text{s. t.: } (4.2), (4.3), (4.4), (4.5), (4.6), \tag{5.7}$$

$$\delta_{tr}^{d} = 0, \qquad d \in \{D_{t[r]} - \bigcup_{t \in T, r \in R_t} \Lambda_{tr}^{d} \mid \overline{\delta_{tr}^{d}} = 1\}, t \in T, r \in R_t. \tag{5.8}$$

Constraints (5.8) limit the search domain to move each piece within a rectangular region over the board. Given a feasible solution $\overline{\boldsymbol{\delta}}$, the best solution of the model (5.6)-(5.8) is its best neighbor.

When there are no more neighborhoods to explore in VND, the grid is refined. With more dots to represent the board, there is a new range of feasible placement positions for each piece. The VND heuristic is performed again to improve the solution further. Algorithm 7 summarizes the improvement phase.

5.1.3 3PM – Compaction Phase

As the solution obtained in the improvement phase has the piece reference points positioned on specific dots, gaps may appear between pieces. Taking this into account, a compaction of this solution is essential to move the pieces as close as possible to each other. To compact a solution, we use the mixed-integer linear model based on Gomes

Algorithm 7: Improvement phase

Input: Set $\mathcal{T}$; initial resolution g_{min}; threshold mr; a solution $\overline{\boldsymbol{\delta}}$ and its value $\overline{L}$;
Output: Improved solution $\overline{\boldsymbol{\delta}}$;
Initialize:
  Choose the first neighborhood, $Neigh = 1$;
Improvement phase:
  While ($g_{min} > mr$)
    Define $\mathcal{D}$ using g_{min};
    Add the dots of $\overline{\boldsymbol{\delta}}$ to $\mathcal{D}$;
    While ($Neigh \leq 3$)
      Find $\boldsymbol{\delta'}$ the best neighbor solution of $\overline{\boldsymbol{\delta}}$ using the neighbourhood $Neigh$;
      If ($L' \geq \overline{L}$), Do $Neigh = Neigh + 1$;
      else, Do $Neigh = 1$; $\overline{\boldsymbol{\delta}} = \boldsymbol{\delta'}$; $\overline{L} = L'$;
    Do $g_{min} = g_{min}/2$;
  Return solution $\overline{\boldsymbol{\delta}}$;

and Oliveira (2006) with some additional constraints. In this model the positioning of each piece reference point is linear and is represented by a pair of real variables (x_i, y_i). To avoid overlaps between pieces i and j, the authors consider the set E_{ij} set with all the lines that contain an edge of NFP_{ij}; then, an integer variable v_{ije} is used to ensure that the pieces are on different sides of at least one of the lines $e \in E_{ij}$. More details on this model can be found in Gomes and Oliveira (2006) and Alvarez-Valdes et al. (2013). In order to define the additional constraints to be added to the Gomes and Oliveira (2006) model, consider the pieces individually, i.e., each piece is mapped according to its type by the integers of the interval $((t-1) \times d_t, t \times d_t]$. The total number of pieces is given by $\mathcal{N} = \sum_{t \in T} d_t$. All the pieces can be found on the interval $[1, \mathcal{N}]$. In addition, consider $\overline{x_i}$ ($\overline{y_i}$), $i = 1, ..., \mathcal{N}$, the position on the x-axis (y-axis) for the piece t at rotation r placed in $\overline{\delta}^d_{tr} = 1$, $d \in D_{t[r]}$. The new constraints imposed in Gomes and Oliveira (2006) model ensure that the pieces can move only over a small region of the board. These regions are defined as squares around the points where each piece is positioned. The side λ_i of each square is given based on the size of the bounding box of each piece i and the number of pieces allocated and is defined in Section 5.2.1.

The Compaction Phase Model (3PM-CPM) is given as (5.9)-(5.15):

$$\text{min: } L \tag{5.9}$$

$$\text{s. t.: } l_i^{left} \leq x_i \leq L - l_i^{right}, \qquad i = 1, ..., \mathcal{N}, \tag{5.10}$$

$$w_i^{top} \leq y_i \leq W - w_i^{bottom}, \qquad i = 1, ..., \mathcal{N}, \tag{5.11}$$

$$\alpha_{ije}(x_j - x_i) + \beta_{ije}(y_j - y_i) \leq \gamma_{ije} + M(1 - v_{ije}), \qquad 1 \leq i < j \leq \mathcal{N}, \forall e \in E_{ij}, \tag{5.12}$$

$$\sum_{e \in E_{ij}} v_{ije} \geq 1, \qquad 1 \leq i < j \leq \mathcal{N}, \tag{5.13}$$

$$\overline{x_i} - \lambda^i \leq x_i \leq \overline{x_i} + \lambda^i, \qquad i = 1, ..., \mathcal{N}, \tag{5.14}$$

$$\overline{y_i} - \lambda^i \leq y_i \leq \overline{y_i} + \lambda^i, \qquad i = 1, ..., \mathcal{N}, \tag{5.15}$$

$$v_{ije} \in \{0, 1\}, \qquad \forall i, j = 1, ..., \mathcal{N}, \forall e \in E_{ij}, \tag{5.16}$$

$$x_i, y_i \geq 0, \qquad i = 1, ..., \mathcal{N}, \tag{5.17}$$

$$L \geq 0. \tag{5.18}$$

where α_{ije}, β_{ije} and γ_{ije} are the coefficients of the line e associated with an edge of NFP_{ij} and M is large enough to make the constraint (5.12) a dummy constraint if $v_{ije} = 0$.

Constraints (5.10) associated with (5.9) define the objective function. Constraints (5.10) and (5.11) ensure that the piece is entirely inside the board, and constraints (5.12) and (5.13) guarantee that the pieces do not overlap. Constraints (5.14) and (5.15) allow the piece to move only within a given square. Finally, the variable domains are given by (5.16), (5.17) and (5.18).

The compaction phase is an iterative process, i.e., if an improved solution is found at the end of the compaction, the compaction is executed again, starting from this improved solution. Algorithm 8 presents an outline of the compaction phase.

Algorithm 8: Compaction phase

Input: Sets $\mathcal{D}$ and T; a feasible solution $\overline{\delta}$;
Output: Compacted solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$;
Initialize:
  Obtain $\overline{\mathbf{x}}$, $\overline{\mathbf{y}}$ and $\overline{L}$ from $\overline{\delta}$;
Compaction phase:
  $L' = 0$;
  While $(L' < \overline{L})$
    Solve the model (5.9) - (5.15) obtaining the solution $\mathbf{x}'$, $\mathbf{y}'$ and length L';
    Do $\overline{\mathbf{x}} = \mathbf{x}'$ and $\overline{\mathbf{y}} = \mathbf{y}'$;
  Return $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ as solution.

5.2 Computational results

The computational experiments were performed on an Intel(R) Xeon(R) E5-2620 2.00 GHz processor with 64 GB of memory running an Ubuntu 12.04 operating system. The methods were implemented in the C/C++ programming language, and the mathematical models were solved using IBM ILOG CPLEX 12.5. To perform the tests, instances from the literature, presented in Table 5.1, were used. The first column presents the instance name. Columns two and three present the number of piece types and the absolute number of pieces, respectively. The available rotations for the pieces and the width of the board are, respectively, presented in columns four and five. Finally, column six presents the origin of the instance.

Table 5.1: Instances used in the benchmark.

Name	**Piece types**	**Pieces quantity**	**Rotations**	**Board width**	**Reference**
Albano	25	25	0,90,180,270	4900	Albano and Sapuppo (1980)
Mao	9	20	0,90,180,270	2550	Bounsaythip and Maouche (1997)
Marques	8	24	0,90,180,270	104	Marques et al. (1991)
Trousers	17	64	0,180	79	Oliveira and Ferreira (1993)
Jakobs1	25	25	0,90,180,270	40	Jakobs (1996)
Jakobs2	25	25	0,90,180,270	70	Jakobs (1996)
Fu	12	12	0,90,180,270	38	Fujita et al. (1993)
Poly1a0	15	15	0	40	Hopper (2000)
Shapes0	4	43	0	40	Oliveira et al. (2000)
Shapes1	4	43	0,180	40	Oliveira et al. (2000)
Shapes2	7	28	0,180	15	Oliveira et al. (2000)
Blaze<i>	7	$7\times < i >$	0	15	Toledo et al. (2013)
Shapes_T$< i >$	4	8	0	40	Toledo et al. (2013)
RCO$< i >$	7	$< i >$	0	15	Toledo et al. (2013)
Shapes_AV4	4	4	0	20	Alvarez-Valdes et al. (2013)
Shapes_AV8	4	8	0	13	Alvarez-Valdes et al. (2013)
Fu$< i >$	$< i >$	$< i >$	0	38	Alvarez-Valdes et al. (2013)
threep$< i >$w$< j >$	3	$3\times < i >$	0	$< j >$	Alvarez-Valdes et al. (2013)

The following subsection presents the parameters used to run the matheuristic. An analysis of the proposed matheuristic showing the influence of each phase in the solution method is presented in Subsection 5.2.2. The proposed matheuristic performance is compared with exact models and heuristic methods in Subsections 5.2.3 and 5.2.4, respectively.

5.2.1 Defining parameters and sets

In this section, the parameters used in the matheuristic are defined. These parameters were chosen based on preliminary computational experiments and on the features of each instance.

The initial value of g_{min} is two in order to generate a grid with a limited number of dots. The idea is to lead the constructive phase to quickly obtain a solution. This parameter can generate some gaps among the pieces, but these gaps should be reduced in the improvement phase.

In each step of the constructive phase, σ elements of Ω must be selected to form Θ. The idea is to define σ such that the subsets Θ of each iteration have a similar number of elements. After preliminary tests, we verified that problems with five pieces or less are solved very fast using the model. These tests also show that problems with more than 12 pieces are difficult to solve within a time span adequate for a constructive phase. The value of sigma is defined as stated in Algorithm 9, where a mod b is the remainder of the division of a by b.

Algorithm 9: Defining sigma

Input: Set Ω, demands q_t;
Output: σ;

If ($|\Omega| \leq 5$)
    Return ($|\Omega|$);
Else
    Do $\sigma = 5$
    For ($s = \min\{12, \sum_{t \in T} q_t\}$ to 6) do
        If ($\sum_{t \in T} q_t$ mod $s = 0$)
            Return (s);
        Else if ($\sum_{t \in T} q_t$ mod $s \geq \sum_{t \in T} q_t$ mod σ)
            Do $\sigma = s$;
Return (σ);

After the constructive phase, the improvement phase runs while $g_{min} \geq mr$, where $mr = 0.5$ to make the pieces closer to each other. The number of dots in each neighborhood of the improvement phase must not be larger than the parameter md. In the initial tests, $md = 3000$, which results in the improvement model performing a fast local search.

Several preliminary tests were run to determine the value of λ^i for each piece of type i, where λ^i is a parameter of the compaction phase model (CPM). Depending on the position of the pieces and on the size of the region where these pieces can move, a pair of pieces could even change their relative positions. These values are based on i) the number of pieces in the instance and ii) the size of the piece bounding box.

- For instances with less than 13 pieces the square around the reference point of piece i has a side equal to $\lambda^i = \max(l_i^{left} + l_i^{right}, w_i^{bottom} + w_i^{tom})$;

- For instances with 13 to 20 pieces the square around the reference point of piece i has a side equal to $\lambda^i = \max(l_i^{left} + l_i^{right}, w_i^{bottom} + w_i^{top})/2$;

- For instances with more than 20 pieces the square around the reference point of piece i has a side equal to $\lambda^i = \min(l_i^{left} + l_i^{right}, w_i^{bottom} + w_i^{top})/2$.

Initial tests show that these square side sizes presented a good trade-off between the solution quality and computational time. Depending on the position of the pieces and on the size of the region where these pieces can move, a pair of pieces could even change their relative positions.

5.2.2 Matheuristic phases analysis

To demonstrate the importance of each phase of the proposed method, in Table 5.2, we summarize the results obtained at each phase. The first column presents the instance name. In columns two and three, the constructive phase solution and time are shown. Columns four and five show the improvement phase solution and its time, respectively. The improvement rate, the time increase and the percentage by which the computational time increases from the constructive phase to the improvement phase are depicted in columns six, seven and eight. In columns nine and ten, the solution value and its computational time are presented. Columns eleven, twelve and thirteen describe the improvement rate, the additional time and the percentage that the computational time increases compared with the constructive plus improvement phases.

As expected, the constructive phase obtained a solution with poor quality in a shorter computational time. Applying the improvement phase to the constructive phase solution on average leads to 19% improvement in the solution quality. The computational time increases by 171.8 seconds on average, varying from 0.1 to 2301 seconds depending on the instance.

Using the complete proposed method, the best results obtained by the search were found. On average, the solutions found by the complete matheuristic are 9.7% better when compared to the solutions found by the improved constructive phase. Furthermore, the computational time increases by 200 seconds on average, varying from 0.1 to 1294 seconds depending on the instance. Specifically, the compaction phase leads on average to 9.7% improvement in the solution quality; however, the computational time doubles.

Based on the results, it can be concluded that the compaction phase is essential to eliminate the grid dependence of the other phases of the method. Moreover, if a fast solution that has good quality is needed and less computational time is available, the construction phase followed by the improvement phase should be used. If a more

Table 5.2: Different phases of the proposed solution method.

	Construct. phase		Construct.+ improvement			Add.	Time	3PM			Add.	Time
Instance	**Sol.**	**Time**	**Sol.**	**Time**	**Impr.**	**time**	**inc.(%)**	**Sol.**	**Time**	**Impr.**	**time**	**inc.(%)**
Blaze1	18.0	0.1	12.0	1.9	33.3%	1.8	94.7%	7.4	23.3	38.3%	21.5	92.3%
Blaze2	16.0	1.0	14.0	8.7	12.5%	7.7	88.5%	14.0	68.9	0.0%	60.2	87.4%
Blaze3	24.0	1.6	21.0	38.8	12.5%	37.2	95.9%	20.5	340.2	2.6%	301.4	88.6%
Blaze4	32.0	2.2	29.0	35.0	9.4%	32.7	93.4%	27.9	517.6	3.9%	482.6	93.2%
Blaze5	40.0	2.7	34.0	153.5	15.0%	150.8	98.2%	34.0	395.2	0.1%	241.7	61.2%
Shapes_T2	30.0	1.2	16.0	7.0	46.7%	5.8	82.9%	14.0	8.1	12.5%	1.1	13.6%
Shapes_T4	30.0	4.1	30.0	10.4	0.0%	6.3	60.6%	26.0	201.5	13.3%	191.1	94.8%
Shapes_T5	36.0	7.5	31.0	45.6	13.9%	38.1	83.6%	31.0	106.0	0.0%	60.4	57.0%
Shapes_T7	60.0	9.9	42.0	116.1	30.0%	106.2	91.5%	42.0	176.3	0.0%	60.3	34.2%
Shapes_T9	71.0	10.2	48.0	170.6	32.4%	160.3	94.0%	48.0	292.3	0.0%	121.7	41.6%
RCO1	14.0	0.1	8.0	4.2	42.9%	4.1	97.6%	8.0	44.4	0.0%	40.3	90.8%
RCO2	16.0	0.5	16.0	2.1	0.0%	1.6	76.2%	15.0	254.5	6.3%	252.5	99.2%
RCO3	24.0	1.2	24.0	4.6	0.0%	3.4	73.9%	22.0	264.7	8.3%	260.1	98.3%
RCO4	32.0	2.1	30.0	33.0	6.3%	30.9	93.6%	29.0	83.1	3.3%	50.1	60.3%
RCO5	40.0	4.7	37.0	29.8	7.5%	25.0	83.9%	36.7	210.2	0.9%	180.5	85.9%
Shapes_AV4	24.0	0.6	24.0	2.1	0.0%	1.6	76.2%	24.0	2.2	0.0%	0.1	4.5%
Shapes_AV8	41.0	0.8	30.0	5.5	26.8%	4.7	85.5%	26.0	186.9	13.3%	181.4	97.1%
Fu5	20.0	0.1	20.0	0.3	0.0%	0.2	66.7%	17.9	1.0	10.6%	0.7	70.0%
Fu6	56.0	0.1	28.0	30.5	50.0%	30.4	99.7%	23.0	31.7	17.9%	1.2	3.8%
Fu7	70.0	0.1	28.0	3.9	60.0%	3.9	100.0%	24.0	5.0	14.3%	1.1	22.0%
Fu8	49.0	0.1	28.0	19.5	42.9%	19.4	99.5%	24.0	20.9	14.3%	1.4	6.7%
Fu9	56.0	0.1	30.0	26.7	46.4%	26.5	99.3%	25.0	52.3	16.7%	25.7	49.1%
Fu10	42.0	0.2	30.0	25.7	28.6%	25.5	99.2%	28.7	265.9	4.4%	240.3	90.4%
Fu12	45.0	0.4	42.0	2.1	6.7%	1.7	81.0%	33.5	186.7	20.4%	184.6	98.9%
threep1w7	6.5	0.6	6.5	1.4	0.0%	0.8	57.1%	6.0	2.5	7.7%	1.0	40.0%
threep2w7	13.5	0.3	11.5	2.4	14.8%	2.1	87.5%	9.3	12.4	18.9%	10.1	81.5%
threep3w7	20.0	0.3	17.0	1.1	15.0%	0.8	72.7%	13.5	183.1	20.4%	182.0	99.4%
threep2w9	12.0	0.1	10.0	0.6	16.7%	0.6	100.0%	8.0	36.2	20.0%	35.5	98.1%
threep3w9	18.0	0.3	13.0	1.9	27.8%	1.6	84.2%	11.0	191.2	15.4%	189.4	99.1%
Shapes0	68.0	33.9	60.0	178.6	11.8%	144.7	81.0%	60.0	239.1	0.0%	60.5	25.3%
Shapes1	62.0	424.4	58.0	1011.6	6.5%	587.2	58.0%	58.0	1132.7	0.0%	121.1	10.7%
Shapes2	31.0	7.2	28.0	130.1	9.7%	122.8	94.4%	27.6	310.7	1.5%	180.6	58.1%
Fu	40.0	5.6	40.0	9.6	0.0%	4.0	41.7%	32.0	252.3	20.0%	242.7	96.2%
Poly1a0	33.0	17.4	18.0	732.9	45.5%	715.5	97.6%	15.8	1048.8	12.2%	315.9	30.1%
Jakobs1	25.0	21.7	15.0	184.0	40.0%	162.4	88.3%	12.0	612.9	20.0%	428.8	70.0%
Jakobs2	36.0	34.1	30.0	645.2	16.7%	611.2	94.7%	26.0	1939.0	13.3%	1293.8	66.7%
Albano	12168.0	14.2	12168.0	342.1	0.0%	328.0	95.9%	10608.0	1614.1	12.8%	1272.0	78.8%
Mao	2315.0	21.0	2294.0	2321.8	0.9%	2300.9	99.1%	1927.2	2621.8	16.0%	300.0	11.4%
Marques	85.0	100.1	85.0	273.2	0.0%	173.1	63.4%	80.0	527.9	5.9%	254.7	48.2%
Trousers	439.0	229.9	296.0	1222.1	32.6%	992.2	81.2%	286.0	1403.7	3.4%	181.7	12.9%
Average					19.0%	171.8	85.3%			9.7%	200.8	61.7%

accurate solution is desired and using more computational time is not a problem, the complete solution method should be applied to the problem.

A variation of this matheuristic composed of only the construction and compaction phases was studied. The quality of the solutions obtained by this variation was always worse than that of the complete matheuristic.

5.2.3 Performance of the matheuristic performance compared with mixed-integer models

In this section, we analyzed the quality of the matheuristic solutions compared with the exact branch and cut method applied to two models from the literature. Table 5.3 presents the results for solving instances using the HS2 model from Alvarez-Valdes et al. (2013) by the dotted board model (DBM) from Toledo et al. (2013) with the piece-based mesh and by the proposed matheuristic. The results of HS2 were taken from Alvarez-Valdes et al. (2013). The specifications of their processor are better than the one used to solve the DBM and the proposed matheuristic[2]. Consequently, a comparison of the results is not unfair from the computational perspective. Moreover, each exact method was run for one hour.

In Table 5.3, the first column presents the instance names. The second and third columns present, respectively, the solution and time to prove the solution optimality of the Alvarez-Valdes et al. (2013) model. Similarly, columns four and six show the solution and time to prove the optimality of the dotted board model. Column five depicts the time that this model took to find the best solution of the search. Finally, in columns seven and eight, the solution obtained by the proposed matheuristic method and its computational time are shown.

The proposed matheuristic obtained better or equal solutions in 35 out of 40 instances when compared with the best solutions of the other two methods. In the table, the best solution values are highlighted. Compared only with the dotted board model, the proposed matheuristic yielded better results for 27 out of 40 instances. For the majority of the instances, the compaction phase makes a difference by removing some gaps from the grid dependence of the dotted board model, resulting in better-quality solutions.

The computational time of the matheuristic is less than that of the HS2 model only in the larger instances. In fact, this occurs because for small instances, the exact method can quickly find and prove the optimality of a solution while the matheuristic method needs to accomplish all three phases. Comparing the computational time of the dotted board model and the matheuristic, it can be observed that the exact method

[2]verified in www.cpubenchmark.net/

Table 5.3: Results from exact methods and the proposed matheuristic.

	HS2[1]		DBM			3PM	
Instance	**Solution**	**Time**	**Solution**	**Time (find)**	**Time**	**Solution**	**Time**
Blaze1	-	-	7.5	12.0	23.3	**7.4**	23.3
Blaze2	-	-	**14.0**	15.1	15.2	**14.0**	68.9
Blaze3	-	-	21.0	80.3	674.0	**20.5**	340.2
Blaze4	-	-	**27.0**	1068.0	1239.2	27.9	517.6
Blaze5	-	-	**34.0**	540.5	TL	**34.0**	395.2
Shapes_T2	-	-	16.0	0.5	1.7	**14.0**	8.1
Shapes_T4	-	-	**26.0**	58.4	89.1	**26.0**	201.5
Shapes_T5	-	-	**30.0**	340.6	365.0	31.0	106.0
Shapes_T7	-	-	**42.0**	2901.0	TL	**42.0**	176.3
Shapes_T9	-	-	49.0	3482.6	TL	**48.0**	292.3
RCO1	-	-	**8.0**	0.6	0.7	**8.0**	44.4
RCO2	-	-	**15.0**	1.2	1.3	**15.0**	254.5
RCO3	-	-	**22.0**	10.7	13.2	**22.0**	264.7
RCO4	-	-	**29.0**	16.7	394.0	**29.0**	83.1
RCO5	-	-	**36.0**	164.6	936.2	36.7	210.2
Albano	-	-	11088.0	592.4	592.4	**10608.0**	1614.1
Fu	-	-	35.0	53.1	53.1	**32.0**	252.3
Jakobs1	-	-	18.0	3285.3	TL	**12.0**	612.9
Jakobs2	-	-	30.0	596.2	TL	**26.0**	1939.0
Mao	-	-	2452.0	99.7	TL	**1927.2**	2621.8
Marques	-	-	88.0	1827.9	TL	**85.0**	527.9
Shapes0	-	-	64.0	3590.5	TL	**60.0**	239.1
Shapes1	-	-	80.0	98.9	TL	**58.0**	1132.7
Shapes2	-	-	**27.0**	900.4	TL	27.6	310.7
Trousers	-	-	495.0	218.4	TL	**286.0**	1403.7
Poly1a0	16.6	TL	17.0	3586.2	TL	**15.8**	1048.8
Shapes_AV4	**24.0**	0.0	**24.0**	1.7	1.7	**24.0**	2.2
Shapes_AV8	**26.0**	272.0	28.0	18.5	21.5	**26.0**	186.9
Fu5	**17.9**	0.1	20.5	2.8	3.4	**17.9**	1.0
Fu6	**23.0**	0.5	24.0	6.0	10.4	**23.0**	31.7
Fu7	**24.0**	1.0	28.0	0.1	0.2	**24.0**	5.0
Fu8	**24.0**	1.3	28.0	0.2	1.0	**24.0**	20.9
Fu9	**25.0**	70.0	28.0	0.4	0.4	**25.0**	52.3
Fu10	**28.7**	3064.0	30.0	0.8	0.9	**28.7**	265.9
Fu12	**31.2**	TL	40.0	1.0	1.0	32.0	186.7
threep1w7	**6.0**	0.8	6.5	0.3	0.3	**6.0**	2.5
threep2w7	**9.3**	3.9	11.0	0.7	0.8	**9.3**	12.4
threep3w7	**13.5**	3394.0	14.5	1.3	1.3	**13.5**	183.1
threep2w9	**8.0**	8.5	8.5	1.4	1.6	**8.0**	36.2
threep3w9	**11.0**	TL	13.0	0.2	0.2	**11.0**	191.2

TL: Time limit.
-: instances not addressed by Alvarez-Valdes et al. (2013).
[1] Results taken from Alvarez-Valdes et al. (2013).

spends less time on small instances. The reason for this is the same as that for the HS2 model. It is important to highlight that in several cases, the matheuristic obtained better solutions than the dotted board model as the model depends on the grid used. Its improvement in terms of the solution quality is more distinguishable with the large instances.

The advantage of the proposed matheuristic is that in comparison with the exact approaches, the time to achieve the objective is less biased by the instance size. Specifically, from the perspective of the computational time, the dimension of an instance does not exert much influence in terms of using the proposed solution method.

As the constructive and improvement phases are based on the dotted board model, instances with many different piece types and/or huge boards such as Albano, Mao and Jakobs2 can lead to longer solution times in these phases of the solution method. Moreover, in the compaction phase, the model used does not take advantage of pieces of the same type, making instances as Trousers, Shapes1 and Shapes0 more difficult to solve in this phase. On the other hand, the additional constraints imposed by the method in the models in each phase attempt to overcome these problems by attenuating the problems related to the matheuristic computational time. Additionally, the interactions between the approaches benefit the solution quality.

On the other hand, the additional constraints included in the models of each phase attempt to overcome the problem, reducing the computational times. Additionally, the interactions between the approaches aim to benefit the solution quality.

5.2.4 Performance of the matheuristic compared with those of other heuristics

In this section, the computational experiments comparing the proposed matheuristic and the heuristics of Leung et al. (2012) and Elkeran (2013) are presented.

The heuristics from the literature were run within different frameworks. The authors presented the best solution and the average solution found by their methods in several runs for each instance.

Table 5.4 presents the results obtained by 3PM and the results obtained by the two most recent heuristics from the literature. In the table, the first column displays the instance name. Columns two and three respectively present the solution found by 3PM and the computational time to obtain this solution. Columns four and five (six and seven) present analogous information for Leung et al. (2012) (Sato et al. (2012)) heuristic.

As 3PM is a deterministic procedure, it is run just once for each instance. In contrast, the heuristics proposed in Leung et al. (2012) and Elkeran (2013) are non-

Table 5.4: Comparison of the results of the exact methods with the 3-Phase Matheuristic (3PM).

	3PM		Leung et al. (2012)		Elkeran (2013)	
Instance	**Solution**	**Time**	**Solution**	**Time**	**Solution**	**Time**
Shapes0	60.0	239.1	59.7	10 x 1207.0	59.32	10 x 600.0
Shapes1	58.0	1132.7	53.7	10 x 1212.0	54.07	10 x 600.0
Shapes2	27.6	310.7	26.2	10 x 1205.0	26.21	10 x 600.0
Fu	32.0	252.3	31.7	10 x 600.0	31.46	10 x 600.0
Jakobs1	12.0	612.9	11.1	10 x 603.0	11.02	10 x 600.0
Jakobs2	26.0	1939.0	23.8	10 x 602.0	23.79	10 x 600.0
Albano	10608.0	1614.1	9969.5	10 x 1203.0	9959.24	10 x 600.0
Mao	1927.2	2621.8	1785.1	10 x 1204.0	1796.86	10 x 600.0
Marques	80.0	527.9	78.3	10 x 1204.0	77.37	10 x 600.0
Trousers	286.0	1403.7	246.7	10 x 1237.0	244.67	10 x 600.0

deterministic procedures that usually are run many times to ensure the quality of solution. The authors ran their heuristics 10 times that in the best case used 600 seconds for each time. Therefore, the proposed matheuristic is substantially faster and yields solutions in average six times faster than these heuristics.

On average, the solutions found by the matheuristic are 6.3% worse than the results obtained by Elkeran (2013) and Leung et al. (2012), which are the most recent heuristics in the literature.

5.3 Conclusions

A new matheuristic to solve the irregular strip packing problem combining mixed integer programming models from the literature is presented. The matheuristic is composed of three phases that use a model to solve each sub-problem. Combining different models, the proposed method takes advantage of the speed of the integer placement model and the solution quality of the linear placement model.

The outcomes of the proposed method show that it can produce solutions with better quality in shorter computational time in most cases when compared with the models. In addition, the performance of the matheuristic is not highly dependent on the instance dimensions, indicating that it is a good approach for tackling large instances.

Comparing 3PM with heuristics form the literature, 3PM found solutions in smaller computational times. Also, the quality of these solutions generally are near to the quality of the best solutions found in the literature.

Chapter 6

A new constraint programming approach to solve nesting problems[1]

The aim of the two-dimensional irregular cutting problem is to place convex or non-convex pieces on a board in order to optimize a given objective while ensuring that the pieces are inside of the board and do not overlap each other.

As presented in Chapter 2, Wäscher et al. (2007) classified these problems into: Placement Problem (*PP*), Identical Item Packing Problem (*IIPP*), Knapsack Problem (*KP*), Cutting Stock Problem (*CSP*), Bin Packing Problem (*BPP*) and Open Dimension Problem (*ODP*). Figure 6.1 illustrates the irregular two dimensional cutting and packing problem variants.

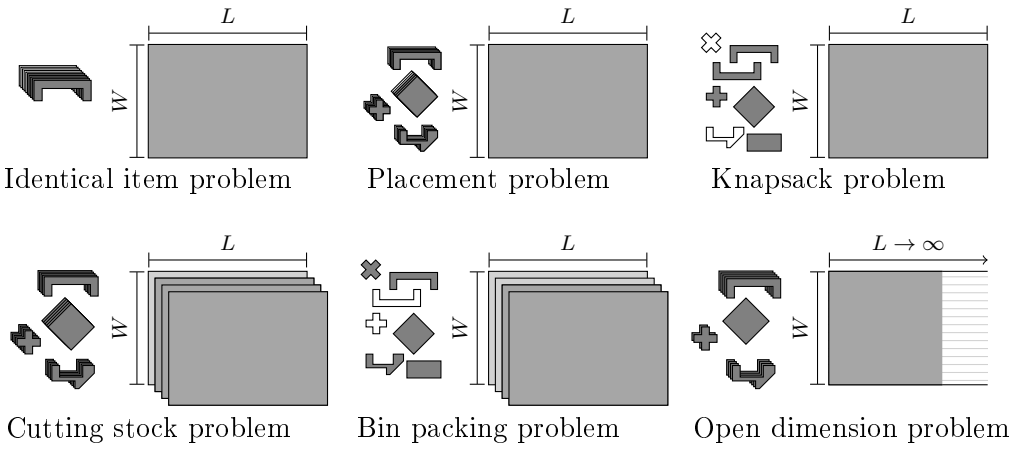


Figure 6.1: Variants of the irregular cutting and packing problems.

These problem variants are used to solve many real-world applications, and the variant that better fits each application depends on the industry characteristics.

[1]This chapter is strongly based on the paper "Optimality in irregular cutting and packing problems: new constraint programming models" which was submitted to a scientific journal

Constraint programming is a computational paradigm where constraints are at the core and the methods for manipulating and propagating constraints are tightly integrated with optimization strategies. Because the representation of the problem is done through logical constraints, there is no restriction or special characteristic required to build the set of constraints. Indeed, different from the mathematical programming approaches, a constraint does not need to be expressed by equations and inequalities. The constraints can be expressed by any logical or conditional relation over the variables. Current constraint programming systems offer a broad selection of constraints that can handle problems such as scheduling or shortest routes, and they have proven very effective in real-life problems where feasibility is an issue, due to its ability to deal with large number of heterogeneous constraints.

Constraint programming has been successfully applied to several combinatorial optimization problems. The first paper that solves the irregular strip packing problem by an exact method was proposed by Carravilla et al. (2003), where the authors use constraint logic programming to solve the problem. To solve the resource portfolio planning of make-to-stock products, Wang et al. (2007) proposed a constraint programming-based genetic algorithm. Clautiaux et al. (2008) proposed a constraint programming approach to solve the two-dimensional orthogonal packing problem outperforming the previous approaches. To solve the project scheduling problem under resource constraints, Trojet et al. (2011) proposed a constraint programming approach using the cumulative global constraint. The approach aims to give support to the decision maker by proposing a set of optimal solutions to the problem. Salas et al. (2014) proposed non-overlap constraints based on Minkowski sums for polygons described by non-linear constraints. The approach is, however, sensitive to the polygon shape, i. e., the more complex the shapes are, the more difficult it is to solve the problem.

The solution methods for general constraint programming models aim to reduce the domains of the variables trying to infer their values. When the values cannot be inferred, an enumeration scheme over a search tree is performed. The enumeration is composed of the branching process and the backtracking process. In the branching process, decisions about which value will be assigned to one variable domain are made, then the inference process (propagation) is performed again. A feasible solution is found when the value of all variables is defined. In order to prove the optimality, a backtracking process is performed, investigating all the possibilities of the search tree. Each node of a partial solution has the solution value up to that point, thus, if its quality is worse than the best solution found so far, this node can be pruned. Note that the value of a partial node in the search is given by the assignments already made to the variables and not by a relaxation. By not solving a relaxation in each node, the branching process can be performed faster than in a branch-and-bound method.

However, a strong lower bound could help pruning more nodes of the search tree and exploring less branches.

The strength of constraint programming comes from the possibility of modeling problems at a higher level, using the so-called global constraints. A global constraint is a specialized constraint for some problems, and allows the solver to use features of the problem which are not manageable if the model is expressed at the atomic level of basic constraints. Global constraints are at the core of the solution method for many classes of problems. The Global Constraint Catalog[2] has an extensive list of global constraints, which have been incorporated in various constraint programming solvers.

Global constraints have been proposed to solve several combinatorial optimization problems. Kovács and Beck (2008) proposed a global constraint for the total weighted completion time of activities for a single capacity resource. Saldanha and Morgado (2003) proposed a global constraint to solve a set the partitioning problem. The constraint has an efficient propagator and it is easy to be modified.

For cutting and packing problems, a global constraint to avoid the overlap between pieces was proposed by Ribeiro and Carravilla (2004). The proposed constraint, named "outside", is based on a model where the decision variables are the (X,Y)-coordinates of the positioning points of the pieces. The main limitation of the approach stems from the two-dimensionality of the problem: having X and Y coordinates represented as different variables limits the effectiveness of constraint propagation. Besides that, the model does not take advantage of pieces with the same shape. Therefore, this approach cannot be easily adapted to solve the variants of the irregular cutting problems where the number of pieces to be cut is not limited.

Here, constraint programming methods to solve irregular cutting and packing problems are investigated. Two approaches to represent the problem variables are developed. In the first one, the domain of the variables is binary and, in the second one, the domain is integer. As the core of the irregular cutting and packing problems is the no overlap constraint, a specialized global constraint to avoid the pieces overlap is proposed. This global constraint can efficiently propagate and can be used in many variants of irregular cutting and packing problems. Furthermore, additional constraints combined with no overlap constraints, that represent several variants of irregular cutting and packing problems, are proposed. Several of these variants were never solved in the literature by an exact method. Computational experiments show that the proposed models can solve the addressed problems. Using the global constraint, the computational time to prove optimality (when it is reached in the given time limit) and the memory used are reduced at least by an order of magnitude. Although the computational time to prove solution optimality is usually high in all the proposed approaches,

[2]Global Constraint Catalog: http://sofdem.github.io/gccat/

all the formulations can quickly find good quality solutions.

As in constraint logic programming the domain of the variables needs to be finite, the dot structure presented in Chapter 4 is used to assist the program (model) generation. The dot structure enables the program to easily retrieve the information on the innerfit and nofit polygon, helping to create the constraints and infer the initial domain of the variables.

The remainder of the chapter is organized as follows: Section 6.1 presents some definitions that are used along the chapter. Section 6.2 shows binary domain variables to represent the board dots and the no-overlap constraint using this variable definition. A representation of the board dots through integer domain variables and the non-overlap constraints made for these variables are proposed in Section 6.3. In Section 6.4, a specialized global constraint to avoid the overlap between pieces is proposed. This global constraint uses integer domain variables and is tailored for Irregular Cutting and Packing Problems. A set of constraints that can represent any variant of cutting and packing problems defined in Wäscher et al. (2007) typology is described in Section 6.5. In Section 6.6, the computational results are analyzed, showing the advantage of each model and also the versatility of the proposed constraint programming approach to solve all the cutting and packing problem variants. Section 6.7 presents the results obtained for larger instances and a comparison with the dotted-board model (Toledo et al., 2013). Finally, in Section 6.8 the conclusions and highlights of the approach are presented.

6.1 General concepts

In all the variants of the irregular cutting problems, a finite number of pieces types T with a fixed number of allowed rotations R must to be placed in a board. The pieces must be positioned inside the board and cannot overlap. The objective to be considered may be to maximize the value extracted from the board or to minimize the used board(s). The pieces are represented by an ordered set of vertices and by a point, chosen to be the piece reference point. In the approach described in this paper, the board is discretized by a regular mesh of dots D, the allowed placement positions of the pieces on the board. The mesh is regular, i.e. the vertical and horizontal distance between the dots are a multiple of a parameter Δ which determines the refinement of the mesh.

To ensure that the pieces are entirely inside the board, the innerfit polygon (IFP) is used. The IFP of piece t at rotation r (IFP_{tr}) defines the region of the board where the positioning point of a piece can be placed, such that the piece is entirely inside the board. As in this chapter the board used to perform the cuts is considered to

be rectangular, the innerfit polygon can be easily defined based on the vertical and horizontal distances between the reference point and the sides of the piece bounding box. More details about the pieces dimensions and how to define the innerfit polygon are presented in Section 2.1.

In our approach, the nofit polygon (*NFP*) is used to enforce the condition that pieces do not overlap. The *NFP* of piece t at rotation r and piece t' at rotation r' ($NFP_{tr}^{t'r'}$) summarizes the geometric relation between these pieces reducing the overlap evaluation to the verification if a point is inside, over or outside a polygon. Details of the nofit polygon can be found in Section 2.1.

As in the models presented in this chapter the board is represented by a grid of dots, it suffices to know the dots of the grid inside the nofit polygon to avoid the overlaps. Therefore, the dots structure proposed in Chapter 4 is used to represent the problem geometry.

6.2 CP formulation based on the Dotted Board Model

In this section, the formulation is based on the dotted board model (Toledo et al., 2013), i.e. the there is a binary variable defined for each dot of the board for each piece type and respective rotation.

Based on this definition of the variables, a set of constraints that avoid the overlap among pieces is proposed. The constraints that prevent the pieces to overlap are the most challenging constraints to be satisfied in irregular cutting and packing problems.

6.2.1 Binary representation

In the binary representation, for each dot $d \in D$, piece type $t = 1, ..., T$ and rotation $r = 1, ..., R_t$, a variable which tells if the piece type t at rotation r is placed or not on the dot d is defined. Specifically, the variable δ_{tr}^{d} is defined as 1 if piece t at rotation r is placed on the dot d; and 0 otherwise. If the dot $d \in D$ does not belong to the IFP_{tr}, δ_{tr}^{d} is equal to zero.

For the OR community this representation can be more intuitive to represent the problem constraints since each variable carries only the information of a specific piece, rotation and dot. However, this representation needs a huge number of variables with small (binary) domains. For example, a small problem where ten piece types, with four possible rotations each, are candidates to be placed in a board with one hundred dots needs about four thousand binary variables ($10 \times 4 \times 100 = 4000$) to represent the placement positions for the pieces.

6.2.2 Non-overlap constraints based on the binary representation

Consider that piece type t at rotation r is placed on dot $d \in D$. In order to enforce non-overlap between this piece and piece type t' at rotation r', all variables $\delta_{t'r'd'}$, for dots $d' \in NFP^{t'r'}_{trd}$, are set to 0 as follows:

$$\texttt{If } (\delta^{d}_{tr} = 1) \texttt{ Then } (\delta^{d'}_{t'r'} = 0),$$
$$t = 1, ..., T, t' = t, ..., T, r = 1, ..., R_t, r' = 1, ..., R_{t'}, d \in D, d' \in \Phi^{t'r'}_{trd}. \quad (6.1)$$

These constraints propagate when some δ_{trd} is set to 1 in the search, which happens when some piece is positioned. As one of these constraints propagates, one binary variable becomes ground, i.e., its domain has only the 0 value. To enforce all possible non-overlap constraints, for each dot D it is necessary to create a constraint associating all T piece types with their R_t possible rotations that can be placed on this dot with the other T piece types at R_t possible rotations that can be placed on the $NFP^{t'r'}_{trd}$ set. To avoid the overlap among the pieces $\sum_{d \in D} \sum_{t=1}^{T} \sum_{t'=t}^{T} \sum_{r=1}^{R_t} \sum_{r'=1}^{R_{t'}} |NFP^{t'r'}_{trd}|$ constraints are needed, where $|.|$ denotes the cardinality of the set.

6.3 CP formulation based on integer domains

Using binary variables to represent the combination of dots, piece types and rotations leads to a large number of variables. As an alternative, we consider integer domain variables to represent the status of the board dots. With this representation the number of variables needed to formulate the constraints is reduced while their domains have more possible values.

6.3.1 Integer representation

irregular cutting and packing problems aim to place pieces, which may be rotated, on one or more boards. In the integer representation, each piece type at a particular rotation is mapped into a single number. We use the piece types introduced before, $t = 1, ..., T$, and their corresponding rotations $r = 1, ..., R_t$. The integer number associated with piece type t at rotation r, n_{tr}, is given by:

$$n_{tr} = (t - 1) \times R^{\max} + r,$$

where $R^{\max} = \max_t \{R_t | t = 1, ..., T\}$. Note that this mapping has a simple inverse

transformation. Given n_{tr}, we obtain the piece type t and its rotation r as

$$t = \lfloor \frac{n_{tr}}{R^{\max}} \rfloor$$

and

$$r = n_{tr} - \lfloor \frac{n_{tr}}{R^{\max}} \rfloor R^{\max}.$$

Let γ_d, $d \in D$, be the variable representing dot d of the board. The domain of γ_d is composed by the possible values for (n_{tr}) of piece types t and rotations r, such that t with rotations r can be placed on dot d. Zero is also a possible domain value, corresponding to the situation where no piece reference point is placed on the dot:

$$\gamma_d = \{n_{tr} | t = 1, ..., T, r = 1, ..., R_t, d \in \mathit{IFP}_{tr}\} \cup \{0\}$$

Consider an example with three piece types, the first with two allowed rotations and the second and third types with one allowed rotation each. The resulting mappings are:

$$n_{11} = (t-1) \times R^{\max} + r = 0 \times 2 + 1 = 1$$

$$n_{12} = (t-1) \times R^{\max} + r = 0 \times 2 + 2 = 2$$

$$n_{21} = (t-1) \times R^{\max} + r = 1 \times 2 + 1 = 3$$

$$n_{31} = (t-1) \times R^{\max} + r = 2 \times 2 + 1 = 5$$

The mapping results in a unique identification for each piece at each rotation. Note that the mapping is not continuous, which is not an obstacle in the constraint programming formulation.

Clearly, using this approach the number of variables used to model the problem is smaller. There is still a problem with this representation: in a solution, each variable must be assigned a single value, and therefore this representation prevents more than one piece to be placed on the same dot. This may however be required by some feasible solutions. This problem is overcome with a careful selection of the piece reference point. Figure 6.2a illustrates two pieces whose chosen reference points lead to a feasible positioning pattern which would not be obtained by our CP engine. To avoid this situation, the reference points for the pieces must be chosen in a way that if two pieces are placed on the same dot, then they overlap. This is the case with the new piece reference points illustrated in Figure 6.2b.

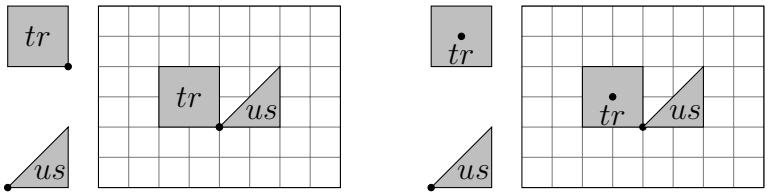


Figure 6.2: (a) represents a placement position that is feasible for the binary representation and is infeasible for the integer representation. This can be avoided by changing, for example, the reference point of piece tr as illustrated in (b).

Non-overlap constraints for the integer representation

Consider that piece t at rotation r, mapped as n_{tr}, is placed on dot $d \in D$. In order to avoid the overlap, the value $n_{t'r'}$ must be removed from the domain of the dots $d' \in NFP^{t'r'}_{trd}$, i.e., pieces of type t' at rotation r' must not be positioned on dot d':

$$\begin{aligned} &\texttt{If } (\gamma_d = n_{tr}) \texttt{ Then } (n_{t'r'} \notin \gamma_{d'}), \\ &t = 1, ..., T, t' = t, ..., T, r = 1, ..., R_t, r' = 1, ..., R_{t'}, d \in D, d' \in \Phi^d_{t[r],t'[r']} \end{aligned} \tag{6.2}$$

Constraints (6.2) propagates when a variable γ_d becomes ground positive integer value. When the domain of a variable is reduced, only some values are eliminated. Although the number of constraints to enforce the non-overlap is exactly the same in the integer and binary representation, this approach uses less variables.

6.4 `NoOverlap`: a new global constraint

In this section, a new global constraint `NoOverlap` is proposed. Despite the amount of information that the integer representation has, the non overlap constraints derived from built-in generic constraints such as if-then can not take advantage of it. The development of a new global constraint tailored to the problem can reduce the number of non-overlap constraints and make a more efficient propagation.

In order to define these new constraints, consider the set:

$$\Psi_{trd} = \{d' | d' \in \bigcap_{t' \in T, r' \in R_{t'}} \Phi^d_{t[r],t'[r']}\}, \tag{6.3}$$

which represents the set of dots where no other piece can be positioned if piece type t at rotation r is placed over dot d. Figure 6.3 illustrates Ψ_{trd} for an example where

three pieces (Figure 6.3a) must be cut. Figure 6.3b presents the nofit polygons of piece type t at rotation r with the other two pieces. When piece t at rotation r is placed on the dot d, $\Phi^d_{t[r],u[s]}$ and $\Phi^d_{t[r],u'[s']}$ intersect as shown in Figure 6.3c. The dots strictly inside the intersection of the shaded regions define the set Ψ_{trd}.

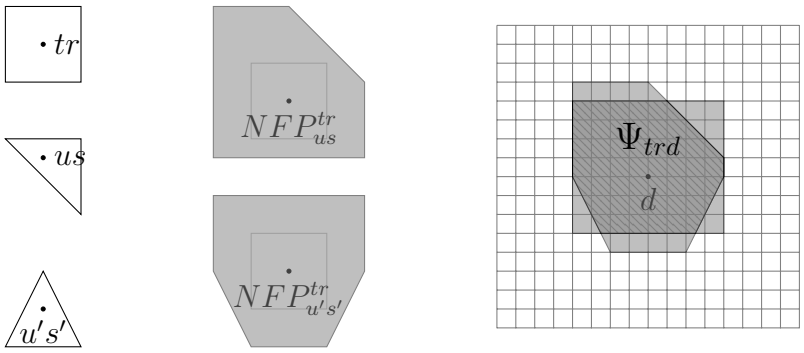


Figure 6.3: Example of set Ψ_{trd}.

Using this set, the `NoOverlap` constraint can be defined reducing the number of operations needed to infer the domains of the variables domain through non-overlap constraints. The specialized constraint is presented in (6.4).

$$\texttt{NoOverlap}(\gamma_d \mid \Psi_{trd}, \Phi^d_{t[r],t'[r']}/\Psi_{trd}). \tag{6.4}$$

Given sets Ψ_{trd} and $\Phi^d_{t[r],t'[r']}/\Psi_{trd}$, the implementation of the constraint propagator is simple. This constraint must be executed each time the variable γ_d assumes a specific value. When this happens, the domain of all variables in the set Ψ_{trd} set can be reduced to zero. The domain of each $\gamma_{d'}$, $d' \in \{\Phi^d_{t[r],t'[r']}/\Psi_{trd}\}$ must be reduced by the value $n_{t'r'}$, for all $t' = 1, ..., T$ and $r' = 1, ..., R_{t'}$. The constraint propagator is represented in Algorithm 10.

This constraint is propagated only when γ_d is bounded by a value greater than zero. Only one constraint is assigned to each dot and an efficient propagation method is used to reduce the domains of the variables.

6.5 CP models for all the variants of irregular cutting and packing problems

The non-overlap constraints are independent of the irregular cutting and packing problem variant, however each variant needs a set of additional constraints to be represented.

Algorithm 10: `NoOverlap` propagator.

- **Input:** The variables γ_d and sets Ψ_{trd} and $\Phi^d_{t[r],t'[r']}/\Psi_{trd}$, for all $d \in D$, $t = 1, ..., T$, $r = 1, ..., R_t$;
- **begin**

  Calculate $t = \lfloor \frac{n_{tr}}{R^{\max}} \rfloor$ and $r = n_{tr} - \lfloor \frac{n_{tr}}{R^{\max}} \rfloor R^{\max}$;

  For all $d' \in \Psi_{trd}$ do

  Do $\gamma'_d = 0$;

  For all $\left(t' = 1, ..., T, r' = 1, ..., R_{t'}, d' \in \Phi^d_{t[r],t'[r']}/\Psi_{trd}\right)$ do

  Remove $n_{t'r'}$ from $\gamma_{d'}$ domain;

  Return;
- **end.**

In the following, the constraint programming models for the problems shown in chapter 2 are presented. These models are composed by a set of built-in constraints based on the binary or integer variables representation and the non overlap constraints presented in sections 6.2, 6.3 and 6.4.

6.5.1 Irregular Placement Problem (`IPP`) and Irregular Identical Item Placement Problem (`IIIPP`)

The models for the `IPP` and the `IIIPP` are similar because the difference between these two problem variants is on the number of piece types to be cut and this is a characteristic of the problem instance.

In these problems, the board dimensions are defined by the instance and therefore the initial domains can be determined in the preprocessing phase using the *IFP* (Section 6.1). The models for the `IPP` (or `IIIPP`) are completed by adding the non-overlap constraints and an adequate objective function.

In the binary representation the non-overlap constraints are represented in (6.1).

The `IPP` and the `IIIPP` are both output maximization problems, therefore the number of boards available to be cut is fixed and the objective function must maximize the value of the pieces cut (extracted) from these boards. Specifically, considering that each piece of type $t = 1, ..., T$ has a value v_t, the objective function that maximizes the profit can be expressed by (6.5).

$$\text{maximize} \sum_{d \in D} \sum_{t=1}^{T} \sum_{r=1}^{R_t} v_t \delta_{tr}^{d}. \tag{6.5}$$

If v_t is defined as the area of the pieces, this objective minimizes the waste. The objective function (6.5) and Constraints (6.1) compose the binary formulation of the `IPP` and `IIIPP`.

In the integer representation the non-overlap constraints may be the one in (6.2) or the proposed `NoOverlap` constraint (6.4).

As the piece types are mapped, the objective function is slightly different. For each piece of type $t = 1, ..., T$ at rotation $r = 1, ..., R_t$ the number of variables that were bounded in n_{tr} must be counted and then this number must be multiplied by the piece value. The sum of these values is the objective function value. This expression can be formulated using the built-in constraint `count` that is usually available in constraint programming solvers. Expression (6.6) is the objective function for output maximization problems for variables with integer domains.

$$\text{maximize} \sum_{t=1}^{T} \underset{d \in D, r=1,...,R_t}{\texttt{count}} (\gamma_d = n_{tr}) v_t. \tag{6.6}$$

The objective function (6.6) together with Constraints (6.2) define the integer formulation of `IPP` and `IIIPP`.

The integer formulation with a specialized non-overlap constraint is obtained if the objective function (6.6) is combined with the new global constraint `NoOverlap` (6.4).

6.5.2 Constrained Irregular Placement Problem (`IPPc`) and Irregular Knapsack Problem (`IKP`)

The models for the `IPPc` and the `IKP` can be built on the model for the `IPP` simply by adding a constraint that limits the number of pieces of each type to cut. In these two variants, as in the `IPP`, the board dimensions are defined by the instance and therefore the initial domains can be determined in the preprocessing phase using the *IFP* (Section 6.1).

The `IPPc` and the `IKP` have similar models, because the difference between these two problem variants is on the demand for each piece type, again a characteristic of

the problem instance.

In the binary representation Constraints (6.7) limit the number of pieces to cut. These constraints count the number of times that a piece type is present in the solution and guarantee that this number is less than or equal to the limit q_t for piece type t.

$$\underset{d\in D, r=1,...,R_t}{\texttt{count}} \left(\delta_{tr}^{d} = 1\right) \leq q_t, \quad t = 1, ..., T, \tag{6.7}$$

The model for `IPPc` and `IKP` for decision variables with binary domains is obtained by adding the objective function (6.5), the Constraints (6.1) and the Constraints (6.7).

In the integer representation Constraints (6.8) ensure that the demand for the pieces is not exceeded. For all the pieces types t, the constraints count the number of times that the variables assume the value n_{tr}, for all $r = 1, ..., R_t$, and require that this number is less than or equal to the limit q_t for piece type t.

$$\underset{d\in D, r=1,...,R_t}{\texttt{count}} \left(\gamma_d = n_{tr}\right) \leq q_t, \quad t = 1, ..., T. \tag{6.8}$$

The model for `IPPc` and `IKP` for decision variables with integer domains is obtained by adding the objective function (6.6), the Constraints (6.2) and the Constraints (6.8).

The integer formulation with the new global constraint `NoOverlap` is obtained by combining the objective function (6.6), the Constraints (6.8) and the global constraint `NoOverlap` (6.4).

6.5.3 Irregular One Open Dimension Problem (`I1ODP`)

The `I1ODP` is an input minimization problem and therefore the length of the board is not known a-priory. However, defining an upper bound to the solution length ($\overline{L}$) is enough for the board to be considered finite and rectangular. Knowing $\overline{L}$ the initial domains of the decision variables can be determined in the preprocessing phase using the *IFP* (Section 6.1).

In all input minimization problems the demand q_t for each piece of type $t = 1, ..., T$ is known and needs to be met. The objective is to minimize the board length used to cut all the demanded pieces.

In the binary representation Constraints (6.9) ensure that the demand for the pieces is met. These constraints count the number of times that piece type t is in the

solution and guarantee that this number is equal to the demand q_t.

$$\underset{d\in D, r=1,...,R_t}{\texttt{count}} \left(\delta_{tr}^{d} = 1\right) = q_t, \quad t = 1, ..., T, \tag{6.9}$$

To express the objective function, the position of each piece must be analyzed in order to determine the objective function value. As the objective is to minimize the used board length, the piece that occupies the rightmost position on the board determines the value for the objective function. In the binary formulation this objective can be represented by (6.10).

$$\text{minimize} \quad \max_{\substack{d \in D \\ t = 1, ..., T \\ r = 1, ..., R_t}} (d_x + l_{tr}^{right})\delta_{tr}^{d}. \tag{6.10}$$

Together, the objective function (6.10) and Constraints (6.9) and (6.1) compose the model of `I1ODP` using binary variables.

In the integer representation the demand is accounted by Constraints (6.11) which for each t=1,...,T count the number of times that variable γ_d assumes the value n_{tr}, for all $r = 1, ..., R_t$, and require this number to be equal to demand q_t of piece type t.

$$\underset{d\in D, r=1,...,R_t}{\texttt{count}} (\gamma_d = n_{tr}) = q_t, \quad t = 1, ..., T. \tag{6.11}$$

In order to measure the objective of `I1ODP` additional constraints must be used. Consider the variable $\mathcal{L}$ that measures the solution length. Constraints (6.12) ensure that $\mathcal{L}$ will be at least as long as the solution length.

$$\texttt{If} \quad (\gamma_d = n_{tr}) \quad \texttt{Then} \quad \left(\mathcal{L} \geq d_x + l_{tr}^{right}\right), \quad d \in D, t = 1, ..., T, r = 1, ..., R_t. \tag{6.12}$$

Using Constraints (6.12), the objective function for the irregular *1ODP* can be expressed by (6.13).

$$\text{minimize } \mathcal{L}. \tag{6.13}$$

The model for the I1ODP with integer-domain variables can be represented by objective function (6.13) supported by Constraints (6.2), (6.11) and (6.12).

The integer formulation with the new global constraint NoOverlap (6.4) comprises the objective function (6.13), the global constraint (6.4) and the Constraints (6.11) and (6.12).

6.5.4 Irregular Cutting Stock Problem (ICSP) and Irregular Bin Packing Problem (IBPP)

The difference of ICSP and IBPP is on the demand for each piece type, i.e. the formulations may be the same and only the instances differ.

This problem aims to cut all the demanded pieces from N boards, minimizing the number of used boards. All the pieces must be completely inside one of the boards and it is considered that all the boards have the same width W and length L_{board}.

To handle this problem, the number of boards required to cut all the pieces must be estimated. In order to represent the problem using the same definition for the variables, consider an extended board of width W and length $L = N \times L_{board}$. $N - 1$ vertical cuts are made in the extended board dividing it in the N original boards. In order to avoid to place the pieces over the cuts, some additional constraints on the IFP's must be considered and will be presented for each different variable domain. Figure 6.4 illustrates an example of a board and the new IFP's for a piece of type t at rotation r (IFP_{tr}).

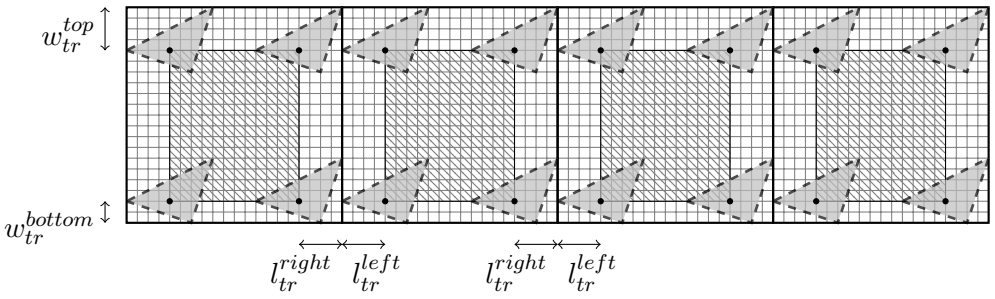


Figure 6.4: Board used on irregular cutting stock problems and irregular bin packing problems.

According to this board definition, the objective for ICSP and IBPP is the same as the one used to represent I1ODP. This objective ensures that the number of boards used will be minimized and that the used length of the last board will be reduced and therefore the material waste of this last board is also minimized.

In the binary representation consider, to define the IFP's, that each dot $d \in D$ has coordinates (d_x, d_y). The region of IFP_{tr} can be inferred by fixing the domain of

δ^d_{tr} to zero to avoid piece type t in rotation r to be over the cuts, i.e. $d_x - l^{left}_{tr} < k \times \frac{L}{N}$ or $d_x + l^{right}_{tr} > (k+1) \times \frac{L}{N}$, for $k = 1, ..., N-1$

Note that as the dots, the dimensions of the pieces and the dimensions of the board are known, the domains of the variables can be reduced in a pre-processing phase.

Considering the board definition presented in this section and the corresponding domain reductions, the objective function (6.10) together with Constraints (6.9) and (6.1) models the `ICSP` and the `IBPP` with binary variables.

In the integer representation similarly to the binary case, the IFP_{tr} can be used to reduce the domains of γ_d by n_{tr} if $d_x - l^{left}_{tr} < k \times \frac{L}{N}$ or $d_x + l^{right}_{tr} > (k+1) \times \frac{L}{N}$ for $k = 1, ..., N-1$. These domain reductions can be done in the pre-processing phase.

Considering the board definition presented in this section and the corresponding domain reductions, the objective function (6.13) together with Constraints (6.11) and (6.2) models the `ICSP` and the `IBPP` with integer variables.

By replacing Constraints (6.2) by the global constraint `NoOverlap` (6.4) the integer formulation with a custom constraint is obtained.

6.5.5 Irregular Two Open Dimension Problem (`I2ODP`)

Developing exact methods for the `I2ODP` demands some more effort compared with the other cutting and packing problem variants. In this problem variant two board dimensions must be estimated, leading to a large board and, consequently, to a formulation with a large number of variables. Nevertheless, depending on the objective function, the domains of some variables may be reduced in the pre-processing phase.

If the objective is to minimize the area of a rectangle that contains all the pieces and if it is known that all the pieces fit in a rectangle of area A, then, for each piece type t and rotation r, the domains of the variables assigned to the dots that do not respect inequality (6.14) must be reduced.

$$(x + l^{right}_{tr}) \times (y + w^{top}_{tr}) \leq A. \tag{6.14}$$

Note that the use of a dot that does not respect this inequality leads to a board with an area larger than A. It is also clear that the length (width) of the board will be at least as long (high) as the longest (highest) piece type t at rotation r. Figure 6.5 shows a rectangular board and the region defined by inequality (6.14) is represented in light gray. The rectangle within the curve is an example of a region that a solution can use. It is important to highlight that these irregular boards can be defined at a pre-processing phase reducing the number of constraints in the model.

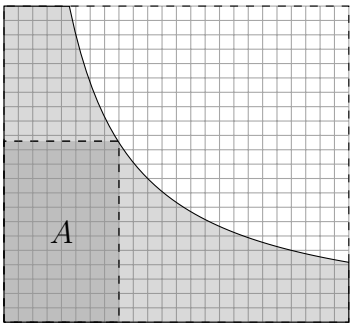


Figure 6.5: Board used on open dimension problem with two open dimensions.

When two dimensions are open and the objective is to minimize the used area, the objective is non-linear, but it is important to highlight that it is possible to solve this nonlinear problem because constraint programming does not require neither the constraints nor the objective function to be linear.

In the binary representation the objective function (6.15) minimizes this rectangular area.

$$\text{minimize} \quad \max_{\substack{d \in D \\ t=1,...,T \\ r=1,...,R_t}} (d_x + l_{tr}^{right})\delta_{tr}^{d} \times \max_{\substack{d \in D \\ t=1,...,T \\ r=1,...,R_t}} (d_y + w_{tr}^{bottom})\delta_{tr}^{d}. \tag{6.15}$$

To represent the `I2ODP`, Constraints (6.1) and (6.9) are combined with objective function (6.15).

In the integer representation additional constraints must be used to define the objective function. Consider the variable $\mathcal{L}$ that measures the length of the solution and $\mathcal{W}$ that measures the width of the solution. Constraints (6.12) and (6.16) ensure that $\mathcal{L}$ and $\mathcal{W}$ will properly represent the board length and the board width.

$$\texttt{If} \quad (\gamma_d = n_{tr}) \quad \texttt{Then} \quad \left(\mathcal{W} \geq d_y + w_{tr}^{bottom}\right), \quad d \in D, t = 1, ..., T, r = 1, ..., R_t. \tag{6.16}$$

To minimize the area of the rectangle that contains all the pieces, objective function (6.17) is used.

$$\text{minimize } \mathcal{L} \times \mathcal{W}. \tag{6.17}$$

The formulation of the `I2ODP` with integer variables is obtained using objective

function 6.17 subject to Constraints (6.2), (6.11), (6.12) and (6.16).

By changing Constraints (6.2) by Constraints (6.4) the overlap between pieces is solved using the new global constraint `NoOverlap`.

It is important to highlight that open dimension problems can have many different objectives. The minimization of the board length and the minimization of the area of the rectangle were chosen because these objectives were already studied in the literature.

6.6 Computational experiments with the different CP models

In this section, the computational experiments with all the proposed constraint programming models are presented. The experiments were run on a computer with an Intel Xeon Processor E5-2450 with 64GB of memory using Scientific Linux 6 operational system. The maximum solution time allowed for each problem with any method was one hour. To implement and solve the problem formulations, the constraint programming solver provided in IBM ILOG CPLEX 12.6 was used.

To identify the problem variant and the constraint programming model used to solve it, the codes are named by the abbreviation of problem variant name and a specification of the model. The constraint programming models are represented by `Bin`, `Int` or `IGC`. For example, the irregular placement problem (`IPP`) solved by the constraint programming model with binary variables (`Bin`) is named `IPP-Bin`.

6.6.1 Defining instances

The instances used to run the computational experiments are based on the ones used in the literature to solve the irregular *ODP* with one open dimension. As this is the most studied variant of the problem, many instances were proposed to evaluate solution methods. To evaluate and compare the performance of the proposed models, a subset of instances was taken from ESICUP[3]. The chosen instances are Blaz1, Blaz2, Shapes0, Shapes1, Fu and Dagli. The mesh used for instances Blaz1, Blaz2, Shapes0 and Shapes1 has a refinement $\Delta = 1$ and the mesh used to solve Fu and Dagli has a refinement $\Delta = 2$. The choice of different values of Δ for the instances guarantees that they can be solved by the three proposed models. In a second phase, the best approach is used to solve larger instances (e.g., Jakobs1, Jakobs2, Shirts) besides Blaz1, Blaz2, Shapes0, Shapes1, Fu and Dagli with a more refined value for Δ, adapted to

[3]EURO Special Interest Group on Cutting and Packing: http://paginas.fe.up.pt/ esicup/

each problem variant studied. In this phase, the mesh refinement Δ of each instance is specified in the tables.

The instances used for each problem variant were derived from these base-instances. For the irregular *ODP* with one open dimension, an upper bound on the length of the board $\overline{L}$ must be defined. The value of $\overline{L}$ must be carefully set, because on the one hand it needs to be large enough to contain feasible solutions, but on the other hand the number of problem variables increases with this dimension. $\overline{L}$ is defined as the first solution found by the model with the global constraint run with a board size big enough to find a feasible solution in less than one minute. This model was chosen to be executed because, as showed in Section 6.6.4, it uses less memory than the other two approaches. Therefore, for the instances Blaz1, Blaz2, Shapes0, Shapes1, Fu and Dagli the board length $\overline{L}$ was defined respectively as 32, 24, 80, 77, 38 and 85.

For the problem variants where the board has fixed dimensions (*KP*, *PP*, *IIPP*, *BPP* and *CSP*), following Song and Bennell (2014) the length of the board is defined as equal to its width. The width of the board is always known as the original instances come from the strip packing problem. In the input minimization problems, where the demand is constrained, the demand of the original instance is maintained. In the problem variants where the pieces must have a value, the value of each piece is set as its area.

In *IIPP* only one piece type must to be placed on the board. To generate instances for this problem, the pieces from instances Blaz2 and Shapes1 were used to create eight new instances. Each instance contains only one of the pieces from the original instance as depicted in Figures 6.6 and 6.7.

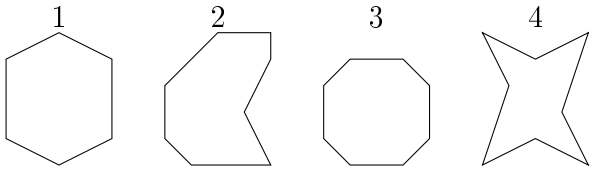


Figure 6.6: Enumerating pieces in the Blaz2 instance.

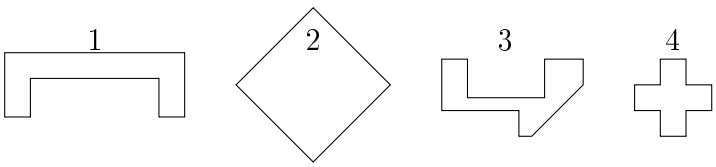


Figure 6.7: Enumerating pieces in the Shapes1 instance.

When, in the original instance, a piece has an allowed rotation, this characteristic is maintained in the new instance.

6.6.2 Output maximization problems

In the output maximization problems the board is well defined and the problem is to decide which pieces will be cut and the placement positions of these pieces on the board. The results comparing the three proposed models for the variants of this class of problems are presented in Sections 6.6.2, 6.6.2, 6.6.2 and 6.6.2.

Irregular *IIPP* (IIIPP)

The irregular identical item placement problem (IIIPP) consists of cutting a single irregular piece type from a board with fixed dimensions. Table 6.1 presents the results for the IIIPP obtained by the three proposed models. In the table, the instance name is in the first column. Columns three and four present the value of the solution and the time for the model with variables with binary domains. Columns four and five (six and seven) have the same content of columns two and three for the model with variables with integer domains (for the model with the proposed global constraint NoOverlap).

Table 6.1: Results reached for irregular IIIPP

	IIIPP-Bin		IIIPP-Int		IIIPP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz2-1	144.0	1040.2*	144.0	779.8*	**144.0**	**0.0***
Blaz2-2	155.0	1.4	155.0	1.3	**155.0**	**0.4**
Blaz2-3	**168.0**	**0.2**	168.0	0.3	168.0	0.3
Blaz2-4	121.0	878.5*	121.0	580.3*	**121.0**	**215.5***
Shapes1-1	880.0	119.4	880.0	473.3	**880.0**	**1.8**
Shapes1-2	1080.0	174.5	1080.0	30.9	**1080.0**	**0.8**
Shapes1-3	**952.0**	2982.4	924.0	1128.9	924.0	3067.4
Shapes1-4	1160.0	115.0	1180.0	301.1	**1220.0**	4.4

*: optimal solution.

As in this problem variant only one piece type is placed the binary and integer representations are very similar. Considering the solution values, all the approaches reach the same values except for the instances Shapes1-3 where the best solution was found by by the model with binary decision variables and Shapes1-4 where the best solution was found by the integer program with the proposed global constraint.

The integer program with the proposed global constraint could find better or equal solutions faster than the other proposed formulations except for the instances Shapes1-3 and Blaz2-3. This fact happens because even for one piece type, the global constraint can significantly reduce the number of constraints necessary to avoid pieces overlap.

Irregular *PP* (IPP)

The irregular placement problem (IPP) aims to place a number of pieces on a board in order to maximize the value extracted by cutting the pieces. Table 6.2 presents the results obtained using the proposed models to solve the irregular placement problem and the columns have the same type of information as in Table 6.1.

Table 6.2: Results reached for IPP without demand constraints.

	IPP-Bin		IPP-Int		IPP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	**214.0**	1449.6	209.0	773.2	210.0	1596.5
Blaz2	178.0	333.6	178.0	293.3	**178.0**	**102.9**
Shapes0	1104.0	1934.0	1108.0	2946.1	**1128.0**	3156.4
Shapes1	1096.0	400.9	1104.0	1888.6	**1104.0**	**467.9**
Fu	om	om	om	om	**1402.0**	3053.0
Dagli	om	om	om	om	**2493.5**	1459.0

om: out of memory.

Among the three models, none was able to prove the solution optimality in the given time limit for any instance. It is possible to verify that, except for the instance Blaz1, IPP-IGC found solutions with the same or better quality than the other two formulations.

As to the computational times, these are difficult to compare because the solutions obtained with the various models are very different. However, for the instance Blaz2 where all the methods reached the same solution quality, IPP-IGC was the fastest model. Also, for Shapes1, IPP-Int and IPP-IGC found solutions with the same quality and IPP-IGC was more than three times faster than IPP-Int.

Irregular *PP* with demand constraints (IPPc)

The irregular placement problem with demand constraints (IPPc) has a smaller solution space compared with the irregular *PP* without demand constraints. It happens because with a reduced number of pieces to allocate, in some cases the variable domains can be quickly reduced. In Table 6.3, the computational experiments for IPPc and for each model are presented. The columns of the table have the same type of content as in Table 6.1.

IPPc-IGC was the only approach able to find a feasible solution for the instances Fu and Dagli. It was also possible to prove optimality for the Fu instance in eight seconds. This happens because, as the demand for the items is constrained, all the pieces can be placed inside the board and therefore there is no better solution. For the Blaz2 instance, IPPc-Bin, IPPc-Int and IPPc-IGC found solutions with the same value

Table 6.3: Results reached for IPP with demand constraints.

	IPPc -Bin		IPPc -Int		IPPc -IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	191.5	601.4	192.0	2820.8	**192.5**	963.0
Blaz2	168.0	28.5	168.0	47.6	**168.0**	**6.5**
Shapes0	1024.0	2501.7	1064.0	1458.5	**1072.0**	1782.7
Shapes1	1044.0	2265.7	1048.0	2478.0	**1052.0**	601.2
Fu	om	om	om	om	**1083.0***	1.4
Dagli	om	om	om	om	**2688.1**	2102.4

*: optimal solution.
om: out of memory.

although IPPc -IGC reached this solution more than four times faster than IPPc -Bin and more than seven times faster than IPPc -Int.

At a first sight it would be expected that, for the same instance, the solutions for IPPc to be worse than the solutions for IPP. However, for the Dagli instance, the solution obtained by IPPc -IGC has a better value than the one obtained by IPP -IGC. It happens because with the demand constraints the IPPc solution space can be drastically reduced allowing the solution method make a more thorough search.

Irregular *KP* (IKP)

The irregular knapsack problem (IKP) can be viewed as a special case of IPPc where the demand of each piece is limited to one unit. An instance of IPPc can therefore be easily converted into an IKP instance considering that each demanded item is a piece of a different type. Clearly it is better to consider the pieces of same type together since it reduces the number of constraints used to represent the problem in all formulations. In order to evaluate the performance of the solution method over this problem variant the same instances solved by IPPc were solved by IKP considering different piece copies as different piece types. Table 6.4 presents the computational experiments for IKP and each proposed problem model. The columns of this table have the same type of content as the one of Table 6.1.

For this problem, the binary and integer programs can only solve the instances Blaz1 and Blaz2. For the other instances the memory used exceeds the limit imposed.

The solution values obtained for the instance Blaz1 with IKP -Bin and with IKP -Int are worse than the value obtained with IKP -IGC. For the instance Blaz2 the three models reached the same solution value, however IKP -IGC was able to find this solution faster than the other formulations.

The IKP -IGC model found feasible solutions for all the instances. This happened because using the proposed global constraint to avoid the pieces overlap, the number

Table 6.4: Results reached for IKP.

	IKP-Bin		IKP-Int		IKP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	180.5	996.88	185.5	2889.40	**190.5**	3166.03
Blaz2	168	171.95	168	202.16	**168**	**26.12**
Shapes0	om	om	om	om	**948**	1291.53
Shapes1	om	om	om	om	**1036**	1388.04
Fu	om	om	om	om	**1083**	1.40
Dagli	om	om	om	om	**2572.9**	1278.64

om: out of memory.

of constraints needed to represent the feasible solution space is reduced, consuming less memory. Furthermore, as this constraint propagates faster, good quality solutions could be found, even considering that each demanded piece is of a different type.

As presented in Section 6.5.2, the models for IPPc and IKP are the same. It is possible to observe that no solution found by the IKP program is better than the solution found by IPPc program. This behavior was expected since considering the pieces of same type in the same constraints reduces the number of constraints in the model and avoids symmetric solutions.

6.6.3 Input minimization problems

Different from output maximization problem, in input minimization problems either the size of the board or the number of the boards to be used are unknown. In this case, it is necessary to estimate bounds for the board size or for the number of boards in order to define the grid of dots used to represent the board.

Irregular *1ODP* (I1ODP)

The irregular one open dimension problem (I1ODP) is the variant of irregular cutting and packing problems most exploited on the literature. Table 6.5 presents the results obtained by the I1ODP formulations proposed. The table columns display the same content type of Table 6.1.

The initial length of the board $\overline{L}$ had to be estimated for all the instances, following the procedure presented in Section 6.6.1. The estimated values of $\overline{L}$ are larger then the board width, meaning that, for these instances, more dots are needed to represent the board compared to output maximization problems and therefore the formulations demand more resources to be run. Consequently, the instances Shapes0, Shapes1, Fu and Dagli could only be solved by the model with the NoOverlap constraint. For the Blaz1 instance, the I1ODP-IGC model was able to find a better solution compared

Table 6.5: Results reached for I1ODP.

	I1ODP-Bin		I1ODP-Int		I1ODP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	30.0	2571.9	30.0	893.3	**28.0**	2791.2
Blaz2	23.0	73.9	22.0	50.1	**22.0**	**5.3**
Shapes0	om	om	om	om	**65.0**	673.81
Shapes1	om	om	om	om	**71.0**	1313.3
Fu	om	om	om	om	**34.0**	221.1
Dagli	om	om	om	om	**76.0**	1577.6

om: out of memory.

with the other proposed approaches. Moreover, I1ODP-IGC and I1ODP-Int found a better solution for Blaz2 compared with the one obtained with I1ODP-Bin. To find this solution I1ODP-IGC was about ten times faster than I1ODP-Int.

Irregular *CSP* (ICSP)

The irregular cutting stock problem (ICSP) aims to cut all the pieces from boards using the minimum number of boards. In this paper, the irregular cutting stock problems is formulated similarly to I1ODP minimizing the number of boards used to perform the cuts and minimizing also the used length of the "last" board. The computational results for ICSP are presented in Table 6.6 where the columns present the same type of information as in Table 6.1.

Table 6.6: Results reached for ICSP.

	ICSP-Bin		ICSP-Int		ICSP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	30.0	2746.3	34.0	72.7	**30.0**	**1385.1**
Blaz2	23.0	18.4	23.0	8.3	**23.0**	**5.2**
Shapes0	om	om	om	om	**69.0**	740.3
Shapes1	om	om	om	om	**70.0**	1250.7
Fu	om	om	om	om	**34.0**	63.8
Dagli	om	om	om	om	**76.0**	3081.0

om: out of memory.

As the formulation of ICSP is similar to the I1ODP formulation it is natural to expect that the methods would be able to solve the same instances. In fact, a feasible solution for all the proposed formulations of ICSP could only be found for instances Blaz1 and Blaz2. For the instance Blaz1, ICSP-Bin and ICSP-IGC found a solution with better quality than ICSP-Int. In this case ICSP-IGC was twice faster than ICSP-Bin to find this solution. All the methods reach the same solution value for Blaz2 but ICSP-IGC

was slightly faster then the other methods. For the other instances, ICSP-IGC was able to find feasible solutions within the time limit.

The ICSP formulations are obtained reducing the domains of some of the variables of the I1ODP formulation. For the instance Shapes1, ICSP-IGC found a better solution compared with the one found with the I1ODP formulation. That is because with reduced variable domains ICSP can explore more intensively the search space and eventually find a better solution within the time limit. There are however some solution values reached with I1ODP that could not be reached with ICSP.

Irregular *BPP* (IBPP)

Just like in the ICSP, the irregular bin packing problem (IBPP) aims to cut all the demanded pieces using the minimum number of boards, but the demand of each piece is limited to one unit. The formulation of this problem is equal to the ICSP formulation, but the instances were adapted to represent the problem as described in Section 6.6.2, i.e. considering that each demanded item is a piece of a different type. The results of IBPP are presented in Table 6.7 with the same organization as in Table 6.1.

Table 6.7: Results reached for IBPP.

	IBPP-Bin		IBPP-Int		IBPP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	om	om	om	om	**34.0**	14.2
Blaz2	23.0	312.1	23.0	146.8	**23.0**	5.3
Shapes0	om	om	om	om	**73.0**	3424.9
Shapes1	om	om	om	om	**75.0**	1649.2
Fu	om	om	om	om	**34.0**	63.8
Dagli	om	om	om	om	**78.0**	2453.5

om: out of memory.

IBPP-Bin and IBPP-Int found a solution only for the instance Blaz2. On the other hand IBPP-IGC solved all the instances. All the models reached the same solution value for the instance Blaz2 but IBPP-IGC was more than 20 times faster then IBPP-Int to find the solution and IBPP-Int found the solution 2 times faster than IBPP-Bin. Comparing the solutions of IBPP-IGC with the solutions obtained for equivalent formulation ICSP-IGC for the same set of instances, with ICSP-IGC strictly better solutions were found, except for the instance Fu, because this instance has already a demand of one for all the pieces.

Irregular *2ODP* (I2ODP)

In the irregular two open dimension problem (I2ODP) the two dimensions of the board have to be estimated. Therefore, even if some dots are not considered, as proposed in Section 6.5.5, the number of dots in the I2ODP models is higher than the number of dots in the other input minimization problems presented. Table 6.8 presents the computational results for the I2ODP formulations. The information in the table is organized in a similar way as in Table 6.1.

Table 6.8: Results reached for I2ODP.

	I2ODP-Bin		I2ODP-Int		I2ODP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	520.0	2800.8	494	3184.9	**423.0**	866.2
Blaz2	284.0	2482.6	**280.0**	4.9	288.0	37.9
Shapes0	om	om	om	om	**3024.0**	1080.9
Shapes1	om	om	om	om	**2704.0**	1558.4
Fu	om	om	om	om	**1225.0**	2349.1
Dagli	om	om	om	om	**4290.0**	3557.6

om: out of memory.

The only model that could solve all the instances was I2ODP-IGC. The other two methods were only able to solve the Blaz1 and Blaz2 instances. For Blaz1 instance, the best solution was found by I2ODP-IGC formulation and the worst solution was found by I2ODP-Bin formulation. On the other hand Solving I2ODP-IGC found the worst solution compared to the other formulations for the instance Blaz2. This can happen since the problem is small enough to be well explored by all formulations and some solutions can be explored first in some formulations. I2ODP-IGC was the only formulation able to solve the remaining instances.

6.6.4 Memory usage

An important information when an approach is chosen is the amount of memory that it uses to solve the problems. It is clear that each problem variant uses a different amount of memory since the number of variables and constraints are different. Different problem variants imply on how the domains are exploited and consequently on the memory used by the solution method to perform the search. Notwithstanding the contrast in memory consumption of the problem variants, the main aspects of the proposed constraint programming models can be observed in all the problem variants in different scales. In this section the memory consumption of the proposed constraint programming models is presented for two irregular cutting and packing problem variants, an example of an output maximization problem, the IPP and an example of an

input minimization problem, the I1ODP.

Table 6.9 shows the memory used to solve the instances presented in Section 6.6.1 for the IPP using the three proposed constraint programming models. In the table, the first column shows the instance names. Columns two, three and four respectively shows the memory used (in gigabytes) by the binary formulation, integer formulation and the integer formulation with NoOverlap constraint.

Table 6.9: For the IPP, the Constraint Programming Model with the Global Constraint (IGC) uses significantly less memory than in the other models.

	IPP-Bin	IPP-Int	IPP-IGC
Blaz1	3.50	3.00	0.10
Blaz2	1.40	0.96	0.10
Shapes0	16.60	15.40	0.80
Shapes1	35.80	33.40	1.00
Fu	om	om	0.30
Dagli	om	om	1.60

om: out of memory.

Comparing the amount of memory used by all the methods, it is possible to see that IPP-IGC uses significantly less memory than IPP-Bin and IPP-Int. The low amount of memory required by IPP-IGC is a result of the reduced number of constraints used to avoid pieces overlap. The number of variables of IPP-Int is smaller than in IPP-Bin, however the memory usage is similar in both approaches. This happens because the number of constraints needed to represent the problem in both approaches is similar and the number of constraints is considerably higher than the number of variables. Specifically, as stated in Sections 6.2 and 6.3, the number of constraints needed to ensure the non overlap among pieces in binary and integer formulation is $\sum_{d\in D}\sum_{t=1}^{T}\sum_{t'=t}^{T}\sum_{r=1}^{R_t}\sum_{r'=1}^{R_{t'}}|\Phi^{d}_{t[r],t'[r']}|$ while the number of constraints needed to avoid this overlap using the NoOverlap constraint is $|D|$. Also, the number of variables needed to create the non overlap constraints in binary formulation is $|D|\times\sum_T R_t$ while in the integer formulations only $|D|$ variables are needed.

Naturally, the pattern of memory usage for the IPPc is very similar to the one for the IPP since the formulations are very similar. The problem variant IKP follows also the same pattern but globally the number of constraints and variables is higher, preventing some instances to be solved. The problem variant IIIPP uses other instances, however the pattern is similar to the one presented at Table 6.9.

Input minimization problems usually use more memory than output maximization problems because the number of decision variables and constraints depends the initial number of dots and consequently on the initial estimation (upper bound) determined for the board length.

Table 6.10 represents the memory used for the problem variant I1ODP to solve the instances presented of Table 6.5. The columns of the table has the same type of information described in Table 6.9.

Table 6.10: For the I1ODP, the Constraint Programming Model with the Global Constraint (IGC) uses significantly less memory than in the other models.

	I1ODP-Bin	I1ODP-Int	I1ODP-IGC
Blaz1	9.80	8.60	0.15
Blaz2	2.10	3.00	0.33
Shapes0	om	om	2.00
Shapes1	om	om	1.40
Fu	om	om	0.17
Dagli	om	om	0.84

om: out of memory.

Once again, I1ODP-Bin and I1ODP-Int used considerably more memory to solve the instances compared with I1ODP-IGC. As for this case the need for memory is high, only instances Blaz1 and Blaz2 could be solved by all the methods. As expected, the binary and integer approaches have a similar memory usage while the approach with the NoOverlap constraint needs a small amount of memory compared to them.

The ICSP problem variant is very similar to the I1ODP differing only on the definition of the boards and consequently their memory usage is similar. The difference between IBPP and ICSP is that in IBPP each piece copy is considered as an unique piece type and in ICSP they are considered by types. Therefore, the memory usage has the same pattern of IBPP and ICSP has the same behavior and is higher for IBPP when there are more than one copy of a piece. Finally the I2ODP problem variant has a larger memory consumption compared with I1ODP or ICSP since the board used in this formulation has two dimensions to be estimated. Despite the differences between the formulations, the pattern of memory usage is however the same as the one presented by the other formulations of input minimization problems.

6.7 Solving larger problems and comparing with the literature

Sections 6.6.2 and 6.6.3 showed the flexibility of constraint programming models to solve different irregular cutting and packing problem variants. It was possible to verify that, for the same problem variants, the constraint programming model with the NoOverlap global constraint, IGC, performs better than the other two proposed constraint programming models. Furthermore, as was verified in Section 6.6.4, the memory

usage of IGC is much smaller than the one of the other models. For these two reasons, the IGC was the constraint programming model chosen to solve the larger instances in Section 6.7.1 and to make the comparisons with the literature in Section 6.7.2.

In Section 6.7.1 three problem variants were analyzed, specifically IPP, ICSP and I1ODP. These problem variants were chosen because of their practical relevance and innovativeness of the solution method.

In Section 6.7.2 the results obtained with the IGC for the problem variant I1ODP are compared with results of the Dotted Board Model proposed by Toledo et al. (2013). All the experiments in both sections were run on the framework described in section 6.6.

6.7.1 Solving larger instances

The size of an irregular cutting and packing instance depends on several factors as the number of piece types, the total number of pieces to be placed, the number of allowed piece rotations, the board size and others. In our approach, the number of variables and constraints of the problem is directly related to the discretization of the board, i.e. the number of dots (or admissible positioning points) on the board. In this sense, an instance can be considered small or large depending on the discretization used and, in this section, all the instances are represented using a mesh with refinement $\Delta = 0.5$.

IPP, ICSP and I1ODP, the variants of the irregular cutting and packing problem selected for this phase were chosen because of their relevance in the literature. IPP is a classical problem highly studied on the one-dimensional and regular two dimensional cutting and packing problems. This problem emerges as a cutting pattern generator in column generation techniques to solve the cutting stock problem, which clearly can be extended to the irregular case. Column generation techniques lead to the optimal relaxed solution of the cutting stock problem and a feasible integer solution should be obtained by heuristics or branch-and-price techniques. On the other hand, the ICSP formulation proposed solves the cutting stock problem exactly and each feasible solution found during the search is an integer feasible solution to the problem. Lastly, the I1ODP is the most studied variant among the irregular cutting and packing problems and therefore should always be chosen to be evaluated with large instances.

The instances used are again the ones proposed for the I1ODP problem: Blaz1, Blaz2, Shapes0, Shapes1, Fu, Dagli, Shirts, Jakobs1, Jakobs2. The instances were taken from the ESICUP[4] website. As in the previous section, the lengths of boards for the IPP and the ICSP are defined as equal to the board width. Since the lengths of the solutions for the instances Jakobs1 and Jakobs2 are shorter than the board width, the

[4]EURO Special Interest Group on Cutting and Packing: http://paginas.fe.up.pt/ esicup/

board width considered for these instances was half the one of the original instances to turn them more interesting to be solved by ICSP.

All the instances and problem variants were solved with the constraint programming model using the global constraint (IGC) and the computational results obtained are presented in Table 6.11. The first column has the instance name, the value of the solution and the computational time needed to find this solution with IPP (ICSP and I1ODP) are presented in columns two and three (four and five and six and seven) respectively.

Table 6.11: A feasible solution was found for all the problem variants and instances with $\Delta = 0.5$

	IPP-IGC		ICSP-IGC		I1ODP-IGC	
Instance	Solution	Time to find	Solution	Time to find	Solution	Time to find
Blaz1	203.5	687.4	30.0	1905.3	30.0	193.2
Blaz2	187.5	1717.9	22.0	2400.8	21.5	50.8
Shapes0	1176.0	2995.9	68.0	986.7	65.0	2938.9
Shapes1	1172.0	1124.5	68.0	2605.0	68.0	1867.4
Fu	1430.0	1243.6	34.0	3462.9	34.0	336.3
Dagli	2967.8	2932.6	75.0	524.3	70.0	367.5
Shirts	1815.0	2517.2	65.0	1427.6	65.5	1353.7
Jakobs1	378.5	1506.7	27.0	1729.6	26.5	620.0
Jakobs2	1058.0	2365.3	56.0	144.1	57.0	2983.5

Even using a smaller discretization, a feasible solution has been found for all the problem variants and instances evaluated. Solving the IPP with a smaller discretization produces strictly better solutions compared with the results presented in Table 6.2 (Section 6.6.2) for the Blaz2, Shapes0, Shapes1, Fu and Dagli instances. The solution for the Blaz1 instance is however worse then the one presented in Table 6.2. This may be explained by the dimension of the solution space which is about four times bigger and may therefore, for this specific instance, disturb the search for solutions. For the three remaining instances that were not considered in the previous tests, Shirts, Jakobs1 and Jakobs2, feasible solutions were found.

For the ICSP, a solution with equal or better quality was found for instances Blaz1, Blaz2, Shapes0, Shapes1, Fu and Dagli with a discretization $\Delta = 0.5$ comparing with the solutions presented in Table 6.1 (Section 6.6.3). Also, feasible solutions were found for the Shirts, Jakobs1 and Jakobs2 instances. Problems with thousands of pieces, as solved by Bennell et al. (2015) heuristic, could not be solved with this constraint programming model, but this approach can be an alternative when the cutting stock problem instances are small. Moreover, using this formulation, the used length of each board is minimized, leading to an waste reduction which can be specially advantageous in problems with a relatively small number of pieces.

Using a discretization of $\Delta = 0.5$ for the I1ODP better solutions were obtained for the instances Blaz2, Shapes0, Shapes1, Fu and Dagli, compared with the ones presented in Table 6.5 (Section 6.6.3). In the case of the Blaz1 instance, a solution with worse quality was found. This is possible since the solution space is larger ($\Delta = 0.5$) and better solutions may not be reached within a given time. For the instances Shirts, Jakobs1 and Jakobs2 the proposed formulation for I1ODP was able to find feasible solutions.

Comparing the solutions in columns ICSP-IGC and I1ODP-IGC of Table 6.11 we can observe that for the instances Shirts and Jakobs2 ICSP could find better solutions then the ones found in I1ODP. This behavior is natural since ICSP is more restrict and thus the solution space is smaller. If, in this smaller search space, there exist better solutions for the ICSP and consequently for the I1ODP, they may be reached during a time-limited search.

6.7.2 Comparing with the literature

The results presented in Section 6.6 demonstrate that an exact approach based on constraint programming models is flexible and can be used to solve many variants of cutting and packing problems. A question that arises is how the constraint programming approach for irregular cutting and packing problems compares to other exact methods in the literature. In fact, to the best of our knowledge, the only irregular cutting and packing problem variant that is addressed with exact methods in the literature is the I1ODP and therefore the comparison can only be done with this problem variant. Among the exact approaches proposed in the literature for the I1ODP, we had to choose the one with the same solution space, i.e. one where the reference point of the pieces can only be placed over dots of a discretized board. The Dotted Board Model (DBM) proposed in Toledo et al. (2013) has these characteristics and therefore the solutions obtained by both methods are comparable. However, as the test setting used by Toledo et al. (2013) was different from the one used for I1ODP-IGC, the computational times were not compared.

The test instances used were BLAZEWICZi, $i = 1, ..., 5$, and SHAPESj, $j = 2, 4, 5, 7, 9, 15$ from Toledo et al. (2013).

Table 6.12 compares the computational results obtained by solving the I1ODP with the constraint programming model with the global constraint (I1ODP-IGC) with the ones presented in Toledo et al. (2013) (I1ODP-DBM). The instance name is in the first column of the table. Columns two and three display the solution lengths obtained by the proposed constraint programming method with the global constraint and by the Dotted Board Model.

The solutions obtained by the constraint programming model are always better or

Table 6.12: Comparing the IGC and DBM to solve the I1ODP.

Instances	I1ODP-IGC	I1ODP-DBM
BLAZEWICZ1	8*	8*
BLAZEWICZ2	14	14*
BLAZEWICZ3	20	20*
BLAZEWICZ4	**27**	28
BLAZEWICZ5	**34**	35
SHAPES2	14*	14*
SHAPES4	25	25*
SHAPES5	30	30
SHAPES7	45	45
SHAPES9	54	54
SHAPES15	**65**	67

*: solution was proven optimal within 5 hours.

equal to the ones obtained by the MIP model, however the Dotted Board Model was able to prove optimality for five instances out of 11 while the constraint programming approach was able to prove optimality for only 2 instances.

6.8 Conclusions

This paper presented for the first time in the literature constraint programming models to solve several variants of irregular cutting and packing problems and proposed a new global constraint NoOverlap to guarantee that the pieces do not overlap.

For each problem variant three constraint programming models were presented. The proposed models were the first in the literature that could solve some instances to optimality.

The constraint programming models use two types of variables, variables with binary and integer domains. The models with binary variables are based on the dotted board model Toledo et al. (2013) where a binary variable is defined for each dot, piece type and piece rotation. In the models with integer domain variables there is only one variable defined for each dot and the domain of the variable represents the piece types and the rotations. To smartly use all the information contained on the integer variable domains a new global constraint NoOverlap was proposed. This global constraint ensures that the pieces do not overlap and is tailored to the problem, implying in a faster propagation. The computational results showed the effectiveness of this new global constraint solving all the problem variants.

Constraint programming is very flexible to model combinatorial optimization problems allowing the use of linear, non-linear or logical constraints to represent the solution space of the problems. Therefore, all the proposed models may be adaptable to real

problems where frequently some additional requirements are needed in a solution.

The proposed models have constraints general enough to be used in other problem variations. As there are some differences in these problem variants, an interesting direction for future work is to investigate each problem variation and develop new global constraints tailored for each problem variant.

Chapter 7

Conclusions and research directions

This thesis addressed the two-dimensional irregular cutting and packing problems which consist of cutting convex and non-convex pieces from a board. Mathematical models and heuristics were proposed to solve the problem and also new geometric structures were developed to support the creation of the methods.

To solve the two-dimensional irregular strip packing problem, two mixed integer programming models were proposed in Chapter 3. These models consider that the placement of the pieces on the board is continuous, i.e., the reference point of the pieces can be placed in any position on the board. To attend the problem constraints, the pieces are decomposed in convex polygons resulting in robust models that can consider piece holes and narrow entries in their default formulation. The first model, direct trigonometry model (DTM), avoids the overlap among pieces using only the information about the piece vertices. In the literature, all the exact methods use the nofit polygon to avoid the overlap among pieces, therefore, the DTM is simpler to be modeled when geometric tools to generate the nofit polygons are not available. Nevertheless, this model achieved competitive results compared to models that use nofit polygons. The nofit polygon covering model ($NFP-CM$) avoids the overlap among pieces using a covering of the nofit polygon. This covering is obtained by generating the nofit polygon among the convex parts of each pair of pieces and composing the nofit polygon of the entirely pieces with them. The $NFP-CM$ outperforms the best results obtained using mixed integer programming models in literature. Both models allow piece rotations in a finite number of angles. There are no other mixed integer programming models in the literature that allow the pieces to be rotated. This chapter originated the following research paper:

- 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*, Available online 11 March 2016.

In Chapter 4, a new dot data structure was proposed to handle the geometry of irregular cutting and packing problems where pieces should be placed over a finite set of dots. This structure is composed by a list of dots which specifies the pieces that can be placed in each dot attending the innerfit polygon constraints. Furthermore, for each possible placement of the pieces, the information of the overlap among pieces for this dot is kept. Therefore, the dot structure embeds the most challenging geometric issues of the irregular cutting and packing problems making easier the task of developing mathematical programming models and heuristics that are based on this type of geometry. Using this structure, it is possible to introduce a distinct set of dots for the pieces, i.e., each piece may have its own set of dots to be placed. Since each piece can be placed in a specific set of dots, using the features of each instance a mesh generation rule can be developed leading the solution methods to obtain better quality solutions in less computational time. The worst case complexity analysis, in time and in space, was calculated for each algorithm needed to build the structure. The dot data structure is general enough to be used in any problem variant classified by Wäscher et al. (2007) and can provide more flexibility for some existing exact and heuristic methods.

Chapter 4 resulted in the following research paper:

- Cherri, L. H., Cherri, A. C., Carravilla, M. A., Oliveira, J. F., Toledo, F. M. B., Vianna, A. C. G., An innovative data structure to handle the geometry of nesting problems, *Submitted for publication, 2016.*

Using the dot data structure, a model based heuristic to solve the irregular strip packing problem was proposed in Chapter 5. The heuristic is deterministic and has three phases. In the first phase, the dotted-board model is used to construct an initial feasible solution and, in the second phase, the quality of this solution is improved. As the dotted-board model can generate layouts with gaps among pieces, in the last phase a compaction model is used to reduce these gaps. Computational experiments showed that the procedure reached more compact layouts for the problem compared with the solutions obtained by the dotted-board model and the model proposed by Alvarez-Valdes et al. (2013) in a fixed amount of time, indicating that combining continuous and discrete models is promising. In addition, compared to the state-of-art heuristics, the proposed method found solutions with quality 6% worse in average. However, considering that the proposed matheuristic is a deterministic procedure, its running time is at least six times smaller than the other heuristics. The outcome of this chapter resulted in the following research paper:

- Cherri, L. H., Carravilla, M. A., Toledo F. M. B., A model based heuristic for the irregular strip packing problem. *Under review, 2016.*

Concerning the other variants of the two-dimensional irregular cutting and packing problems, in Chapter 6 new constraint programming models were proposed. Specifically, for each problem variant classified by Wäscher et al. (2007), three models were developed. Considering these models, one has binary domain variables, similarly to the ones used in the dotted-board model. The other two models consider variables with integer domains and they differ in how the overlap among pieces is avoided. For five out of six classified variants, there was no exact solution method nor a mathematical model in the literature, therefore, the proposed models were the first exact representation for these variants. Moreover, a new global constraint to eliminate the overlap among pieces was proposed. Using this constraint, the amount of memory used to represent the models and the time that the solver took to solve this model is significantly reduced. Comparing the results obtained by the proposed models and by the dotted-board model for the irregular strip packing problem, the constraint programming formulation founds solutions with equal or better quality within the given time limit. However, the dotted-board proved the optimality of more instances. Chapter 6 resulted in the following research paper:

- Cherri, L. H., Carravilla, M. A., Ribeiro, C., Toledo F. M. B., Optimality in irregular cutting and packing problems: new constraint programming models. *Submitted for publication, 2016.*

7.1 Research directions

Considering the outcomes of this thesis, some research directions emerge. We identified some promising research topics connected to each chapter of this thesis that will be described now.

In Chapter 3, two mixed integer programming models for the irregular strip packing problem were proposed. Both models can have their performance improved if new valid inequalities and cuts are introduced or a specialized branch-and-cut algorithm for the model is developed. In addition, as the DTM does not use nofit polygons to avoid the overlap among pieces, it can be rewritten to consider the continuous rotations for the pieces resulting in a mixed integer non-linear programming model. As well as DTM, this new model will need only the information of the piece vertices to be constructed. In the literature, all the models that consider continuous rotations for the pieces use phi-functions or a set of circles to represent the pieces. In this sense a non-linear model based on DTM is promising, demanding simpler structures to be built.

In Chapter 4, a new data structure to represent the geometry of the irregular cutting and packing problems was proposed. Using this structure new meshes of dots can be defined leading the solution methods to obtain solutions with better quality in smaller

computational time. Furthermore, specific meshes could make possible to prove that the optimal solution of the problem with a finite set of placement positions for the pieces has the same quality of a solution obtained by the problem where the pieces can be continuously placed on the board. Such characterization of placement dots has been already explored for rectangle packing problems (Herz, 1972; Scheithauer, 1997; Birgin et al., 2008).

In Chapter 5, a model based heuristic to solve the irregular strip packing problem was proposed. This heuristic can be adapted to solve other irregular cutting and packing problem variants. In addition, the proposed matheuristic can inspire the development of new ones which can be based on the exact methods proposed in this thesis (Chapter 3 and Chapter 6).

In Chapter 6, constraint programming models were proposed to solve several variants of the irregular cutting and packing problems. These models are constructed using general constraints that allow more intuitive representations for all problems. To improve these models, each formulation could be individually investigated aiming to propose constraints based on the problem structure. Moreover, the integration of constraint programming with other optimization techniques should be evaluated.

Appendix A: The pieces in metal instance

The pieces vertices are presented below.

```
Piece 1
Number of vertices: 4
Vertices:
0.00    0.00
256.00  0.00
256.00  144.00
0.00    144.00
Number of holes: 0

Piece 2
Number of vertices: 4
Vertices:
0.00    0.00
100.00  0.00
100.00  120.00
0.00    120.00
Number of holes: 0

Piece 3
Number of vertices: 4
Vertices:
30.00   30.00
30.00   -198.00
-215.00 -198.00
-215.00  30.00
Number of holes: 1
Number of hole vertices: 4
Vertices:
0.00     0.00
-185.00  0.00
-185.00  -168.00
0.00     -168.00

Piece 4
Number of vertices: 4
Vertices:
0.00    0.00
0.00    -70.00
184.00  -70.00
184.00  0.00
Number of holes: 0

Piece 5
Number of vertices: 8
Vertices:
0.00    0.00
28.00   0.00
45.00   17.00
45.00   45.00
28.00   62.00
0.00    62.00
-17.00  45.00
-17.00  17.00
Number of holes: 0

Piece 6
Number of vertices: 6
Vertices:
24.00   38.00
24.00   -92.00
-84.00  -202.00
-236.00 -202.00
-260.00 -184.00
-260.00 38.00
Number of holes: 1
```

```
Number of hole vertices: 5
0.00    0.00
-228.00 0.00
-228.00 -180.00
-94.00  -180.00
0.00    -84.00

Piece 7
Number of vertices: 4
Vertices:
0.00    0.00
100.00  0.00
100.00  150.00
0.00    150.00
Number of holes: 0

Piece 8
Number of vertices: 8
Vertices:
0.00    0.00
0.00    -136.00
12.00   -156.00
262.00  -156.00
286.00  -130.00
286.00  -106.00
268.00  -82.00
32.00   10.00
Number of holes: 0

Piece 9
Number of vertices: 4
Vertices:
0.00    0.00
160.00  0.00
160.00  110.00
0.00    110.00
Number of holes: 0

Piece 10
Number of vertices: 4
Vertices:
0.00    0.00
0.00    96.00
154.00  96.00
154.00  0.00
Number of holes: 0
```

Bibliography

Albano, A., Sapuppo, G.. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. IEEE Transactions on Systems, Man, and Cybernetics 1980;SMC-10(5):242–248.

Alvarez-Valdes, R., Martinez, A., Tamarit, J.. A branch & bound algorithm for cutting and packing irregularly shaped pieces. International Journal of Production Economics 2013;145(2):463 – 477.

Alves, C., Brás, P., de Carvalho, J.V., Pinto, T.. New constructive algorithms for leather nesting in the automotive industry. Computers & Operations Research 2012;39(7):1487 – 1505.

Art, R.C.. An approach to the two dimensional irregular cutting stock problem. Technical Report 36. Y08; IBM Cambridge Scientific Center; Cambridge, Massachusetts, USA; 1966.

Babu, A., Babu, N.. A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms. Computer-Aided Design 2001;33(12):879–891.

Baldacci, R., Boschetti, M.A., Ganovelli, M., Maniezzo, V.. Algorithms for nesting with defects. Discrete Applied Mathematics 2014;163, Part 1(0):17 – 33.

Bennell, J., Dowsland, K.. Hybridising tabu search with optimisation techniques for irregular stock cutting. Management Science 2001;47(8):1160–1172.

Bennell, J., Scheithauer, G., Stoyan, Y., Romanova, T., Pankratov, A.. Optimal clustering of a pair of irregular objects. Journal of Global Optimization 2015;61(3):497–524.

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

Bennell, J.A., Oliveira, J.F.. A tutorial in irregular shape packing problems. Journal of the Operational Research Society 2009;.

Birgin, E.G., Lobato, R.D., R., M.. An effective recursive partitioning approach for the packing of identical rectangles in a rectangle. Journal of the Operational Research Society 2008;61:306–320.

Bounsaythip, C., Maouche, S.. Irregular shape nesting and placing with evolutionary approach. In: Systems, Man, and Cybernetics. volume 4; 1997. p. 3425–3430.

Burke, E., Hellier, R., Kendall, G., Whitwell, G.. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. Operations Research 2006;54(3):587–601.

Carravilla, M.A., Ribeiro, C.. 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 2005;:113–127.

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

Chernov, N., Stoyan, Y., Romanova, T.. Mathematical model and efficient algorithms for object packing problem. Computational Geometry 2010;43(5):535 – 553.

Cherri, L.H., Mundim, L.R., Andretta, M., Toledo, F.M., Oliveira, J.F., Carravilla, M.A.. Robust mixed-integer linear programming models for the irregular strip packing problem. European Journal of Operational Research 2016;:1–33.

Clautiaux, F., Jouglet, A., Carlier, J., Moukrim, A.. A new constraint programming approach for the orthogonal packing problem. Computers & Operations Research 2008;35(3):944 – 959.

Crispin, A., Clay, P., Taylor, G., Bayes, T., Reedman, D.. Genetic algorithm coding methods for leather nesting. Applied Intelligence 2005;23(1):9–20.

Cuninghame-Green, R.. Geometry, shoemaking and the milk tray problem. New Scientist 1989;(1677):50–53.

Dalalah, D., Khrais, S., Bataineh, K.. Waste minimization in irregular stock cutting. Journal of Manufacturing Systems 2014;33(1):27 – 40.

Dolan, E.D., Moré, J.J.. Benchmarking optimization software with performance profiles. Mathematical Programming 2002;91(2):201–213.

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 1998;49(6):647–658.

Dowsland, K.A., Vaid, S., Dowsland, W.B.. An algorithm for polygon placement using a bottom-left strategy. European Journal of Operational Research 2002;141(2):371–381.

Dyckhoff, H., Finke, U.. Cutting and packing in production and distribution: Typology and bibliography. Heidelberg: Springler-Verlag, 1992.

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

Elkeran, A.. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. European Journal of Operational Research 2013;231(3):757 – 769.

Fischetti, M., Luzzi, I.. Mixed-integer programming models for nesting problems. Journal of Heuristics 2009;15(3):201–226.

Flisberg, P., Lidén, B., Rönnqvist, M.. A hybrid method based on linear programming and tabu search for routing of logging trucks. Computers & Operations Research 2009;36(4):1122–1144.

Fowler, R.J., Paterson, M., Tanimoto, S.L.. Optimal packing and covering in the plane are np-complete. Information Processing Letters 1981;12(3):133–137.

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 1993;65(1):477–484.

Gomes, A.M., Oliveira, J.F.. A 2-exchange heuristic for nesting problems. European Journal of Operational Research 2002;141(2):359 – 370.

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

Greene, D.H.. The decomposition of polygons into convex parts. Computational Geometry 1983;1:235–259.

Herz, J.C.. Recursive computational procedure for two-dimensional stock cutting. IBM Journal of Research and Development 1972;16(5):462–469.

Hopper, E.. Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods. Ph.D. thesis; University of Wales, Cardiff; 2000.

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

Jaffar, J., Maher, M.J.. Constraint logic programming: a survey. The Journal of Logic Programming 1994;19–20, Supplement 1(1):503 – 581.

Jakobs, S.. On genetic algorithms for the packing of polygons. European Journal of Operational Research 1996;88(1):165–181.

Kovács, A., Beck, J.C.. A global constraint for total weighted completion time for cumulative resources. Engineering Applications of Artificial Intelligence 2008;21(5):691 – 697.

Lamousin, H., Waggenspack, W.N.. Nesting of two-dimensional irregular parts using a shape reasoning heuristic. Computer-Aided Design 1997;29(3):221 – 238.

Leao, A.A.S., Toledo, F.M.B., Oliveira, J.F., Carravilla, M.A.. A semi-continuous mip model for the irregular strip packing problem. International Journal of Production Research 2016;54(3):712–721.

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 2012;39(3):678–686.

Li, Z., Milenkovic, V.. Compaction and separation algorithms for nonconvex polygons and their applications. European Journal of Operational Research 1995;84(3):539–561.

López-Camacho, E., Ochoa, G., Terashima-Marín, H., Burke, E.. An effective heuristic for the two-dimensional irregular bin packing problem. Annals of Operations Research 2013;206(1):241–264.

Maniezzo, V., Stützle, T., Vob, S.. Matheuristics: Hybridizing Metaheuristics and Mathematical Programming. volume 1 of *Annals of Information Systems*. Springer, 2009.

Marques, V.M.M., Bispo, C.F.G., Sentieiro, J.J.S.. A system for the compaction of two-dimensional irregular shapes based on simulated annealing. In: Industrial Electronics, Control and Instrumentation. volume 3; 1991. p. 1911–1916.

Martello, S., Monaci, M., Vigo, D.. An exact approach to the strip–packing problem. NFORMS Journal on Computing 2003;3(15):310–319.

Muller, L.F., Spoorendonk, S., Pisinger, D.. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. European Journal of Operational Research 2012;218(3):614–623.

Nielsen, B.K., Odgaard, A.. Fast neighbourhood search for nesting problem. Technical Report 03/02; DIKU, Department of Computer Science, University of Copenhagen; 2003.

Okano, H.. A scanline-based algorithm for the 2d free-form bin packing problem. Journal of the Operations Research Society of Japan 2002;:145–161.

Oliveira, J.F., Ferreira, J.. Algorithms for nesting problems, applied simulated annealing. In Vidal, RVV (Ed, Lecture Notes in Economics and Maths Systems) 1993;396(1):255–274.

Oliveira, J.F., Gomes, A.M., Ferreira, J.S.. TOPOS - A new constructive algorithm for nesting problems. OR Spektrum 2000;22(2):263–284.

Ribeiro, C., Carravilla, M.A.. A global constraint for nesting problems. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer Berlin Heidelberg; volume 3011 of *Lecture Notes in Computer Science*; 2004. p. 256–270.

Rodrigues, M.O.. Modelos matemáticos para o problema de empacotamento em faixas de peças irregulares. Master's thesis; Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo; 2015.

Salas, I., Chabert, G., Goldsztejn, A.. The non-overlapping constraint between objects described by non-linear inequalities. In: Principles and Practice of Constraint Programming. Springer International Publishing; volume 8656 of *Lecture Notes in Computer Science*; 2014. p. 672–687.

Saldanha, R., Morgado, E.. Solving set partitioning problems with global constraint propagation. In: Progress in Artificial Intelligence. Springer Berlin Heidelberg; volume 2902 of *Lecture Notes in Computer Science*; 2003. p. 101–115.

Sato, A.K., Martins, T.C., Tsuzuki, M.S.G.. An algorithm for the strip packing problem using collision free region and exact fitting placement. Computer-Aided Design 2012;44(8):766–777.

Scheithauer, G.. Equivalence and dominance for problems of optimal packing of rectangles. Recierca Operativa 1997;27:3–34.

Song, X., Bennell, J.A.. Column generation and sequential heuristic procedure. Journal Operations Research Society 2014;65(7):1037–1052.

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

Terashima-Marín, H., Ross, P., Farías-Zárate, C., López-Camacho, E., Valenzuela-Rendón, M.. Generalized hyper-heuristics for solving 2d regular and irregular packing problems. Annals of Operations Research 2010;179(1):369–392.

Cgal, . Cgal, Computational Geometry Algorithms Library. Http://www.cgal.org.

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 2013;145(2):478 – 487.

Trojet, M., H'Mida, F., Lopez, P.. Project scheduling under resource constraints: Application of the cumulative global constraint in a decision support framework. Computers & Industrial Engineering 2011;61(2):357 – 363.

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 2009;16(6):661–683.

Valle, A.M.D., de Queiroz, T.A., Miyazawa, F.K., Xavier, E.C.. Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape. Expert Systems with Applications 2012;39(16):12589–12598.

Wang, S., Chen, J., Wang, K.J.. Resource portfolio planning of make-to-stock products using a constraint programming-based genetic algorithm. Omega 2007;35(2):237 – 246.

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

Woodcock, A.J., Wilson, J.M.. A hybrid tabu search/branch & bound approach to solving the generalized assignment problem. European Journal of Operational Research 2010;207(2):566–578.