UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Mathematical models and heuristic methods for nesting problems

Leandro Resende Mundim

Tese de Doutorado do Programa de Pós-Graduação em Ciências de
Computação e Matemática Computacional (PPG-CCMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Leandro Resende Mundim

Mathematical models and heuristic methods for nesting problems

Doctoral dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Doctorate Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Marina Andretta

Co-advisor: Prof. Dr. José Fernando da Costa Oliveira

USP – São Carlos
September 2017

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

M965m Mundim, Leandro Resende
Mathematical models and heuristic methods for
nesting problems / Leandro Resende Mundim;
orientador Marina Andretta; coorientador José
Fernando da Costa Oliveira. -- São Carlos, 2017.
166 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, 2017.

1. Irregular Cutting and Packing Problems. 2.
Mathematical Programming. 3. Heuristics. I.
Andretta, Marina, orient. II. Oliveira, José
Fernando da Costa, coorient. III. Título.

Leandro Resende Mundim

**Modelos matemáticos e métodos heurísticos para os
problemas de corte de itens irregulares**

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Profa. Dra. Marina Andretta

Coorientador: Prof. Dr. José Fernando da Costa Oliveira

**USP – São Carlos
Setembro de 2017**

To my family and my friends.

ACKNOWLEDGEMENTS

Agradeço a Deus por ter me dado saúde e tranquilidade para a conclusão deste trabalho. Quero agradecer a Bandeira da minha família: São João Batista, Santo Antônio e São Pedro, que sempre nos cuidam, protegem e nos guardam de todo mal.

Os agradecimentos principais são direcionados aos meus pais, meus avós, minha irmã e demais familiares. Devo estender, os agradecimentos a todos os meus amigos, colegas e conhecidos.

Sendo o principal autor desta tese, devo confessar que só atingi a qualidade aqui apresentada devido a meus queridos orientadores: Marina Andretta e José Fernando. Eles foram fundamentais para o desenvolvimento deste trabalho e meu crescimento profissional. Demais pesquisadores que contribuíram em partes pontuais deste trabalho e na minha formação foram: Thiago Queiroz, Maria Antônia, Franklina Toledo e Luiz Cherri.

Um agradecimento especial ao ICMC e a CAPES que acreditaram no meu trabalho, durante o mestrado e doutorado.

“He who thinks little, errs much.”
(Leonardo da Vinci)

ABSTRACT

MUNDIM, L. R. **Mathematical models and heuristic methods for nesting problems**. 2017. 166 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Irregular cutting and packing problems, with convex and non-convex polygons, are found in many industries such as metal mechanics, textiles, of shoe making, the furniture making and others. In this thesis we study the two-dimensional version of these problems, where we want to allocate a set of items, without overlap, inside one or more containers, limited or unlimited, so as to optimize an objective function. In this document we study the knapsack problem, placement problem, strip packing problem, cutting stock problem and bin packing problem. For these problems, the heuristic methods and mathematical programming models are proposed and presented very promising results, surpassing in many cases the best results in the specialized literature.

This thesis is organized as follows. In Chapter 1, we present a review of the studied problems, the value proposition for this thesis with the main contributions and ideas. In Chapter 2, we propose a metaheuristic for the strip packing problem with irregular items and circles. Then, in Chapter 3, we present a generic heuristic for the allocation of irregular items that may be weakly or strongly heterogeneous and will be allocated in a container (output maximization problems) or multiple containers (input minimization problems). In Chapter 4, we propose a solution method for the cutting stock problem with deterministic demand and stochastic demand. In Chapters 5 and 6, we present mathematical programming models for the strip packing problem. Finally, in Chapter 7, we present a conclusion and a concise direction for future works.

Keywords: Irregular Cutting and Packing Problems, Mathematical Programming, Heuristics.

RESUMO

MUNDIM, L. R. **Modelos matemáticos e métodos heurísticos para os problemas de corte de itens irregulares**. 2017. 166 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2017.

Os problemas de corte e empacotamento de itens irregulares, polígonos convexos e não convexos, são encontrados em diversas indústrias, tais como a metal-mecânica, a têxtil, a de calçados, a moveleira e outras. Nesta tese estudamos a versão bidimensional destes problemas, na qual desejamos alocar um conjunto de itens, sem sobreposição, no interior de um ou mais recipientes, limitados ou ilimitados, de modo a otimizar uma função objetivo. Neste trabalho estudamos o problema da mochila, o problema do assentamento, o problema de empacotamento em faixa, o problema de corte de estoque e o problema de empacotamento de contêineres. Para estes problemas, os métodos heurísticos e modelos de programação matemática propostos e apresentam resultados muito promissores, ultrapassando em muitos casos os melhores resultados da literatura especializada.

Esta tese está organizada da seguinte maneira. No Capítulo 1, apresentamos uma revisão dos problemas estudados, a proposta de valor deste doutorado com as principais contribuições e ideias. No Capítulo 2, propomos uma meta-heurística para o problema de empacotamento em faixa para itens irregulares e círculos. Em seguida, no Capítulo 3 apresentamos uma heurística genérica para a alocação de itens irregulares que podem ser fracamente ou fortemente heterogêneos e serão alocados em um recipiente (problema de maximização de saída) ou de múltiplos recipientes (problemas de minimização de entrada). O Capítulo 4 propõe um método de solução para o problema de corte de estoque com demanda conhecida e demanda estocástica. Nos Capítulos 5 e 6 apresentamos modelos de programação matemática para o problema de corte de itens irregulares em faixa. Finalmente, no Capítulo 7, apresentamos a conclusão e uma sucinta direção para os trabalhos futuros.

Palavras-chave: Problemas de Corte e Empacotamento de Itens Irregulares, Programação Matemática, Métodos Heurísticos.

LIST OF FIGURES

Figure 1 – Types of basic problems, extracted from Wäscher, Haußner and Schumann (2007).	27
Figure 2 – Representation extracted from Oliveira and Ferreira (1993).	29
Figure 3 – Example of phi-function of circles i and j	29
Figure 4 – Values of D-function and its implications with the classical coordinate system, i.e. x-coordinates grow to the right and y-coordinates grow upwards. . .	30
Figure 5 – Example of the relative position of two items via direct trigonometry, extracted from Oliveira and Ferreira (1993).	30
Figure 6 – The no-fit polygon for convex polygons, extracted from Burke <i>et al.</i> (2007).	31
Figure 7 – Representation of polygons by circles, extracted from Jones (2014).	31
Figure 8 – Bottom-left heuristics: (a) without allocating items in holes, (b) allocating items in holes. Extracted from Dowsland, Vaid and Dowsland (2002)	32
Figure 9 – Evaluation criteria for layouts. Extracted from Oliveira, Gomes and Ferreira (2000).	33
Figure 10 – Criteria for ordering items. Extracted from Oliveira, Gomes and Ferreira (2000).	33
Figure 11 – Neighborhood of the sequence proposed by Gomes and Oliveira (2002). . .	34
Figure 12 – Example of horizontal (a) and vertical (b) search, to obtain a new solution (c). Extracted from Egeblad, Nielsen and Odgaard (2007).	35
Figure 13 – Vision of the cuckoo search algorithm proposed by Elkeran (2013).	36
Figure 14 – Example of irregular bin from the leather cutting industry presented by Alves <i>et al.</i> (2012).	36
Figure 15 – The blue regions represent the positions where the item can be allocated in (a) discrete methods, (b) semi-continuous methods and (c) continuous methods. . .	37
Figure 16 – Four irregular shaped items according to Segenreich and Braga (1986). . .	48
Figure 17 – The inner-fit raster of some irregular shaped items are represented by the blue dots in the bin's grid. Note that the inner-fit raster of each item contains only points of the grid where such items can be packed generating a feasible solution.	48
Figure 18 – Examples of no-fit raster, which are the blue dots (feasible positions) to pack, while the unfeasible positions are marked by black dots and x	49
Figure 19 – Example of complex items that can use the concept of no-fit raster to check overlap.	50

Figure 20 – Example of one chromosome associating items and keys.	52
Figure 21 – Sorting criterion applied to the key vector to obtain a new sequence for packing items.	53
Figure 22 – Items are packed observing the sequence 3, 4, 2, 1, and 4 using the bottom-left heuristic.	54
Figure 23 – Items are packed observing the sequence 3, 4, 2, 1, and 4 using the bottom-left rectangle algorithm.	54
Figure 24 – BRKGA generating the population of the next generation $K + 1$ from generation K	57
Figure 25 – Example illustrating the parametrized uniform crossover for the BRKGA.	58
Figure 26 – Solutions computed with BRKGA70 for SHAPES7 and SHAPES9.	61
Figure 27 – Average time in seconds over all the instances.	64
Figure 28 – Overview of BRKGA70 results in terms of the best solution and best average occupation in comparison with the literature.	65
Figure 29 – Best solutions computed so far by BRKGA70 for the benchmark instances from ESICUP.	65
Figure 30 – Solutions computed with BRKGA70 for the instances with circles.	67
Figure 31 – Solutions computed for the 2D two-open dimension problem.	69
Figure 32 – Illustration of the items.	70
Figure 33 – Illustration of the items.	71
Figure 34 – Examples of no-fit polygons and the corresponding no-fit rasters.	77
Figure 35 – Examples for inner-fit polygons and inner-fit rasters.	78
Figure 36 – The six placement rules used in the H4NP.	80
Figure 37 – a) Number of best solutions for the LPP; (b) Number of best solutions for the UPP; (c) Percentage of best solutions for the PP (LPP+UPP); (d) Relative deviation to the best heuristic for the PP (LPP+UPP).	83
Figure 38 – Result for the BLFP instance obtained by the random corner placement rule.	84
Figure 39 – Instance Fu adapted by Terashima-Marín <i>et al.</i> (2010).	93
Figure 40 – H4NP obtained equal or better solutions in 73% of the benchmark instances.	95
Figure 41 – Best solutions computed with the H4NP for the UPP.	97
Figure 42 – Best solutions computed with H4NP for the LPP.	98
Figure 43 – Best solutions for the KP computed with the H4NP.	99
Figure 44 – The placement rules used in the H4NP extracted from Chapter 3	104
Figure 45 – Configuration examples.	106
Figure 46 – Representation of a piece and its distance parameters in NFP-CM.	121
Figure 47 – Outer no-fit polygon partitions. (a) presents the partition used in Cherri <i>et al.</i> (2016) and (b) presents the partition used in Alvarez-Valdes, Martinez and Tamarit (2013)	122
Figure 48 – Non-overlap regions using the proposed partitioning.	124

Figure 49 – Colinear lines.	124
Figure 50 – Co-linear lines. The hatched area is the placement region if the co-linear lines are activated	125
Figure 51 – Optimal solutions proved by iNFP-CM and not proved for NFP-CM in less than 3,600 seconds.	130
Figure 52 – Overview about the results in comparasion with the literature.	131
Figure 53 – Example of a striped trouser cut	135
Figure 54 – Item <i>i</i> and its reference point. Extracted from Alvarez-Valdes, Martinez and Tamarit (2013)	136
Figure 55 – Different no-overlap regions of NFP-CM and HS2.	137
Figure 56 – Overview about the results in the discrete domain.	141
Figure 57 – Overview about the results in the semi-continuous domain.	143
Figure 58 – Solutions strictly better than the semi-continuous model leao	143
Figure 59 – My PhD document.	146

LIST OF TABLES

Table 1 – Parameters used by Algorithm 1	57
Table 2 – Results for the instances of Toledo <i>et al.</i> (2013).	60
Table 3 – Comparison with the recent approaches from the literature.	60
Table 4 – Details of the benchmark instances from ESICUP.	61
Table 5 – Results for the benchmark instances from ESICUP.	62
Table 6 – Comparison with the literature for the benchmark instances from ESICUP. . .	63
Table 7 – Results for the instances with circles.	66
Table 8 – Results for the 2D two-open dimension problem on benchmark instances. . .	68
Table 9 – Demand of the instances proposed by Błażewicz, Hawryluk and Walkowiak (1993).	71
Table 10 – Overview of the C&P problem variants studied in this chapter.	74
Table 11 – Instances for the Limited Placement Problem (LPP).	85
Table 12 – Instances proposed by Valle <i>et al.</i> (2012) for the UPP.	85
Table 13 – Results for the LPP instances	87
Table 14 – Results for the UPP instances.	88
Table 15 – Description of the KP instances.	89
Table 16 – Results for the KP instances.	89
Table 17 – Description of the CSP instances.	91
Table 18 – Results for the CSP instances.	92
Table 19 – Description of the Terashima 1 and Terashima 2 classes of instances.	93
Table 20 – Results obtained with H4NP, with grids 10:1 and 1:1, for the instances Terashima 1.	94
Table 21 – Results obtained with H4NP, with grids 10:1 and 1:1, for the instances Terashima 2.	94
Table 22 – Instances data from Song and Bennell (2014).	110
Table 23 – Comparison between EEP and the methods from Song and Bennell (2014). .	111
Table 24 – Comparison between H4NP and EEP.	111
Table 25 – Instances data from Valle <i>et al.</i> (2012).	112
Table 26 – Comparison between EEP, CGH from Valle <i>et al.</i> (2012) and H4NP from Chapter 3	113
Table 27 – Instances stochastic for CSP.	115
Table 28 – Results for stochastic demand with 1000 patterns and $d_{is} = N(1000, \sigma^2)$. . .	116
Table 29 – Features of benchmark instances.	128

Table 30 – Comparison between MIP models. 129

Table 31 – Comparison with the literature. 130

Table 32 – Instances used in the computational experiments. 139

Table 33 – Comparing the results with the dotted-board model from Toledo *et al.* (2013). 140

Table 34 – Comparing the semi-continuous model with HS2 and NFP-CM adapted to
semi-continuous domains. 142

CONTENTS

1	INTRODUCTION	25
1.1	Cutting and packing problems	26
1.1.1	<i>Typology of cutting and packing problems</i>	26
1.2	Geometry of irregular items and the best known resolution methods	28
1.2.1	<i>Geometry of the nesting problems</i>	28
1.2.2	<i>Main resolution methods for nesting problems</i>	32
1.3	Main contributions to the nesting problems literature	38
1.4	Thesis organization	38
1.4.1	<i>Chapter 2</i>	39
1.4.2	<i>Chapter 3</i>	39
1.4.3	<i>Chapter 4</i>	40
1.4.4	<i>Chapter 5</i>	41
1.4.5	<i>Chapter 6</i>	42
1.4.6	<i>Chapter 7</i>	42
2	A BIASED RANDOM KEY GENETIC ALGORITHM FOR OPEN DIMENSION NESTING PROBLEMS USING NO-FIT RASTER . .	43
2.1	Introduction	44
2.2	Computational geometry	47
2.3	Mathematical Models	49
2.4	Developed Heuristic	51
2.4.1	<i>Coding and Decoding Chromosomes</i>	52
2.4.2	<i>BRKGA</i>	54
2.5	Computational Experiments	56
2.5.1	<i>Parameter tuning</i>	57
2.5.2	<i>Results for the 2D one-open dimension problem</i>	59
2.5.2.1	<i>Instances in Toledo et al. (2013)</i>	59
2.5.2.2	<i>Benchmark instances</i>	61
2.5.2.3	<i>Instances with circles</i>	64
2.5.3	<i>Results for the 2D two-open dimension problem</i>	66
2.6	Conclusions	67
2.7	Data of Instances with Circles	70
2.7.1	<i>Instances proposed in Segenreich and Braga (1986)</i>	70

2.7.2	<i>Instances proposed in Błażewicz, Hawryluk and Walkowiak (1993)</i>	70
3	A GENERAL HEURISTIC FOR TWO-DIMENSIONAL NESTING PROBLEMS WITH LIMITED-SIZE CONTAINERS	73
3.1	Introduction	73
3.2	Review of the literature	75
3.3	Geometry	76
3.3.1	<i>No-fit raster (NFR)</i>	76
3.3.2	<i>Inner-fit raster (IFR)</i>	77
3.4	Solution method	78
3.4.1	<i>Constructive heuristic</i>	78
3.4.2	<i>Placement rules</i>	79
3.4.3	<i>H4NP, the Heuristic for Nesting Problems</i>	80
3.5	Computational experiments	80
3.5.1	<i>Parameter tuning</i>	81
3.5.1.1	<i>Computer and programming language</i>	81
3.5.1.2	<i>Piece types</i>	81
3.5.1.3	<i>Determining the number of iterations</i>	82
3.5.1.4	<i>Selection of placement rules</i>	82
3.5.2	<i>Placement Problem</i>	84
3.5.2.1	<i>Instances for the Limited Placement Problem (LPP)</i>	84
3.5.2.2	<i>Instances for the Unconstrained Placement Problem (UPP)</i>	84
3.5.2.3	<i>Computational results for the Placement Problem</i>	86
3.5.3	<i>Knapsack Problem</i>	88
3.5.4	<i>Cutting Stock Problem</i>	90
3.5.5	<i>Bin Packing Problem</i>	90
3.6	Contributions for the literature	92
3.7	Conclusions	95
3.8	The draw of the solutions	96
4	THE EXPLICIT ENUMERATION FOR IRREGULAR CUTTING STOCK PROBLEM UNDER DETERMINISTIC AND STOCHASTIC DEMAND	101
4.1	Introduction	101
4.2	Pattern generation	103
4.3	MIP models for a set of patterns	104
4.3.1	<i>Simple MIP model for given patterns in cutting stock problem</i>	105
4.3.2	<i>General MIP model for given patterns in cutting stock problem</i>	105
4.4	Two-stage MIP model for stochastic demand	107
4.5	The solution method: Explicit Enumeration of Patterns	108

4.6	Computational experiments	109
4.6.1	<i>Instances with proportional demand from Song and Bennell (2014)</i>	110
4.6.2	<i>Instances with deterministic demand from Valle et al. (2012)</i>	112
4.6.3	<i>Instances with stochastic demand</i>	113
4.6.3.1	<i>The expected value of perfect information</i>	113
4.6.3.2	<i>The value of stochastic solution</i>	114
4.6.3.3	<i>Numerical results</i>	115
4.7	Conclusion	117
5	IMPROVING NON-OVERLAP PARTITIONS FOR IRREGULAR STRIP PACKING MODELS	119
5.1	Introduction	119
5.2	Revisiting mixed integer programming models for strip packing prob- lems	120
5.3	New partition model	122
5.3.1	<i>Partitioning the solution space</i>	123
5.3.2	<i>Variable eliminations</i>	124
5.4	Symmetry breaking constraints	125
5.4.1	<i>Connecting pieces</i>	125
5.4.2	<i>Connecting pieces with the board</i>	126
5.4.3	<i>Pieces with same shape</i>	126
5.5	Results	127
5.6	Conclusions	131
6	MIXED-INTEGER LINEAR PROGRAMMING MODELS FOR THE IRREGULAR STRIP PACKING PROBLEM: ANALYZING THE USE OF CONTINUOUS MODELS IN SEMI-CONTINUOUS AND DIS- CRETE DOMAINS	133
6.1	Introduction	133
6.2	Continuous MIP models for nesting problems	136
6.3	Computational results	138
6.3.1	<i>Instances</i>	138
6.3.2	<i>Discrete domain</i>	139
6.3.3	<i>Semi-continuous domain</i>	141
6.4	Conclusions	143
7	CONCLUSION	145
7.0.1	<i>Future work</i>	147
	BIBLIOGRAPHY	149

APPENDIX A	DRAWING PATTERNS IN C&P PROBLEMS USING	
	$\text{\LaTeX}$ /TIKZ	159
A.1	Introduction	159
A.2	How to use $\text{\LaTeX}$ /TikZ	159
A.3	Basic components	160
A.4	Drawing primitive structures	161
A.5	Conclusions	162
A.6	Drawings and code	162

INTRODUCTION

Operational research is a scientific method for making decisions that searches a good decision to make given a space of feasible solutions. In everyday events, we can make good decisions using operational research or the experience when, for example: choosing the smallest path not to get late to work, making a family planning so you do not have financial problems or working out a balanced nutritional diet to improve your health. In the organizational environment the decisions have great importance in an economic and sustainable point of view. Generally, the operational research is divided into strategic decisions (taken at the higher level of management), tactical decisions (taken in the middle level of management) and operational decisions (linked to the control and operational activities of the organization). Any organization that do not care about the process of decision making, even empirically, will probably lose competitiveness and may leave the market. On the other hand, a private or public company that makes the best decisions are more likely to succeed.

In the path to find the best solutions for problems, operational research studies the decision making via mathematical programming, heuristic methods, exact methods, metaheuristic, matheuristics and others. In this thesis, we study methods in order to optimize problems in the class of combinatorial optimization problems. In this class of problems we can highlight the problems the salesman problem, the designation problem, the production planning problem, the logistics problem and the one studied in this thesis, the cutting and packing problem.

This chapter is organized as follows. A classification of cutting and packing problems is presented in Section 1.1. In Section 1.2, we present a description of the problems studied in this thesis and in Section 1.3, the objectives of this study with our main contributions. Finally, we present the organization of this thesis in Section 1.4.

1.1 Cutting and packing problems

Cutting and packing problems are largely studied in operational research. These problems are found mainly at lower levels of management of operational decisions, where it is wanted to determine the layout of small objects inside larger objects in order to optimize some objective function, such as reduce raw material waste, reduce operational time, maximize resource utilization (as containers and bins), minimize produced industrial waste and others.

These problems can be found in the one-dimensional version, as in the cutting of paper rolls, beams and tubes; in the two-dimensional version, as in the pallet packing and plates cutting; and in the three-dimensional version, where it is wanted to allocate boxes in trucks. In addition to these examples, there are many other applications, including viewing other problems, such as scheduling problems, as a cutting and packing problem.

Despite its easy definition, cutting and packing problems have been challenging researchers and organizations for several decades, as can be seen in [Gilmore and Gomory \(1961\)](#), [Art \(1966\)](#), [Albano and Sapuppo \(1980\)](#) and [Cunninghame-Green \(1989\)](#). This fact is due to the practical importance of the problem and its combinatorial nature that is NP-hard for the majority of cases (see [Fowler, Paterson and Tanimoto \(1981\)](#)).

In summary, cutting and packing problems can be defined by a list of small objects (items) and larger objects (bins). Small objects should be placed fully inside larger objects without overlap. The first typology, proposed by [Dyckhoff and Finke \(1992\)](#), classifies cutting and packing problems using four characteristics, to know: dimensionality (one-, two-, three- or N -dimensional), kind of assignment (all bins are used to allocate a set of items or a set of bins is selected to allocate all items), assortment of the larger objects (one bin, several identical bins or several different bins), assortment of small items (similar items, many items of different shapes, many items with a few different shapes or identical items).

The typology from [Dyckhoff and Finke \(1992\)](#) was very important for the classification of cutting and packing problems. [Wäscher, Haußner and Schumann \(2007\)](#) presented some inconsistencies and limitations of this typology and extended it. The next section presents a brief description of this improved classification and the main problems studied in this thesis.

1.1.1 Typology of cutting and packing problems

The typology of [Wäscher, Haußner and Schumann \(2007\)](#) uses five criteria to classify cutting and packing problems, to know:

- Dimensionality: the number of dimensions of the problem studied. The problems can be classified as one-dimensional, two-dimensional or three-dimensional. Differently from the classification proposed by [Dyckhoff and Finke \(1992\)](#), problems with more than three dimensions are classified as extensions of cutting and packing problems.

- Kind of assignment: the objective can be classified into output maximization or input minimization. In output maximization problems, bins are not sufficient to accommodate all items, while in input minimization problems all the items must be allocated, but it is wanted to allocate them at the lowest possible cost.
- Assortment of small objects (or items): the items may be identical, weakly heterogeneous (many items alike) or strongly heterogeneous (many different items).
- Assortment of larger objects (or bins): we can have one bin with all dimensions fixed, one bin with one or more open dimensions and several bins with fixed dimensions, which may be identical, weakly heterogeneous or strongly heterogeneous.
- Shape of small items: we can classify the two-dimensional and three-dimensional items into regular and irregular. Regular items are rectangles, circles, boxes, cylinders and balls. The authors define irregular items as non-regular items, and we understand these items as convex and non-convex polygons that may contain holes.

In this thesis, we are interested in problems with irregular items (called nesting problems). The basic cutting and packing problems classified by [Wäscher, Haußner and Schumann \(2007\)](#) are studied, to know: *placement problem*, *knapsack problem*, *open dimensional problem*, *cutting stock problem* and *bin packing problem*. Figure 1 presents these problems.

Figure 1 – Types of basic problems, extracted from [Wäscher, Haußner and Schumann \(2007\)](#).

We describe below the basic problems that were the focus of this doctoral thesis.

1. The **placement problem** considers a set of irregular items (weakly heterogeneous) that can be allocated inside a limited set of bins. In this case, the goal is to maximize the sum of the value of the items allocated in the bins.

2. The **knapsack problem** considers a strongly heterogeneous set of irregular items, each item with an associated value, which can be allocated in a bin with fixed dimensions. The objective is to maximize the value of the items allocated in the bin.
3. The **open dimensional problem** considers an arbitrary set of irregular items (identical, homogeneous or heterogeneous) that must be allocated in a bin with one or more open dimensions. In this problem, it is wanted to minimize the part of the bin used to allocate the items.
4. The **cutting stock problem** considers a set of weakly homogeneous irregular items that must be allocated in a set of bins. Each bin has an associated cost and the objective is to minimize the cost of the bins used.
5. The **bin packing problem** differs from the cutting stock problem by allocating a set strongly heterogeneous irregular items. These items must be allocated in a set of bins. Each bin has an associated cost and the objective is to minimize the cost of the bins used.

1.2 Geometry of irregular items and the best known resolution methods

Our focus is on two-dimensional problems, where it is wanted to allocate a set of irregular items, without overlap, inside one or more bins in order to optimize an objective function. In the problems studied, we consider irregular items (convex and non-convex polygons), limited bins and unlimited bins (with one or two open dimensions). These problems have a special difficulty, which is related to the geometry of the items. Due to the importance of this difficulty, we review the main representations used to deal with irregular items in Section 1.2.1. Then, in Section 1.2.2, we present a brief review of the best resolution methods in the literature for these problems.

1.2.1 Geometry of the nesting problems

When dealing with nesting problems, the geometry must receive special attention. Several techniques and approaches have been developed, including a tutorial to deal with this issue (see [Bennell and Oliveira \(2008\)](#)). The main approaches to represent the geometry of the problem described in [Bennell and Oliveira \(2008\)](#) are: the raster method, the direct trigonometry, the D-function, the no-fit polygon and the phi-function.

In the raster method (or pixel method), the objects (items and bins) are represented via matrices. The objects are divided into areas, represented by a matrix, and an encoding scheme is proposed to avoid overlap between items. One of the simplest encoding schemes, proposed by [Oliveira and Ferreira \(1993\)](#), defines each element of the matrix as 1, if there is an item in the area represented by this element; and 0, otherwise. With this encoding, shown in Figure 2, to ensure

that the items do not overlap, it is sufficient to verify if the sum of the matrices representing the items is less than or equal to 1. Different encoding strategies were proposed by [Segenreich and Braga \(1986\)](#) and [Babu and Babu \(2001\)](#).

Figure 2 – Representation extracted from [Oliveira and Ferreira \(1993\)](#).

The use of the raster method can, in many cases, be applied successfully when items and bins are tetris shapes, that is, it is possible represent the real form of the objects by an encoding strategy. This approach is not interesting for large items and rather irregular items with many edges specifically non-orthogonal ones. These features require the usage of large dimension matrices to refine the objects approximation, which increases the computational cost.

Another technique to verify the relationship between two items is the use of mathematical equations known as phi-functions, that report the relative position between a pair of items. In general, the value of the phi-function is zero if the items are touching; greater than zero, when items are separated; and smaller than zero, when the items are overlapping. We can exemplify the concept of a phi-function with two circles, i and j , which are centered at points (x_i, y_i) and (x_j, y_j) and radii equal to r_i and r_j , respectively. The mathematical equation that verifies the overlap (or phi-function) in this case is given by the difference between the Euclidian distance between the center of the circles and the sum of their radii, i. e., $\phi_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - (r_i + r_j)$. This example is illustrated in Figure 3.

Figure 3 – Example of phi-function of circles i and j .

We can also write phi-functions to check the overlap between of primary objects such as convex polygons and circles (see [Stoyan et al. \(2001\)](#)) or complex objects represented by the union of primary objects (see [Stoyan et al. \(2004\)](#)).

The D-function is a mathematical equation that returns whether a point is to the right to the left or over a line segment. For example, 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 is defined by equation: $D_{abr} = (a_x - b_x)(a_y - r_y) - (a_y - b_y)(a_x - r_x)$. Figure 4 present the different values that the D-function (D_{abr}) can have

and its implications. Notice that there are three situations, to know: 1) if $D_{abr} = 0$ the point r is over the line defined by $\overline{ab}$; 2) if $D_{abr} > 0$ the point r is on the right side of line $\overline{ab}$; and 3) if $D_{abr} < 0$ the point r is on the left side of line $\overline{ab}$.

Figure 4 – Values of D-function and its implications with the classical coordinate system, i.e. x-coordinates grow to the right and y-coordinates grow upperwards.

The polygonal representation for items and bins is widely used. In this representation the objects are represented by polygons, usually represented by a list of vertices. In this case, the overlap is calculated using direct trigonometry. An efficient verification of overlap between two items, presented by [Bennell and Oliveira \(2008\)](#), is based in four tests. The first test checks the overlap of the smaller rectangles that enclose the items: if these surrounding rectangles are separated, there is no overlap between the items and we can finalize the tests. The second test checks the overlap between the rectangular envelopes of all the pairs of edges of the items: if there is no overlap between the envelopes edges, the items do not overlap and so we can finalize the tests. In the third test we check if there are intersections between the edges of the items: if there are, the items are overlapping and, if not, the fourth test is performed to verify if the items are overlapping or separated. The fourth test verifies if one of the items is completely contained in the other item. For this, we verify if the vertices of one item are all inside the other item. If the fourth test ensures that no item are inscribed in the other, the items are separated. Otherwise, the items are overlapping. Figure 5 shows the four tests. The first three tests are illustrated in Figures **a**, **b** and **c**. The fourth test is presented in Figure 5 **d** (items with overlap) and Figure 5 **e** (items without overlap).

Figure 5 – Example of the relative position of two items via direct trigonometry, extracted from [Oliveira and Ferreira \(1993\)](#).

The no-fit polygon is a geometric construction from two polygons, with oriented edges, which facilitates the overlap verification. Consider any two polygons, one of them fixed in the plane and the other orbital, with a reference point. We can calculate a priori the region where the

reference point of the orbital item can not be allocated without overlapping the fixed polygon. There are several ways to calculate this region. For convex polygons there is an efficient method, proposed by [Cunningham-Green \(1989\)](#), than uses Minkowski sum. Given two polygons, A and B , the no-fit polygon is given by the sum of the polygon A and $-B$. [Cunningham-Green \(1989\)](#) consider A as the fixed polygon in plane with counterclockwise orientation and B as the orbital polygon with a reference point and clockwise orientation. All edges, formed by two consecutive vertices of the polygons A and B , are shifted to a point, and then these edges are concatenated in clockwise orientation, following the increasing order of edge slope. Figure 6 presents the construction of the no-fit polygon.

Figure 6 – The no-fit polygon for convex polygons, extracted from [Burke et al. \(2007\)](#).

The difficulty increases considerably when dealing with non-convex polygons. Therefore, other methods have been proposed in recent years: [Hu-yao and Yuan-jun \(2006\)](#), [Burke et al. \(2007\)](#), [Bennell and Song \(2008\)](#) and [Burke et al. \(2010\)](#).

As far as we know, the concept of no-fit polygon on a mesh of points has been presented in the last couple of years in the study from [Toledo et al. \(2013\)](#). In this thesis, we study the use of a mesh of points on the non overlap region between two or more items. Extending the proposed idea from [Toledo et al. \(2013\)](#) to heuristic methods and non-polygonal items. This concept is called **no-fit raster** (see the no-fit raster in detail in Chapters 2 and 3).

An alternative representation, still not used very often, is the approximation of items by circles and ellipses. With this representation, the overlap between items is calculated over the circles and ellipses that represent these items. Figure 7 shows the representation used by [Jones \(2014\)](#). The author proposes an exact method that solves the problem avoiding the overlap between circles, being able to include more circles if there is overlap between items.

Figure 7 – Representation of polygons by circles, extracted from [Jones \(2014\)](#).

1.2.2 Main resolution methods for nesting problems

This section presents a brief review of resolution methods for nesting problems that inspired this thesis. It is important to highlight that all the problems studied here are NP-Hard, that is, assuming that $P \neq NP$, we can say that there are no exact algorithms that use polynomial time to solve these problems. Therefore, most of the papers propose heuristic approaches to solve them. [Bennell and Oliveira \(2009\)](#) say that the evolution of heuristic approaches for irregular cutting and packing problems is related to the improvement of computers, *softwares* and *hardwares*, and the advancement of geometric techniques used to verify overlap. Here we present a review of the main approaches used to solve nesting problems.

The most popular heuristics in the nesting problem area use allocation rules about the sequence of items (also found in the literature as an allocation policy), where the partial solutions are constructed item by item, i.e., allocated one item at a time to build a final solution. Heuristics with the bottom left corner policy (or bottom-left heuristics), initially proposed by [Baker, Jr and Rivest \(1980\)](#), are widely used. In these heuristics, after determining a sequence to insert the items, one item is allocated at a time in the leftmost and bottommost possible position.

According to [Dowsland, Dowsland and Bennell \(1998\)](#), the bottom-left heuristics can be summarized in two iterative steps. Consider that an item is allocated in the upper right corner of the bin. Then, the steps are: (1) move the item vertically downward until it touches either another item or an edge of the bin; (2) move the item to the left until it touches either another item or an edge of the bin. These steps are repeated until no more movement is possible. Figure 8 (a) presents an example of this strategy for items 1 through 5. [Dowsland, Dowsland and Bennell \(1998\)](#) highlight that heuristics that do not consider empty regions (or holes) to place items inside of the layout can lead to solutions with very low-density. The authors propose six heuristics, based on the bottom-left corner rule, where three of them do not consider holes and the other three fill the holes. Figure 8 (b) presents a solution obtained by a strategy that can fill holes. Other methods that exploit the bottom-left heuristics can be found in [Dowsland, Vaid and Dowsland \(2002\)](#), [Gomes and Oliveira \(2002\)](#) and [Burke et al. \(2006\)](#).

Figure 8 – Bottom-left heuristics: (a) without allocating items in holes, (b) allocating items in holes. Extracted from [Dowsland, Vaid and Dowsland \(2002\)](#)

[Oliveira, Gomes and Ferreira \(2000\)](#) present a constructive method for the irregular strip

packing problem, with several allocation criteria and ways to evaluate the solution. This method, called *Técnicas de Optimização para o Posicionamento de Formas Irregulares* (TOPOS), starts by allocating an item and then always adding new items in partial layouts, i.e., with one or more items allocated. To define the position of a new item, TOPOS considers different allocation criteria: the difference between the area of the rectangular envelope of the partial solution (considering the new item placed) and the area of all items already allocated (Figure 9 (a)); the overlap between the rectangular envelope of the new item and the rectangular envelope of each item present in the partial solution (Figure 9 (b)); and the Euclidean distance between the center of the new item's rectangular envelope and the center of the rectangular envelope of the partial solution (Figure 9 (c)).

Figure 9 – Evaluation criteria for layouts. Extracted from [Oliveira, Gomes and Ferreira \(2000\)](#).

Because TOPOS is a constructive heuristic and depends on the sequence in which items are allocated in the layout, seven items ordering criteria were proposed by [Oliveira, Gomes and Ferreira \(2000\)](#), to know: (a) decreasing length; (b) decreasing area; (c) decreasing concavity (difference between the area of the convex envelope of the item and the area of the item); (d) increasing rectangularity (difference between the area of the rectangle envelope of the item and the area of the item); (e) decreasing total area (area of the item multiplied by its demand or the number of copies to be inserted); and (f) random. The criteria from (a) to (e) are presented in Figure 10. Recently, [Bennell and Song \(2010\)](#) proposed other heuristics and criteria based on TOPOS.

Figure 10 – Criteria for ordering items. Extracted from [Oliveira, Gomes and Ferreira \(2000\)](#).

The TOPOS method improved the results for irregular strip packing problem and some sorting strategies and criteria are widely used until today. [Oliveira, Gomes and Ferreira \(2000\)](#) is one of the most cited paper on heuristics for irregular cutting problems of cut items in the literature, with 116 citations in Google Scholar.

In contrast with approaches based on partial solutions, complete solutions approach deal with complete solutions usually obtained by constructive methods. [Jakobs \(1996\)](#) is one of the first papers to present a meta-heuristic for cutting irregular items. The author applies a genetic algorithm to the problem, representing each individual by a vector of items. The individual (or chromosome) represents the sequence in which the items will be allocated by a bottom-left heuristic. The difference from their approach to partial solutions approaches is that the genetic algorithm uses information from two complete sets (two chromosomes) to cross and obtain a new complete layout.

[Gomes and Oliveira \(2002\)](#) propose a new bottom-left heuristic that considers holes. The authors use the sorting criteria of TOPOS to obtain complete solutions and then exchange two items in the sequence and apply the new bottom-left heuristic. Figure 11 presents the three different levels of exchanges that can occur in the sequence. $\Delta = 1$ exchange adjacent items in the sequence, while $\Delta = 2$ and $\Delta = 3$ exchange items at a distance of two and three items in the sequence, respectively.

Figure 11 – Neighborhood of the sequence proposed by [Gomes and Oliveira \(2002\)](#).

Constructive heuristics with searches in the sequence of items obtained the best results in the literature up to the middle of 2006. In this year, two papers appeared refashioning the area of nesting problems, specifically in irregular strip packing problem: [Gomes and Oliveira \(2006\)](#) and [Egeblad, Nielsen and Odgaard \(2007\)](#). In these papers, the authors studied the region of infeasibility of the problems, i.e., considered the search for solutions on regions where there is overlap, which allowed them to better explore the solution space of the problem. Based on this, they obtained better results than the previous papers from the literature.

In [Gomes and Oliveira \(2006\)](#), the authors presented a simulated annealing meta-heuristic with a mixed integer programming model for the irregular strip packing problem. The meta-heuristic uses a compaction model to compact the solution, and separation model to remove the overlap between items. This model is considered the first mixed integer programming model for nesting problems. [Egeblad, Nielsen and Odgaard \(2007\)](#) proposed a method of resolution for two-dimensional and three-dimensional nesting problems based on the guided local search heuristic. Given an initial solution that may contain overlap, the proposed method consists of

minimize overlap by making horizontal and vertical searches within the bin. This local search is shown in Figure 12.

Figure 12 – Example of horizontal (a) and vertical (b) search, to obtain a new solution (c). Extracted from Egeblad, Nielsen and Odgaard (2007).

Other successful heuristics that search for solutions visiting infeasibility layouts, minimizing the overlap between items, are: the iterated local search from Imamichi, Yagiura and Nagamochi (2009), the guided local search from Umetani *et al.* (2009), the extended local search algorithm from Leung, Lin and Zhang (2012) and the cuckoo search meta-heuristic using the guided local search proposed by Elkeran (2013), that is the solving method capable of finding the best results in the literature so far. Elkeran (2013) improved the results for twelve out of fifteen classical instances and obtained the same result for the others. In the paper, the author finds a feasible and compact solution and then reduces the length of the layout, generating some overlap. After compacting the length of the layout, the method seeks to minimize the overlap between the items and, if it finds a feasible solution, reduces the length of the new layout again. If it does not find a feasible solution, it increases the length of the layout to find a feasible solution. This is the main idea of cuckoo search meta-heuristic that is illustrated in Figure 13. Another interesting idea presented by the author is the grouping of pairs of items (or pairwise clustering), reducing the complexity of the problem.

Papers that deal with irregular bins, i.e., convex and non-convex bins that may contain holes and regions of quality, as in the approaches presented so far in this review, basically propose heuristic approaches (see 14). Heistermann and Lengauer (1995) presented one of the first studies to deal with cutting of leather parts. The authors proposed a constructive algorithm, which has been used in the industrial environment since June 1992. Recent studies developed for the cutting leather industry are: Yuping, Shouwei and Chunli (2005), Crispin *et al.* (2005), Alves *et al.* (2012) and Baldacci *et al.* (2014).

Yuping, Shouwei and Chunli (2005) presented a simulated annealing algorithm for the leather nesting problem, where the bins are leather. The authors used severe penalties in the

Figure 13 – Vision of the cuckoo search algorithm proposed by [Elkeran \(2013\)](#).

Figure 14 – Example of irregular bin from the leather cutting industry presented by [Alves et al. \(2012\)](#).

objective function to find feasible positions in the layout. Rotation and translation strategies are proposed to avoid overlap and ensure that items are placed inside the leather pieces. [Crispin et al. \(2005\)](#) proposed a genetic algorithm that uses the no-fit polygon to check the nonoverlap between items. [Alves et al. \(2012\)](#) presented several constructive heuristics for leather cutting, based on the choice of interesting regions of the bin and on the selection of items to be inserted in the layout using the no-fit polygon. Finally, [Baldacci et al. \(2014\)](#) formulate models for the knapsack problem and use it to solve problems with multiple bins and quality regions using a matrix representation for items and bins. To solve these models the authors presented an iterative heuristic using Lagrangian relaxation.

Considering exact methods and models for irregular strip packing problem that have been proposed, the items can be placed into: discrete coordinates, semi-continuous coordinates and continuous coordinates. For methods that use discrete coordinates (Figure 15 (a)), the items can be placed in a finite set of points of the bin. For methods that use the semi-continuous coordinates (Figure 15 (b)), the items can be placed in a finite set of lines of the bin. For methods

that use continuous coordinates to allocate the items, the items can be placed anywhere in the bin (Figure 15 (c)).

Figure 15 – The blue regions represent the positions where the item can be allocated in (a) discrete methods, (b) semi-continuous methods and (c) continuous methods.

Models with the optimum solution restricted to a mesh of points or discrete coordinates (as shown in Figure 15 (a)) were proposed by Carravilla, Ribeiro and Oliveira (2003) (constraint programming model) and by Toledo *et al.* (2013) (mixed integer linear programming model). Rodrigues and Toledo (2016) proposed a clique covering model that reduces the number of constraints of the dotted-board model from Toledo *et al.* (2013), improving its linear relaxation. Recently, Cherri, Carravilla and Toledo (2016) present a matheuristics for irregular strip packing problem based on the dotted-board model from Toledo *et al.* (2013).

Fischetti and Luzzi (2009) present a mixed integer linear programming model that is not conditioned to a mesh of points, i.e., the items may be allocated in all positions where they are completely contained in the bin (the continuous coordinates Figure 15 (b)). Alvarez-Valdes, Martinez and Tamarit (2013) extend the compaction model of Gomes and Oliveira (2006) and propose two reformulations for the model from Fischetti and Luzzi (2009). The authors also propose a branch-and-bound method for the problem. These method and all these models verify the overlap between items using the concept of no-fit polygon. All mixed integer linear programming models were solve to optimality only for small instances with up to 16 items in the continuous domain (instance DIGHE1OK from Alvarez-Valdes, Martinez and Tamarit (2013)) and even 56 items to a mesh of points (instance BLAZP2P4_28_28 from Toledo *et al.* (2013)).

Leao *et al.* (2016) attempt to combine the advantages of the discrete model presented in Toledo *et al.* (2013) and the continuous models (in Fischetti and Luzzi (2009) and Alvarez-Valdes, Martinez and Tamarit (2013)) proposing the first semi-continuous mixed-integer linear programming model for irregular strip packing problem. The semi-continuous model from Leao *et al.* (2016) solve was to optimality for small instances with up to 10 items (as J1-10-20-0, J1-10-20-1, J1-10-20-2, J1-10-20-3 and J1-10-20-4). The semi-continuous model in comparison with the continuous model from Alvarez-Valdes, Martinez and Tamarit (2013) in terms of solution value is 12 times worse, 10 times better and 16 times equal, while in comparison with the discrete model from Toledo *et al.* (2013) is 11 times worse, 43 times better and 18 times equal.

After this overview of the area of irregular items, in the following sections we present a brief overview of the main contributions of this thesis and of each of the next chapters.

1.3 Main contributions to the nesting problems literature

This thesis investigates five basic irregular cutting and packing problems presented in the Section 1.1.1, to know: placement problem, knapsack problem, open dimensional problem, cutting stock problem and bin packing problem. The main contributions are:

1. We are the first to use of no-fit raster in heuristics, for polygonal and non-polygonal items.
2. We propose a generic heuristic, competitive with the literature, for all five basic problems. This heuristic improved 56% of benchmark instances.
3. We present the first constructive heuristic for the cutting stock problem, improving the results obtained for more than 58% of benchmark instances.
4. We propose a hybrid method to explicitly generate patterns for the irregular cutting stock problem than can be more efficient than column generation - improving more than 62% of the literature benchmark.
5. We propose the first method to deal with uncertain demand for irregular items.
6. We develop the first method in the literature for packing irregular items into bins with two open dimensions, presenting solutions with up to 99 items. And we propose, for the first time in the literature, a model for the open dimension problem with two open dimensions.
7. We improve the non overlap partitions for the best known mathematical model for the irregular strip packing problem.
8. We propose an adaptation for two mathematical models for the continuous irregular strip packing problem for discrete and semi-continuous domain. These adaptation presented better or equal results, to when compared to the specific models, for 96.77% of the benchmark instances.

1.4 Thesis organization

In this section we present the organization of this thesis. Each chapter contains all the information the reader needs, therefore it can be read in the preferred sequence. To facilitate the reading, we summarize each chapter in sections 1.4.1 to 1.4.6. The heuristic methods are present in Chapters 2 to 4 and mathematical models in Chapters 5 and 6. The conclusions are presents in Chapter 7. For the reader who wants to draw the items and bins as presented in this document, we have a tutorial for it using L^AT_EX in Appendix A.

1.4.1 Chapter 2

Title: “A biased random key genetic algorithm for open dimension nesting problems using no-fit raster”.

Chapter 2 presents a study about the problems with one or two open dimensions and the list of items is composed of circles, polygons and non-polygonal items. In this chapter we present a biased random key genetic algorithm to determine an allocation sequence for items via two heuristics: the bottom-left heuristic (to the problem with one open dimension) and the rectangle bottom-left heuristic (to the problem with two open dimensions).

The main contributions of this chapter are:

- The first study to use the biased random key genetic algorithm to solve nesting problems.
- The proposal of a heuristic for the problem with two open dimensions.
- The proposal of the first mathematical model to the problem with two open dimensions.
- The adaptation of a set of instances for the irregular strip packing problem to the problem with two open dimensions, opening new areas of research and serving as a reference for future studies.

This chapter is an article written in collaboration with Marina Andretta (ICMC-USP) and Thiago Alves de Queiroz (DM-UFG) that was published in the international journal *Expert Systems With Applications* [Mundim, Andretta and Queiroz \(2017\)](#). An adaptation of this study for problems with fixed dimensions was published as a book chapter in [Mundim, Andretta and Queiroz \(2015\)](#).

1.4.2 Chapter 3

Title: “A general heuristic for two-dimensional nesting problems with limited-size containers”.

Chapter 3 proposes a generic heuristic for problems with fixed dimensions rectangular and irregular bins. The proposed heuristic manages a set of positioning rules and improves a large set of literature results for the placement problem, knapsack problem, cutting stock problem and bin packing problem.

The main contributions of this chapter are:

- A succinct review of all problems with fixed dimensions.

- An effective way to calculate the inner-fit raster for irregular bins.
- A large set of computational experiments, with 1109 instances from the literature, presented in an organized and structured way.
- The improvement of solutions for 39 out of 70 instances of different problems.

A preliminary version of the chapter in an abstract format was presented in the ESICUP (see [Mundim et al. \(2016\)](#)). This chapter is an article written in collaboration with Marina Andretta (ICMC-USP), Maria A. Carravilla (FEUP/INESC TEC) and José F. Oliveira (FEUP/INESC TEC). This article will be submitted to an international journal.

1.4.3 Chapter 4

Title: “*When explicit enumeration surpasses column generation: the case of the irregular cutting stock problem under deterministic and stochastic demand*”.

Chapter 4 proposes a hybrid method for the irregular cutting stock problem with both deterministic and stochastic demand. The method proposed consists of two steps: 1) the generation of a large number of patterns; 2) finding the best solution within the generated patterns. The method proposed for irregular cutting stock problem with known demand finds the best solution with Gilmore and Gomory model ([Gilmore and Gomory \(1961\)](#) and [Gilmore and Gomory \(1963\)](#)), while the method proposed for irregular cutting stock problem with stochastic demand finds the best solution with a new two-stage programming model and use a Gilmore and Gomory model adaptation.

The main contributions of this chapter are:

- The explicit enumeration method, for the irregular cutting stock problem, obtained better solutions than the column generation method.
- The proposed method improves the results for more than 62% instances of the literature for the irregular cutting stock problem with known demand.
- The proposal of a two-stage programming model for irregular cutting stock problem with stochastic item demand.
- The proposal of the first set of instances for the irregular cutting stock problem with stochastic demand.

This article was written in collaboration with Marina Andretta (ICMC-USP). Some ideas about the irregular cutting stock problem with deterministic demand have contribution of Maria

A. Carravilla (FEUP/INESC TEC) and Jose F. Oliveira (FEUP/INESC TEC). This article will be submitted to an international journal.

1.4.4 Chapter 5

Title: “*Improving non-overlap partitions for irregular strip packing models*”.

The results presented in Chapter 5 improved non overlapping partitions for open dimensional problem models proposed in previous studies. These studies are not present here to in order to shorten the thesis. But if the reader is interested in these studies, we published them in a full paper at *European Journal of Operational Research* in collaboration with Luiz H. Cherri (ICMC-USP), Marina Andretta (ICMC-USP), Franklina M. B. Toledo (ICMC-USP), Maria A. Carravilla (FEUP/INESC TEC) and José F. Oliveira (FEUP/INESC TEC) (see [Cherri et al. \(2016\)](#)). A preliminary version of the [Cherri et al. \(2016\)](#) which considers only convex items and uses direct trigonometry was present in XLVII SBPO - *Simpósio Brasileiro de Pesquisa Operacional*, at the City of Porto de Galinhas - PE, in August 2015 (see [Mundim et al. \(2015\)](#)).

The content of this chapter presents the improvements in these previous studies, in terms of the non overlapping partitions. Chapter 5 presents a mixed integer linear programming models for the irregular strip packing problem.

The results obtained by the adapted model outperforms the literature in terms of lower bounds, running-time and number of nodes explored to prove optimality.

The main contributions of this chapter are:

- Combination of the best characteristics of the best literature models present in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and [Cherri et al. \(2016\)](#).
- The proposal of a new symmetry breaking constraints into a new mixed integer model for the irregular strip packing problem.
- Improvement of lower bounds presented in the previous studies.
- Reduction of running-time presented in the previous studies. In additional, reduction of the number of nodes explored to prove optimality of the previous models.

This chapter is an article written in collaboration with Luiz H. Cherri (ICMC-USP). A initial study for convex polygon was presented in the Applied mathematical programming and Modelling, at the City of Brno - Czech Republic, in June 2016 (see [Rodrigues, Cherri and Mundim \(2016\)](#)), wich was written in collaboration with Marcos O. Rodrigues (ICMC-USP) and Luiz H. Cherri (ICMC-USP). This article will be submitted to an international journal.

1.4.5 Chapter 6

Title: “MIP models for the irregular strip packing problem: analyzing the of using continuous models in semi-continuous and discrete domains”.

In Chapter 6 we propose a change of domain for two mixed integer programming models for the irregular strip packing problem. We adapted HS2 from [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and NFP-CM from [Cherri et al. \(2016\)](#) so the items can be placed either in discrete or in semi-continuous positions of the bin.

The main contributions of this chapter are:

- Comparative analysis between two known models.
- The domain changes were important for two reasons: 1) this adaptation solved the problem faster and 2) these adaptations solve specific problems of several industries, as for example situations where the material to be cut is striped or plaid.
- The new models proved for the first time the optimality of 22.58% of benchmark of the literature.
- Competitive results with the specialized models, with better or equal results to the specialized models for 96.77% of the instances.

This chapter is an article written in collaboration with Marina Andretta (ICMC-USP) and Luiz H. Cherri (ICMC-USP). This article will be submitted to an international journal.

1.4.6 Chapter 7

Chapter 7 presents the conclusions of this thesis, where we synthesize the last considerations and present suggestions for future studies.

A BIASED RANDOM KEY GENETIC ALGORITHM FOR OPEN DIMENSION NESTING PROBLEMS USING NO-FIT RASTER

We consider two NP-hard open dimension nesting problems for which a set of items has to be packed without overlapping into a two-dimensional bin in order to minimize one or both dimensions of this bin. These problems are faced by real-life applications, such as textile, footwear and automotive industries. Therefore, there is a need for specialized systems to help in a decision making process. Bearing this in mind, we derive new concepts as the no-fit raster, which can be used to check overlapping between any two-dimensional generic-shaped items. We also use a biased random key genetic algorithm to determine the sequence in which items are packed. Once the sequence of items is determined, we propose two heuristics based on bottom-left moves and the no-fit raster concept, which are in turn used to arrange these items into the given bin observing the objective criteria. As far as we know, the problem with two-open dimensions is being solved for the first time in the context of nesting problems and we present the first whole quadratic model for these problems. Computational experiments conducted on benchmark instances from the literature (some from the textile industry and others including circles, convex, and non-convex polygons) show the competitiveness of the approaches developed as they were able to calculate the best results for 74.14% of the instances. It can be observed that these results show new directions in terms of solving nesting problems whereby approaches can be coupled in existing intelligent systems to support decision makers in this field.

2.1 Introduction

Nesting problems are related with packing and cutting problems in which items and bins are not necessarily regular-shaped pieces, such as rectangles, but instead they can be any convex or non-convex polygons. Problems of this nature are frequently found in manufacturing and transformation industries, such as the footwear [Yang and Lin \(2009\)](#), automotive [Alves et al. \(2012\)](#), leather garment and furniture [Baldacci et al. \(2014\)](#) industries. Furthermore, they aim to minimize total waste after the packing or cutting process occurs. It is common to use the term “cutting” for industries as a large bin is frequently cut by a laser machine in order to obtain small items, although from a theoretical point of view the terms “cutting” and “packing” refer to the same type of problem and, therefore can be used interchangeably [Queiroz et al. \(2012\)](#).

Most of these laser machines are operated by a decision maker who needs to figure out how to arrange items over the bin without overlapping and avoiding waste areas as much as possible. Of course, this is a hard task as an exponential number of combinations exists when arranging the items over an area. In other words, the decision maker is dealing with an NP-hard problem that requires specialized approaches to arrange the items and minimize the waste generated [Bennell and Oliveira \(2009\)](#). Naturally, these approaches can substitute the ones existing in the machines when presenting better results and then support the decision maker conveniently, or even machines with intelligent systems that in turn require efficient solution procedures to learn about the process and make the right decision whenever possible.

Taking this into account, we solve two nesting problems, namely the two-dimensional versions of the *one-open dimension problem* and *two-open dimension problem*. In the first problem (one-open dimension), the rectangular bin has one dimension with a fixed value, while the other is open and must be minimized. For the second problem (two-open dimensions), both dimensions of the rectangular bin are open and have to be minimized. Therefore, there is a set of items, which can rotate according to predefined angles, defined as convex or non-convex polygons (with a set of Cartesian points) or circles (with a given diameter) that must be packed (cut) in the bin without overlapping. In other words, both problems attempt to pack all items in the bin so that the sizes of the open dimensions of the bin are minimized. It is worth mentioning that both problems are NP-hard [Garey and Johnson \(1979\)](#) and, to the best of our knowledge, the two-open dimension problem is solved for the first time when items are irregular shaped pieces, although this problem has a real application in Very Large Scale Integration (VLSI) design, timber cutting, the textile industry and layout of newspaper [Chen et al. \(2010\)](#).

As items can be irregular shaped pieces, the major challenge is to handle the geometry to avoid overlapping between items. Another challenge is the combinatorial feature of the problem, which depends on the number of items. These challenges reflect the few studies proposing exact approaches to solve nesting problems, which directly affect the size of the instances which can be solved by optimality. [Fischetti and Luzzi \(2009\)](#) presented the first mixed-integer linear programming model in this field. [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) reformulated

the model proposed by [Fischetti and Luzzi \(2009\)](#) and extended the linear programming model from [Gomes and Oliveira \(2006\)](#) to propose a branch-and-bound algorithm for the 2D one-open dimension nesting problem, furthermore the authors. Furthermore, [Toledo et al. \(2013\)](#) presented a mixed-integer linear programming model that works over a grid of points and then could solve to optimality the largest instances to optimality found in the literature. In paper [Cherri et al. \(2016\)](#) we presented two mixed-integer linear programming models based on the decomposition of convex polygons: Direct Trigonometry Model (DTM), in which the non-overlapping constraints are based on usage of direct trigonometry; and the No-Fit Polygon Covering Model (NFP-CM), in which the non-overlapping constraints are based on usage of no-fit polygons.

One alternative to exact approaches is heuristics. Although heuristics cannot guarantee optimality, nor given good quality solutions (with a reasonable occupancy ratio) in some situations, they are suitable for real-industry applications as they can return solutions requiring very low computation time and can handle efficiently instances efficiently that involve many items and are of very different forms. Most of the literature concerning nesting problems deals with heuristics, which vary in nature (based on evolutionary principles, biological systems, sorting criteria, etc.), complexity and the number of moves, type of compacting strategy, type of positioning strategy, among other aspects (see [Bennell and Oliveira \(2009\)](#)). In the list of simple but effective heuristics are the ones based on the bottom-left concept, in which items are arranged iteratively in the left-most bottom position available in the bin.

[Bennell and Oliveira \(2009\)](#) discussed the good results that bottom-left heuristics can generate for nesting problems. [Dowsland, Vaid and Dowsland \(2002\)](#), [Gomes and Oliveira \(2002\)](#) and [Burke et al. \(2006\)](#) proposed bottom-left based heuristics, which can improve previous results published in the literature.

One advantage of bottom-left heuristics is their capacity of being used as a subroutine of more sophisticated approaches. In the case of nesting problems, [Jakobs \(1996\)](#) presented a heuristic based on a genetic algorithm, which considers a bottom-left heuristic to arrange items as a subroutine. [Oliveira, Gomes and Ferreira \(2000\)](#) presented various constructive heuristics in which some of them use bottom-left heuristics to solve the 2D one-open dimension problem.

Although bottom-left based heuristics can return good solutions very quickly, they cannot overcome approaches based on metaheuristics. For nesting problems, we can highlight the iterated local search proposed by [Imamichi, Yagiura and Nagamochi \(2009\)](#), the guided local search by [Umetani et al. \(2009\)](#) and the cuckoo search from [Elkeran \(2013\)](#). The best metaheuristic so far in terms of results for the one-open dimension problem is the cuckoo search proposed by [Elkeran \(2013\)](#), which includes a guided a local search to escape from local optima solutions, usage of a linear programming model to compact the layout and separation strategies to remove potential overlapping between items. Recently, [Pinheiro, Júnior and Saraiva \(2015\)](#) presented the first random-key genetic algorithm for nesting problems, particularly for the one-open dimension problem. This heuristic uses three rules for positioning the items that are based

on constructive procedures, in which chromosomes are responsible for determining the sequence and rotation to arrange items.

The literature about nesting problems with two-open dimensions is very limited as the instances solved involve only rectangular items or have very few irregular shaped items. Birgin and Sobral (2008) proposed non-linear models to find the smallest circle, triangle, square and rectangle capable of allocating a given set of circles. The heuristic proposed by Stoyan and Patsuk (2000) uses the phi-function concept and it is restricted to pack only one type of item. Costa, Gomes and Oliveira (2009) proposed a heuristic limited to solve instances with only one type of item. Kallrath (2009) presented an exact non-linear model to find the smallest square for packing circles and convex polygons with free rotations. Valle *et al.* (2012) and Martello and Monaci (2015) presented heuristics to compute rectangles with a high occupation area aiming to pack irregular shaped items. Results involving pairwise clustering of irregular shaped items were obtained from Adamowicz and Albano (1976), who proposed an extended search combined with the no-fit polygon to find the smallest rectangular envelope. Furthermore, Bennell *et al.* (2015) proposed a non-linear model that uses the phi-function concept to cluster pairs of irregular shaped items in the smallest circle, triangle or convex hull. Sato, Martins and Tsuzuki (2015) also proposed a pairwise placement heuristic, but considered the one-open dimension problem.

This study contributes to the literature of nesting problems and helps researchers, practitioners, decision makers, developers of expert systems, and anyone else who requires efficient approaches to support and guide the process of packing or cutting two-dimensional irregular shaped items into bins minimizing the total waste. The main contributions are listed below:

- For the first time, the two-open dimension problem with irregular shaped pieces is solved and has a quadratic integer model;
- Concepts based on the no-fit raster and inner-fit raster are presented to check overlapping between any type of irregular shaped items and to search for feasible positions for packing;
- Approaches based on a combination of a Biased Random-Key Genetic Algorithm (BRKGA), which is used to define the sequence and rotation in which each item is packed, are developed. They use bottom-left heuristics, which are used for positioning items into the bin, are developed.
- Instances consisting of convex and non-convex polygons and circles, with up to 99 items, are solved and the results are better than the ones from the literature for 74.14% of the instances.
- Results of medium and large-sized instances proposed for the two-open dimension problem are presented and made available for researches and practitioners to compare. Naturally, this problem is “harder” than the one-open dimension problem due to the two open dimensions, which increase the number of possibilities for arranging items significantly.

This chapter is organized as follows. The next section describes concepts used to deal with the computational geometry of irregular shaped items. Section 2.3 presents integer formulations, one with a quadratic objective function, for both problems studied here. Although these formulations can solve the problem to optimality, they are not suitable for large-sized instances due to the need of large amount of memory and computation time to solve each instance. Section 2.4 presents the BRKGA and its operations describing what each chromosome represents and its transformation in a solution of the nesting problems by using the bottom-left heuristics, the way that crossover operation is performed and the fitness function used. Section 2.5 shows the computational results for distinct sets of instances, as well as a comparison with other approaches from the literature. Conclusions and suggestions for an expert system are given in Section 2.5. The description of some instances with circles are presented in 2.7.

2.2 Computational geometry

Dealing with irregular shaped items at a low computational time requires using an appropriate strategy to ensure non-overlapping between items. For this task, some strategies have been developed in the literature. Bennell and Oliveira (2008) presented a tutorial with the most used techniques, which are: raster method (or pixel method), direct trigonometry, no-fit polygon and phi-functions. The raster method represents items and bins using a matrix and the detection of overlapping is performed over this matrix. Direct trigonometry uses mathematical operations to measure the distance between items and bins in order to detect overlapping. The no-fit polygon creates polygons that indicate where to position items without overlapping. The phi-function verifies overlap between items using mathematical equations defined over their dimensions. We can also highlight the studies carried out by Jones (2014) and Rocha *et al.* (2015) in which overlapping is detected using circles placed inside the items.

The approach presented in this chapter can verify overlapping between any type of two-dimensional items (regardless of their shape). To the best of our knowledge, this is the first strategy in the nesting literature which can this task without involving hard non-linear equations as in the phi-function and direct trigonometry methods. Figure 16 illustrates the irregular shapes studied here, which are convex polygon (item 1), non-convex polygons (items 2 and 3) and circles (item 4). Each of these items is defined by a list of vertices sorted clockwise, except for circles that are defined by a positive diameter. Each item has a reference point located at its left extremity and is used when packing such items into a bin. For polygons, this reference point corresponds to the highest-leftmost vertex, while in circles this point is the leftmost one. In addition, each item j has a positive demand d_j and for the items given in the figure, their demands are $d_1 = 7$, $d_2 = 9$, $d_3 = 15$, and $d_4 = 12$.

One strategy adopted in the literature, for example see Toledo *et al.* (2013), to represent the bin is by using a discrete grid of points. Considering this, the reference point of each item is

Figure 16 – Four irregular shaped items according to [Segenreich and Braga \(1986\)](#).

arranged at a given point of this grid. Figure 17 shows an example of a grid for a bin and the feasible points where is possible to arrange each item can be arranged Figure 16. We define the *inner-fit raster* of an item as the set of feasible points where such items can be packed without extrapolating the fixed dimensions of the bin. Note in Figure 17 that the square item packed at a point outside the inner-fit raster (the red point) extrapolates the bin dimensions, then occasioning an infeasible solution.

(a) Example of inner-fit raster for the 2D one-open dimension problem in which the right side is open.

(b) Example of inner-fit raster for the 2D two-open dimension problem. In this case, both right and upper sides are open.

Figure 17 – The inner-fit raster of some irregular shaped items are represented by the blue dots in the bin's grid. Note that the inner-fit raster of each item contains only points of the grid where such items can be packed generating a feasible solution.

By using a grid of points in combination with the inner-fit raster, we still cannot avoid overlapping between items, which is a major challenge when dealing with irregular shaped items. In order to do this, we present the concept of *no-fit raster*. A similar concept was proposed in [Toledo et al. \(2013\)](#), but limited to convex and non-convex polygonal items. The no-fit raster can be applied to any type of item, regardless of whether items are made of polygons or complex curves.

Definition 1. Let S be the set of items already packed in the bin and i be the next item to be packed. Denote $N_{S,i}$ as the set of points in the grid composed by the intersection of the inner-fit

raster of all items $j \in S$ and i . The no-fit raster of S and i is the subset of $N_{S,i}$, in which any point k of this subset enables us to pack i without overlapping with any $j \in S$.

In Figure 18, there are two examples of no-fit raster. In Figure 17a, the circle has been already been packed and the next item to be allocated is the polygon. The no-fit raster consists of all blue dots where the polygon can be packed without generating overlapping with the items already been packed. The no-fit raster in Figure 17b now represents the case in which the polygon has been already arranged, while the circle is the next to be packed. It is interesting to note that the no-fit raster in this Figure is equal to the one in Figure 17a by a rotation of 180° .

Figure 18 – Examples of no-fit raster, which are the blue dots (feasible positions) to pack, while the unfeasible positions are marked by black dots and x .

Computation of the no-fit raster can be done by using direct trigonometry (vertex to vertex), no-fit polygon (a simplification of direct trigonometry) and phi-functions (using non-linear equations). However, regardless of the method adopted (here direct trigonometry is used due to the representation used for the inner-fit raster), the no-fit raster can be computed for any type of irregular shaped items. On the other hand, its limitation is due to the size of the grid adopted to the inner-fit raster. One way to overcome this limitation is by using a grid of points in which the distance between any two subsequent points is as small as possible. Figure 19 presents an example of complex irregular shaped items (consisting of curves and holes) that can use the no-fit raster concept.

2.3 Mathematical Models

In this section we present integer models for each of the problems under consideration, in which the second model has a quadratic objective function and come as extension from the model proposed in Toledo *et al.* (2013) for the 2D one-open dimension problem with irregular shaped items.

The parameters used in the models are:

The variables are:

Figure 19 – Example of complex items that can use the concept of no-fit raster to check overlap.

- d - point of the grid;
- t - type of item;
- x_t - horizontal distance between the reference point and the rightmost point of item t ;
- x_d - horizontal distance between the origin of the bin and point d ;
- y_t - vertical distance between the reference point and the uppermost point of item t ;
- y_d - vertical distance between the origin of the bin and point d ;
- d_t - demand of item t ;
- $NFR_{t,u}^d$ - no-fit raster between items t and u in which the reference point of t is at point d ;
- IFR_t - inner-fit raster of item t .
- δ_t^d - is 1 if the reference point of item t is at point d ; and 0, otherwise;
- L - is the size of the length for the bin;
- W - is the size of the width for the bin.

First, we present the formulation for the 2D one-open dimension problem as in [Toledo et al. \(2013\)](#), then we extend it to the problem with two-open dimensions. Therefore, the integer linear model for the 2D one-open dimension problem with irregular shaped items can be found in Eqs. (2.1) to (2.6).

$$\text{minimize } L \quad (2.1)$$

$$\text{subject to: } (x_d + x_t)\delta_t^d \leq L, \quad \text{for all items } t; \forall d \in IFR_t; \quad (2.2)$$

$$\sum_{d \in IFR_t} \delta_t^d = q_t, \quad \text{for all items } t; \quad (2.3)$$

$$\delta_u^e + \delta_t^d \leq 1, \quad \text{for all items } t, u; \forall d \in IFR_t; \forall e \in NFP_{t,u}^d; \quad (2.4)$$

$$\delta_t^d \in \{0, 1\}, \quad \text{for all items } t; \forall d \in IFP; \quad (2.5)$$

$$L \geq 0. \quad (2.6)$$

The objective function (2.1) minimizes the open dimension (length) of the bin requiring that all items must be packed without overlapping. For that, constraints (2.2) assure that the items are packed, thus respecting length L of the bin, while constraints (2.3) ensure that the demand of each item t must be met. Constraints (2.4) impose the non-overlapping conditions for items u and t , while constraints (2.5) and (2.6) indicate the domain of each variable. Note that this model returns an optimal solution in accordance with the grid adopted for the inner-raster fit, which in turn may not be the optimum solution for the continuous grid.

The model for the 2D two-open dimension problem for irregular shaped items is presented in Eqs. (2.7) to (2.10), which are an extension from the model for the one-open dimension problem. As the two-open dimension problem has two dimensions to be determined during optimization, naturally its model is non-linear due to the quadratic objective function, although its constraints are all linear in the variables.

$$\text{minimize } LW \tag{2.7}$$

$$\text{subject to: } (y_d + y_t)\delta_t^d \leq W \quad \text{for all items } t; \forall d \in IFR_t; \tag{2.8}$$

$$W \geq 0; \tag{2.9}$$

$$(2.2), (2.3), (2.4), (2.5) \quad \text{and} \quad (2.6). \tag{2.10}$$

The objective function of the model for the 2D two-open dimension problem can be found in Eq. (2.7) and aims to minimize the area of the rectangular envelope limited by variables L and W once all items are packed without overlapping. In this model, constraints (2.8) ensure that W limits the size of the bin's width when packing items (similar to constraints (2.2)), while constraint (2.9) indicates the domain of W . The other constraints are obtained from the previous model as indicated in (2.10).

2.4 Developed Heuristic

Genetic Algorithms are metaheuristics inspired in biological systems that use the concept of individuals (represent solutions via a chromosome), heredity (transmission of genetic characteristics), mutation (introduction of new genetic characteristics), crossover (crossing between individuals to create descendants), fitness function (evaluate how good is an individual is in comparison with the other ones) and natural evolution (best individuals may produce good descendants that can transmit genetic characteristics to the next population). These algorithms were introduced by Holland (1975) and since then various approaches have been developed using them.

A classic genetic algorithm begins with a list of individuals and a fitness function. Therefore, while a stopping criterion is not satisfied, the algorithm selects individuals using the

fitness function, it applies the crossover operation over them to generate descendants, then the mutation operation is applied to insert new characteristics and finally the population of the next generation is produced, which is composed of the best individuals (the ones from the current generation and their descendants).

The Random-Key Genetic Algorithm (RKGA) was proposed by [Bean \(1994\)](#) as an improvement over the classic genetic algorithm. In RKGA, each individual is a chromosome that represents a solution of the problem. The representation of a chromosome is given by an array of random-keys, in which keys are real numbers in the $[0, 1)$ interval. The best individuals (with the best fitness value) form the elite set that will be in the population of the next generation. New mutant individuals are also generated to integrate the population of the next generation. In order to complete the population of the next generation, RKGA executes crossover choosing pairs of chromosomes from the current population and then applying the parametrized uniform crossover presented in [Spears and Jong \(1991\)](#).

The Biased Random-Key Genetic Algorithm (BRKGA) is an improvement of the RKGA, which occurs in the crossover operation. In BRKGA, for each pair of chromosomes, one is chosen from the elite set and the other from the remaining population. Besides that, during the parameterized uniform crossover, the probability for each characteristic to come from the elite chromosome is biased (usually greater than 50%), while in RKGA this does not always happen.

To solving the open dimension problems of this study, we use the version of the BRKGA proposed in [Toso and Resende \(2015\)](#), for which we need to propose a smart way to code and decode a solution using a random-keys vector. Moreover, the fitness function needs to be well defined in order to ensure a fair evaluation of chromosomes, and therefore, the BRKGA will be able to evolve for good populations at each iteration. More details are given in the next subsections.

2.4.1 Coding and Decoding Chromosomes

Each individual is represented by a chromosome with n alleles associating alleles with items. In other words, for a chromosome there is a vector of n positions in which n is the total number of items available for packing. Each position of the vector has a real number in the $[0, 1)$ interval, namely a key. The i -th position of the vector represents the i -th item. As items have positive demands, equal or greater than one, then $n = \sum d_j$, for each item j . Figure 20 illustrates a chromosome for items in Figure 16, for which $d_1 = d_2 = d_3 = 1$ and $d_4 = 2$.

Figure 20 – Example of one chromosome associating items and keys.

The rotation in which each item is going to be placed is obtained observing its corresponding key in the vector. Since the keys are real numbers in the $[0, 1)$ interval and considering that an item has k admissible rotations, we divide the $[0, 1)$ interval into k equally sized subintervals. Therefore, the value of the key indicates which rotation is considered for the item. The keys are also used to define the order in which each item is going to be packed into the bin. To do this, we sort the keys into non-decreasing order and using this order we pack each item.

Figure 21 shows an example of a chromosome after sorting the keys in a non-decreasing order.

Figure 21 – Sorting criterion applied to the key vector to obtain a new sequence for packing items.

Concerning the key vector after sorting the operation performed, it is now necessary to generate a solution for both two-dimensional nesting problems under consideration. For the one-open dimension problem, we consider a bottom-left heuristic, while for the two-open dimension problem we develop a bottom-left rectangle heuristic. These heuristics are better explained using the examples in Figure 22 and Figure 23. In terms of fitness function, we consider length L after packing all items, while for two-open dimension problem it is used the area of the smallest rectangle is used that encloses items is used. Of course, the best individuals are those that generate solutions with the lowest fitness value.

Figure 22 presents how a solution for the 2D one-open dimension problem is constructed using the bottom-left algorithm. Based on the sequence (3, 4, 2, 1, 4) determined in Figure 21, the items are packed one by one into the bin by positioning their reference point at the lowest leftmost feasible points of the grid. To find a feasible point for a given item, the algorithm searches in the no-fit raster obtained from the items already packed and the item to be packed now.

Figure 23 shows how to construct a solution for a 2D two-open dimension problem using the bottom-left rectangle heuristic. The difference between the bottom-left and the bottom-left rectangle is the way that the bottom-left rectangle searches for feasible points. In this heuristic, the best point is the one obtained in the no-fit raster by searching in the grid from the bottom to the top and from left to right that generates the smallest rectangular envelope in terms of the total area. In the event of a tie, the point closer to the origin of the bin is chosen.

Figure 22 – Items are packed observing the sequence 3, 4, 2, 1, and 4 using the bottom-left heuristic.

Figure 23 – Items are packed observing the sequence 3, 4, 2, 1, and 4 using the bottom-left rectangle algorithm.

2.4.2 BRKGA

Next, we describe next the general structure of the BRKGA, which uses the bottom-left heuristics as a subroutine to pack items. The population of each generation has P individuals (or chromosomes) separated into two groups: the elite group P_e with the individuals having the best fitness value; and the non-elite group P_n with the remaining individuals of $P - P_e$. Each individual is a vector of n positions in which each position is a key, and the fitness value is associated with the objective of the problem under consideration (the one-open dimension or the two-open dimension problems). The BRKGA evolves for the next generation in which its population consists of elite group P_e , new randomly generated individuals that form the group of mutants P_m , and the best descendants obtained from the biased parametrized uniform crossover

operation over individuals from the elite and non-elite groups. The creation of descendants is biased in the sense that information from the an elite parent is obtained with a probability greater than 50%.

The BRKGA of this paper uses the specific framework proposed in [Toso and Resende \(2015\)](#) for which we need to define the decoding function in accordance with the problem being solved. Algorithm 1 presents the structure of the BRKGA with the bottom-left heuristics. The loop at lines 1 to 27 corresponds to the evolution of each generation, which is repeated while the number of generations is less than a maximum number imposed. The algorithm can also stop if 100% of the bin is occupied after packing all items and determining the size of the open dimensions (see line 24).

The BRKGA, at line 2, evaluates each chromosome by decoding it to a solution of the problem using the bottom-left heuristic and then the fitness function according to the nesting problem being solved (as explained in the previous subsection). Afterwards, at line 3, the BRKGA separates the individuals into two groups: elite P_e and non-elite $P_{\bar{e}}$. The mutant individuals are generated at random (see line 4), and afterwards the elite and mutant groups are added to the next population (see line 5). The loop from line lines 6 to 19 represents the biased parametrized uniform crossover, in which one parent is chosen at random from the elite group and the other one from the non-elite group and the probability ρ of the allele comes from the elite parent, which is greater than 50%. Next, the best evaluated chromosome according to the fitness function is saved (see lines 20 to 26). At the end, the best solution is returned. It is worth mentioning that the number of chromosomes in population P and consequently in groups P_e , $P_{\bar{e}}$, and P_m are set by the user.

In order to show how the BRKGA works over one of the nesting problems under consideration (particularly the one-open dimension problem), we present a small example in Figure 24. For an initial population with $|P| = 3$ chromosomes, we consider one chromosome (the best one) for the elite group P_e . The remaining chromosomes go to the non-elite group $P_{\bar{e}}$. The population of the next generation consists of 3 chromosomes: (i) one comes from the elite group P_e ; (ii) one is new and generated at random to form the mutant group P_m ; and, (iii) one is a descendent obtained by crossover. In the crossover, one parent is from P_e and the other comes at random from $P_{\bar{e}}$. After generating all descendants necessary to comply with the population of the next generation, the main loop of BRKGA restarts in which the first step is to evaluate this new population.

The descendent in Figure 24 is obtained from the crossover operation between the elite parent and the first chromosome in the non-elite group. The parents' key vectors are: $[0.6;0.7;0.1;0.8;0.9]$ of the elite parent; and $[0.9;0.5;0.8;0.2;0.7]$ of the non-elite parent. The descendant obtains 3 alleles from the elite parent due to biased probability and 2 alleles from the non-elite parent. Then, the descendant's key vector is $[0.6;0.5;0.1;0.2;0.9]$. Figure 25 shows this parametrized uniform crossover operation in detail.

Algorithm 1: Structure of the BRKGA for nesting problems

Input: P, P_e, P_m, n and ρ .
Output: x^* is the chromosome of the population with the best fitness value.

```

1 for  $g \leftarrow 1$  to maximum number of generations do
2   Decode each chromosome of population  $P$  by applying the sorting criteria, pack
   using the correspondent bottom-left heuristic and evaluate the fitness of the
   final packing according to the problem being solved;
3   Partition  $P$  in the elite group  $P_e$  and the non-elite group  $P_{\bar{e}}$ ;
4   Generate a group of mutant individuals  $P_m$  at random;
5   Add the elite group  $P_e$  and the mutant group  $P_m$  to the next population  $P^+$ ;
6   for  $i \leftarrow 1$  to  $|P - P_e - P_m|$  do
7     Choose one parent  $e$  randomly of  $P_e$ ;
8     Choose one parent  $\bar{e}$  randomly of  $P_{\bar{e}}$ ;
9     for  $j \leftarrow 1$  to  $n$  do
10      Flip a coin with probability  $\rho$  to show heads;
11      if it is heads then
12         $chromo_{desc}[j] \leftarrow chromo_e[j]$ ;
13      end
14      else
15         $chromo_{desc}[j] \leftarrow chromo_{\bar{e}}[j]$ ;
16      end
17    end
18    Add the descendant  $chromo_{desc}$  into the next population:
     $P^+ \leftarrow P^+ \cup \{chromo_{desc}\}$ ;
19  end
20  Find the chromosome  $x^+$  in  $P$  with the best fitness value;
21  if  $fitness(x^+) < fitness(x^*)$  then
22     $x^* \leftarrow x^+$ ;
23    if  $x^*$  has a packing that fulfills the bin after to obtaining the size of the open
    dimensions then
24      return the packing generated by  $x^*$ ;
25    end
26  end
27 end
28 return the packing generated by  $x^*$ ;

```

2.5 Computational Experiments

This section is divided into three parts. The first one is focused on discussing the parameters adopted by the BRKGA and how the grid is generated. The next two parts show the results for the 2D one-open dimension problem and the 2D two-open dimension problem, respectively, for which we present a detailed comparison of the BRKGA with other approaches from the recent literature in order to show that the BRKGA developed is very competitive. The description of the instances used for comparison is also given in these last two parts.

Figure 24 – BRKGA generating the population of the next generation $K + 1$ from generation K .

2.5.1 Parameter tuning

Next are presented the parameters adopted by the BRKGA and how the grid is discretized are then presented. Table 1 presents all the parameters that are required by Algorithm 1. Some of these parameters had their values obtained observing Toso and Resende (2015), while we consider two different probabilities for the key inherited from an elite father, thus generating two versions of the BRKGA, namely BRKGA50 with $\rho = 50\%$ and BRKGA70 with $\rho = 70\%$.

Table 1 – Parameters used by Algorithm 1

Parameter	Description	Value
n	number of elements in each chromosome	total number of items
P	size of the population	1000
P_e	size of the elite population	200
P_m	size of the mutant population	100
ρ	probability of each key being inherited from an elite father	50% and 70%
$max\ generations$	stopping criteria	1000

As can be observed in Goncalves and Resende (2015), which uses the BRKGA framework, it is very common to consider $\rho > 50\%$ with the assumption that good descendants can be generated. However, we obtained some good results in preliminary computational experiments for $\rho = 50\%$, and therefore, we decided to also report such results. As Toso and Resende (2015) mentioned, it is hard to define good parameters for genetic algorithms. In our preliminary tests, we considered values for ρ varying from 5% to 95% updated by 5% in which the best solutions were obtained for 50% and 70%. This is the reason why we are considering these two different values for ρ .

To the best of our knowledge, there is a procedure capable of computing an optimal grid for each instance. We noticed in the preliminary experiments that bad solutions, mainly in terms of runtime, were achieved when the size of the grid increased, i.e., a scale of 1:1000 (1 point for 1000 units) is worse than a scale of 1:100, which in turn is worse than a scale of 1:10. It

Figure 24 – BRKGA generating the population of the next generation $K + 1$ from generation K .

Figure 25 – Example illustrating the parametrized uniform crossover for the BRKGA.

follows that both the bottom-left and the bottom-left rectangle heuristics obtained the best results when using a 1:0.5 as a scale. This might have happened because most of the literature instances have items with vertices defined at integer points. On the other hand, instances with circles are more dependent on the grid chosen. Therefore, we define the following rule to compute the grid: for instances with a width smaller than 2500, we use a scale of 1:1; for instances with a width between 2500 and 5000, we use a scale of 1:10; and, for instances with width greater than 5000, we use the 1:100 as a scale.

The approaches developed in this chapter were implemented in the C++ programming language and the computational experiments were performed in an Intel® Core™ i7-2600, 3.40 GHz, 16 GB RAM and Ubuntu 12.04 as an operating system. The results in the next subsections were obtained considering 10 executions of the proposed algorithm, each one limited to 3600 seconds. These results clearly validate the BRKGA, as well as the parameters adopted as the results are very competitive with the recent nesting literature.

2.5.2 Results for the 2D one-open dimension problem

We divided the computational experiments involving the 2D one-open dimension problem for irregular shaped items into three parts. First, we compared the BRKGA with approaches from the literature; one based on an integer linear model that uses the grid as defined here. In the next section, we present the results for the classical nesting instances available in ESICUP¹ and we compare the BRKGA with the best heuristic approaches in the literature. Finally, we present results for instances adapted from the literature that consider circles.

2.5.2.1 Instances in Toledo et al. (2013)

We consider three sets of instances that were proposed in Toledo et al. (2013) and used by MirHassani and Bashirzadeh (2015) recently. The aim is to evaluate the performance of the proposed algorithm for the grid defined in Section 2.5.1 as exact solutions for some of these instances are known. In addition, we can compare our results with the ones obtained by heuristics from the literature. There are 16 instances under consideration that are variations of RCO, BLAZEWICZ and SHAPES instances. The number of items vary between 7 and 43 and the width vary between 15 and 40. More details about them can be found in Toledo et al. (2013).

Table 2 reports the results of BRKGA50 and BRKGA70 for the 16 instances used. Each row of the table gives the name, number of items, and optimal length (if known) of the instance; the best length computed among the 10 runs, the average length and the average runtime (in seconds) for both the BRKGA50 and BRKGA70.

The results show that the best length computed with BRKGA70 is always better than or equal to the one computed with BRKGA50 and the average length reported by BRKGA70 is the best for 11 instances (RCO3, RCO4, RCO5, BLAZEWICZ2, BLAZEWICZ4, BLAZEWICZ5, SHAPES4, SHAPES5, SHAPES7, SHAPES9, and SHAPES15). It is important to observe that by modifying only the parameter $\rho > 50\%$, the algorithm had a better behavior while maintaining a similar runtime. For this reason, we consider only BRKGA70 in the comparison with the literature.

The comparison of BRKGA70 with other approaches from the literature is shown in Table 3. In this table, we compare our results with those from the MIP model from Toledo et al. (2013) and the RGRASP and GRASP proposed by MirHassani and Bashirzadeh (2015). The runtime (in seconds) report by MirHassani and Bashirzadeh (2015) is the average runtime using all the values of α .

Observing the results, our method achieved the best (or equal) solution for 13 out of 16 instances, outperforming the other approaches. When compared with the MIP model, BRKGA70 obtained a worse solution only for the instance BLAZEWICZ3, but the authors found this solution taking 5583.8 seconds, in contrast with 13.4 seconds for BRKGA70. On the other hand,

¹ <https://www.fe.up.pt/esicup>

Table 2 – Results for the instances of Toledo *et al.* (2013).

Instance	Total items	Optimal Length	BRKGA50			BRKGA70		
			Best Length	Average Length	Average RT (s)	Best Length	Average Length	Average RT (s)
BLAZEWICZ1	7	8	8	8.0	2.9	8	8.0	2.9
BLAZEWICZ2	14	14	15	15.0	7.4	14	14.2	7.3
BLAZEWICZ3	21	20	21	21.0	13.6	21	21.0	13.4
BLAZEWICZ4	28	-	28	28.2	20.1	28	28.0	19.8
BLAZEWICZ5	35	-	35	35.9	28.4	35	35.0	37.9
RCO1	7	8	8	8.0	2.7	8	8.0	2.9
RCO2	14	15	15	15.0	7.6	15	15.0	7.5
RCO3	21	22	22	22.4	13.5	22	22.0	13.5
RCO4	28	-	30	30.0	20.3	29	29.9	20.3
RCO5	35	-	37	37.4	29.1	37	37.0	28.5
SHAPES2	8	14	14	14.0	10.2	14	14.0	10.4
SHAPES4	16	25	25	25.1	25.4	25	25.0	25.8
SHAPES5	20	-	30	30.7	35.3	30	30.0	35.3
SHAPES7	28	-	42	42.2	60.2	42	42.0	60.2
SHAPES9	34	-	49	50.0	81.3	48	48.7	81.5
SHAPES15	43	-	63	63.3	121.7	61	61.8	120.9

when we compare the results of the best heuristic from the literature that solves these instances (namely RGRASP), we note that BRKGA70 outperforms it in 7 instances; it has a similar performance in 6 instances and performs worse in 3 instances. Even with these approaches tested on different computers, we note that both have similar runtimes. It is important to mention that BRKGA70 improved the solution of SHAPES7 and SHAPES9 by 6.67% and 11.11% if compared with the respective best literature solution. A graphical illustration of these solutions is given in Figure 26.

Table 3 – Comparison with the recent approaches from the literature.

Instance	BRKGA70		RGRASP	GRASP				RT (s)	MIP model		
	Best	RT (s)		$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.9$		Solution	T (s)	GAP
BLAZEWICZ1	8	2.9	8	8	8	8	8	0.9	8	0.7	0.0
BLAZEWICZ2	14	7.3	14	15	15	15	16	4.8	14	16.0	0.0
BLAZEWICZ3	21	13.4	20	23	23	22	23	13.8	20	5583.8	0.0
BLAZEWICZ4	28	19.8	30	31	31	30	31	28	28	TL	10.7
BLAZEWICZ5	35	37.9	35	37	37	36	37	40.3	35	TL	14.3
RCO1	8	2.9	8	8	8	8	8	1	8	0.6	0.0
RCO2	15	7.5	16	17	16	15	16	5.7	15	6.3	0.0
RCO3	22	13.5	22	22	23	22	22	14.5	22	2393.4	0.0
RCO4	29	20.3	27	30	30	29	28	28.2	29	TL	3.5
RCO5	37	28.5	38	40	39	38	39	43.9	37	TL	8.1
SHAPES2	14	10.4	14	15	15	14	14	3.9	14	0.5	0.0
SHAPES4	25	25.8	28	28	27	27	27	42	25	17951.3	0.0
SHAPES5	30	35.3	32	33	33	33	33	79.3	30	TL	13.3
SHAPES7	42	60.2	47	49	48	47	47	240.6	45	TL	39.4
SHAPES9	48	81.5	55	60	59	57	58	301.7	54	TL	40.4
SHAPES15	61	120.9	57	61	59	58	59	311.1	67	TL	40.8

RGRASP and GRASP executed in a computer with 4 GB of RAM and a 2.2-GHz processor.

MIP model executed in a Xeon 3.47 GHz with time limit set to 18,000 seconds.

Figure 26 – Solutions computed with BRKGA70 for SHAPES7 and SHAPES9.

2.5.2.2 Benchmark instances

In this section, we report the results obtained from a set of 15 instances available in the ESICUP. These instances vary from artificial ones to ones from real applications of the textile industry, and they are considered classical benchmark instances for nesting problems. The number of items ranges from 10 to 99 and the bin width ranges from 15 to 5752. Table 4 presents some details of these instances: the number of different items, the total number of items (as some items may have a greater demand greater than one), the angles in which the items are allowed to rotate and the bin width.

Table 4 – Details of the benchmark instances from ESICUP.

Instance	Different items	Total items	Rotation angles in degrees	Bin width
Albano	8	24	0, 180	4900
Dagli	10	30	0, 180	60
Dighe1	16	16	0	100
Dighe2	10	10	0	100
Fu	11	12	0, 90, 180, 270	38
Jakobs1	22	25	0, 90, 180, 270	40
Jakobs2	22	25	0, 90, 180, 270	70
Mao	9	20	0, 90, 180, 270	2550
Marques	8	24	0, 180	104
Shapes0	4	43	0	40
Shapes1	4	43	0, 180	40
Shapes2	7	28	0, 180	15
Shirts	8	99	0, 180	40
Swim	10	48	0, 180	5752
Trousers	17	64	0, 180	79

The results for BRKGA50 and BRKGA70 on the benchmark instances are reported in Table 5, which shows the bin occupation (in percentage) for the best and average solution, as

well the average runtime (in seconds) for 10 executions. It is worth mentioning that the bin (area) occupation is reported instead of the length, because in the literature it is common to report this information for such instances (note that minimizing the length also implies in reducing the occupied area). Once again, BRKGA70 shows its superiority over BRKGA50 in terms of average occupation as the former is better for 8, equal for 3 and worse for 4 instances. Similarly, now comparing the best solutions, BRKGA70 outperforms BRKGA50 for 5 instances and ties for 6 instances, while the latter is better for 4 ones. Their runtimes are very close for almost all instances, except for DIGHE1 and DIGHE2, for which they have a significant difference (greater than 50%).

Table 5 – Results for the benchmark instances from ESICUP.

Instance	BRKGA50			BRKGA70		
	Best Occup. (%)	Average Occup. (%)	Average RT (s)	Best Occup. (%)	Average Occup. (%)	Average RT (s)
Albano	84.82	83.65	3600.0	84.33	83.77	3600.0
Dagli	82.56	82.56	498.1	81.84	81.84	500.3
Dighe1	100.00	100	195.5	100.00	100	98.0
Dighe2	100.00	100	36.7	100.00	100	302.0
Fu	86.36	86.36	985.6	86.36	86.36	1022.0
Jakobs1	81.67	81.67	491.5	89.10	83.15	479.3
Jakobs2	74.23	74.23	1824.8	80.42	77.28	1748.4
Mao	77.74	77.39	3600.0	75.66	74.65	3600.0
Marques	87.56	87.56	3600.0	87.56	87.45	3600.0
Shapes0	65.4	65.4	1236.2	66.50	66.5	1125.1
Shapes1	71.25	71.25	2152.6	72.55	72.55	2008.4
Shapes2	80.00	77.71	379.7	80.00	80	424.4
Shirts	87.10	86.75	3600.0	87.10	87.1	3600.0
Swim	66.75	66.64	3600.0	67.72	66.49	3600.0
Trousers	88.90	88.04	3600.0	88.55	88.18	3600.0

As it outperformed BRKGA50 in the previous comparison, a comparative analysis of BRKGA70 is made in Table 6 with the most recent and best approaches from the literature that report results on the benchmark instances from ESICUP. They are: the Random-Key Genetic Algorithm for the Nesting Problem (RKGA-NP) from [Pinheiro, Júnior and Saraiva \(2015\)](#), the Pairwise Exact Placement Algorithm (PEPA) from [Sato, Martins and Tsuzuki \(2015\)](#), the Guided Cuckoo Search (GCS) from [Elkeran \(2013\)](#), the Beam Search (BS) from [Bennell and Song \(2010\)](#) and the Bottom-Left-Fill from [Burke et al. \(2006\)](#). We noted that some of these approaches adopted similar strategies compared to the ones proposed in this chapter, which is the case of RKGA-NP and BLF, based on genetic algorithms and bottom-left heuristics, respectively. It is important to point out that all results were extracted from the original articles and these approaches did not consider the bin discretized over a grid, instead they assumed a continuous domain when arranging the items. Moreover, RKGA-NP from [Pinheiro, Júnior and Saraiva \(2015\)](#) was applied in only 10 out of the 15 instances.

The results in Table 6 show that BRKGA70 is equal to or outperforms (in terms of

Table 6 – Comparison with the literature for the benchmark instances from ESICUP.

Instance	BRKGA70		RKGA-NP		PEPA		GCS		BS		BLF	
	Best (%)	RT (s)	Best (%)	RT (s)	Best (%)	RT (s)	Best (%)	RT (s)	Best (%)	RT (s)	Best (%)	RT (s)
Albano	84.33	3600.0	84.7	10230.0	86.66	7200.0	89.58	1200.0	87.88	5460.0	84.6	118.0
Dagli	81.84	500.3			85.23	7200.0	89.51	1200.0	87.97	17331.0	83.7	204.0
Dighe1	100	98.0	100.0	2053.0	100	7200.0	100	600.0	100	1.4	77.4	22.0
Dighe2	100	302.0	100.0	2291.0	89.61	7200.0	100	600.0	100	0.3	79.4	10.0
Fu	86.36	1022.0			91.84	7200.0	92.41	600.0	90.28	1192.0	86.9	24.0
Jakobs1	89.1	479.3	81.7	10268.0	89.07	7200.0	89.1	600.0	85.96	2193.0	82.6	74.0
Jakobs2	80.42	1748.4	79.8	12205.0	80.42	7200.0	87.73	600.0	80.4	75.0	74.8	213.0
Mao	75.66	3600.0			81.9	7200.0	85.44	1200.0	84.07	16757.0	79.5	38.0
Marques	87.56	3600.0	84.5	6871.0	88.91	7200.0	90.59	1200.0	88.92	10692.0	86.5	25.0
Shapes0	66.5	1125.1	67.3	1826.0	66.5	7200.0	68.79	1200.0	64.35	8.0	60.5	93.0
Shapes1	72.55	2008.4	66.4	3025.0	70.82	7200.0	76.73	1200.0	72.55	398.0	66.5	82.0
Shapes2	80	424.4			82.03	7200.0	84.84	1200.0	81.29	5603.0	77.7	32.0
Shirts	87.1	3600.0			84.48	21600.0	88.96	1200.0	89.69	6217.0	84.6	499.0
Swim	67.72	3600.0	73.6	14352.0	67.65	21600.0	75.94	1200.0	75.04	15721.0	68.4	1239.0
Trousers	88.55	3600.0	90.9	16025.0	87.57	21600.0	91	1200.0	90.38	5988.0	88.5	789.0

RKGA-NP executed in a 3.60 GHz Intel i5 with 4 GB of RAM memory and time elapsed for best solution.
 PEPA executed in a 2.4 GHz Xeon with time limit set to 21,600 seconds.
 GCS executed in a 2.2 GHz Intel i7 with 6 GB of RAM memory and time limit set to 12,000 seconds.
 BS executed in a 2.8 GHz Intel Pentium D without time limit.
 BLF executed in a 2.0 GHz Intel Pentium 4 and time of 100 iteration runs.

solution quality) RKGA-NP in 6 out of 10 instances, PEPA in 9 out of 15 instances, and BLF in 10 out of 15 instances. For the instances in which BRKGA70 is worse than these approaches, the worst result when compared with RKGA-NP is for the Swim instance with a difference of 5.87%. When compared with PEPA, the worst result is for Mao with a difference of 6.24%. When compared with BLF, the worst result is for Mao with a difference of 3.84%.

On the other hand, BRKGA70 is not capable of computing better solutions than BS for 9 instances. Moreover, it is also not able to compute better solutions than GCS for 12 out of 15 instances. In comparison to BS, the worst result is for the Mao instance with a difference of 8.41%, but this drops to 2.07% when considering the average solution. Now, comparing with GCS, the results are worse, with the worst difference of 9.78% for the Mao instance (4.20% considering the average solution). In terms of runtime, for completeness, the average runtime for all instances can be observed in Table 27 highlighting that computers are different.

Figure 27 – Average time in seconds over all the instances.

Analyzing the data in Table 6, the comparison of average occupation obtained by these approaches is given by: BRKGA70 (82.35%), RKGA-NP (81.45%), PEPA (77.14%), GCS (85.48%), BS (85.16%) and BLF (78.77%). For RKGA-NP, we considered only the 10 instances reported in [Pinheiro, Júnior and Saraiva \(2015\)](#), for which BRKGA70 obtained an average occupation of 82.54%. In addition, for BLF we considered the average of the best results because [Burke et al. \(2006\)](#) do not present the average results for all instances.

An overview of the best results in Table 6 is given in Figure 28, where the pie charts contain the number of instances for which BRKGA70 is better, equal or worse than each of the approaches from the literature presented here. The bar charts present the average of occupation calculated over all the instances under consideration. Moreover, Figure 29 presents some solutions that are considered the best ones in the literature that BRKGA70 calculated.

2.5.2.3 Instances with circles

With the objective of showing that our algorithms can be applied satisfactorily to deal with general shaped items (and not only polygons), we present the next results for instances that also consider circles as well, which were proposed by [Segenreich and Braga \(1986\)](#) with 2

Figure 28 – Overview of BRKGA70 results in terms of the best solution and best average occupation in comparison with the literature.

Figure 29 – Best solutions computed so far by BRKGA70 for the benchmark instances from ESICUP.

instances, and [Błażewicz, Hawryluk and Walkowiak \(1993\)](#) with 10 instances. The number of items vary between 8 and 43, and the bin width is 15 for the 10 instances of [Błażewicz, Hawryluk and Walkowiak \(1993\)](#) and 40 for 2 instances, Primitive0 and Primitive1, of [Segenreich and Braga \(1986\)](#). A detailed description of these adapted instances is given in [Appendix 2.7](#). To the best of our knowledge, there is no other paper which addresses the nesting literature that used them as 2D one-open dimension problem instances, which is why we do not make any comparisons with the literature.

The results for the instances with circles are presented in [Table 7](#) and the solutions are in terms of bin occupation, as commented before. We note that BRKGA70 is better than or equal to BRKGA50 observing the best occupation solution for all the instances. On the other hand, the difference considering all the instances is very small, i.e, 0.58% (in favor of BRKGA70) in terms of the average occupation and the same remains true for the average runtime. [Figure 30](#)

illustrates the solutions with the best occupation that were computed with BRKGA70 in the context of the 2D one-open dimension problem.

Table 7 – Results for the instances with circles.

Instance	BRKGA50			BRKGA70		
	Best Occup. (%)	Average Occup. (%)	Average RT (s)	Best Occup. (%)	Average Occup. (%)	Average RT (s)
Test1	71.11	71.11	86.8	71.11	71.11	85.8
Test2	60.70	60.70	222.7	60.70	60.70	219.2
Test3	72.14	72.14	362.4	72.14	72.14	347.2
Test4	48.89	48.89	295.3	48.89	48.89	288.8
Test5	80.57	79.35	507.1	80.57	79.48	497.7
Test6	66.67	66.67	350.7	66.67	66.67	350.2
Test7	75.95	75.95	535.5	75.95	75.95	530.0
Test8	74.91	74.91	566.0	74.91	74.91	561.5
Test9	75.22	74.79	914.2	75.85	75.23	913.5
Test10	78.01	77.50	1220.9	80.98	79.71	1176.4
Primitive0	72.15	72.15	1767.2	73.32	73.32	1794.9
Primitive1	73.32	73.32	2311.9	73.32	73.32	2533.6

2.5.3 Results for the 2D two-open dimension problem

In the next part, we present the results for the instances solved for the 2D two-open dimension problem, which has applications, for example, in the microprocessor and furniture industry. We have not found any instances for this problem that consider a reasonable number of items as the literature on this problem solved small instances, sometimes using free rotations and without presenting sufficient details that enable us to reproduce them [Kallrath \(2009\)](#); [Bennell et al. \(2015\)](#).

Bearing this in mind, we consider the same benchmark instances from ESICUP, therefore the researchers and practitioners can reproduce the BRKGA results and made comparisons when necessary. For creating the instances, only the items are necessary as the objective of the 2D two-open dimension problem is to find a rectangular envelope of a minimum area that encloses all items. The results are presented in Table 4 and follow the same format of the previous discussion, i.e., we report the best and average occupation (in percentage), as well as the average runtime (in seconds).

In Table 4, it can be observed that BRKGA70 presents better results than or equal to BRKGA50 for 10 instances and BRKGA50 outperforms BRKGA70 for 5 instances (Albano, Dagli, Mao, Shapes1, and Swim). However, when comparing the differences in the average occupation, we note that both obtained results close to each other with a very small difference of 0.15% when considering the average for all the instances (76.05% for BRKGA50 and 75.90% for BRKGA70). The average over the best occupation column reaches a difference of 0.11% to BRKGA70. In terms of the average runtime, both heuristics reported similar results.

Figure 30 – Solutions computed with BRKGA70 for the instances with circles.

As these instances were also solved in the context of the 2D one-open dimension problem, we decided to compare the solutions obtained by comparing the average occupation (we used the best result found by BRKGA50 or BRKGA70) and the average runtime (the respective one was obtained from the average occupation value as shown in Tables 5 and 4). For the average occupation, all the results for the 2D one-open dimension problem are better than those for the two-open dimension problem with a difference of 6.34% on average. Considering the respective average runtime, the difference is 214 seconds, on average, to the one-open dimension problem. As we suspected, the two-open dimension problem is harder than the one-open dimension problem as the former has two dimensions to be optimized, while the latter has only has one and then the number of positions to arrange the items decreases accordingly.

2.6 Conclusions

This paper presents a competitive heuristic based on the Biased Random-Key Genetic Algorithm combined with bottom-left heuristics for packing irregular shaped items in a two-dimensional bin with one and two dimensions open. In order to solve these two problems, we generalize the concept of no-fit raster for any type of irregular shaped items and use it to check

Table 8 – Results for the 2D two-open dimension problem on benchmark instances.

Instance	BRKGA50			BRKGA70		
	Best Occup. (%)	Average Occup. (%)	Average RT (s)	Best Occup. (%)	Average Occup. (%)	Average RT (s)
Albano	82.45	80.43	3600.0	81.56	79.95	3600.0
Dagli	79.55	76.47	3600.0	77.47	75.51	3600.0
Dighe1	79.95	75.89	3600.0	79.95	75.71	3600.0
Dighe2	100.00	95.37	234.9	100.00	95.45	57.7
Fu	80.04	78.95	615.1	83.82	80.18	562.9
Jakobs1	81.16	80.43	267.3	81.16	79.73	516.9
Jakobs2	74.81	72.79	278.3	76.59	73.33	940.0
Mao	80.05	76.53	3600.0	75.08	74.57	3600.0
Marques	80.61	79.73	3600.0	81.75	79.78	3600.0
Shapes0	61.46	59.85	414.7	61.74	59.94	384.1
Shapes1	68.03	65.81	898.9	66.72	65.22	1010.7
Shapes2	74.14	73.74	149.4	77.51	74.76	428.2
Shirts	81.82	81.04	3600.0	83.56	81.03	3600.0
Swim	66.74	64.86	3600.0	65.61	64.83	3600.0
Trousers	80.03	79.03	3600.0	80.03	78.61	3600.0

non-overlapping between any pair of items. The bottom-left heuristics are used to place items according to the sequence and angle of rotation given by the BRKGA.

For each nesting problem, one mathematical model based on a grid of points was presented, however both models were not used to solve the instances due to the size of the instances and consequently the lack of memory and high computational time required for the models. Results are then reported for two variants of the Biased Random-Key Genetic Algorithm: BRKGA50 and BRKGA70, which differ only in the crossover phase, particularly in the biased probability used to obtain information from elite parents. The comparison between these two algorithms showed that for the 58 instances considered, BRKGA70 was better in 18, worse in 9 and equal in 31. Moreover, the runtime was very similar for almost all the instances.

Results for the 2D one-open dimension problem were divided into three groups: (i) comparison with approaches considering the same domain for the grid (set with 16 instances), in which the BRKGA was able to find an optimal solution for 8 instances and improve the results of two instances; (ii) comparison involving the hard and benchmark set of 15 ESICUP instances for which the BRKGA outperformed the approaches from [Burke *et al.* \(2006\)](#), [Pinheiro, Júnior and Saraiva \(2015\)](#) and [Sato, Martins and Tsuzuki \(2015\)](#), while compared to [Elkeran \(2013\)](#) and [Bennell and Song \(2010\)](#) the BRKGA reached a difference concerning the solutions of 3.13% for [Elkeran \(2013\)](#) and 2.81% for [Bennell and Song \(2010\)](#), which are small values; and, (iii) new results for 12 instances with circles for which the literature could not resolve adequately. In the case of the 2D two-open dimension problem, we propose the first heuristic of the literature capable of solving medium and large-sized instances.

After all, the BRKGA with the bottom-left heuristics is a simple but effective approach

Figure 31 – Solutions computed for the 2D two-open dimension problem.

capable of returning good solutions very quickly for these nesting problems. For any expert system, this approach can be the core engine that generates solutions in conformity with the inputs. As the system has the capacity of learning as the time passes, naturally its parameters will be calibrated by the system according to the inputs and solutions being produced. The decision maker may help the system giving new values for the parameters as necessary.

2.7 Data of Instances with Circles

2.7.1 Instances proposed in *Segenreich and Braga (1986)*

The Primitive0 and Primitive1 instances consist of the items in Figure 32. The items cannot rotate in Primitive0, but they can rotate of 0 and 180 degrees in Primitive1. The characteristics of each item are shown below.

Figure 32 – Illustration of the items.

- Item 1 has a demand equal to 7 and four vertices: (0, 0), (6, -6), (12, 0) and (6, 6).
- Item 2 has a demand equal to 9 and six vertices: (0, 0), (6, 0), (7, -2), (11, 2), (11, 4) and (0, 4).
- Item 3 has a demand equal to 15 and nine vertices: (0, 0), (2, 0), (2, 4), (13, 4), (13, 0), (14, 0), (14, 6), (2, 6) and (0, 4).
- Item 4 has a demand equal to 12 and a diameter equal to 6.

2.7.2 Instances proposed in *Błażewicz, Hawryluk and Walkowiak (1993)*

The ten instances of *Błażewicz, Hawryluk and Walkowiak (1993)* are based on the seven items described below and are shown in Figure 33. Table 9 has the demand of each item.

- Item 1 has six vertices: (0, 0), (2, 1), (4, 0), (4, -3), (2, -4) and (0, -3).
- Item 2 has eight vertices: (0, 0), (3, 0), (2, -2), (3, -4), (3, -5), (1, -5), (-1, -3) and (-1, 1).
- Item 3 has a circle with diameter 4.
- Item 4 has eight vertices: (0, 0), (2, -1), (4, -0), (3, -2), (4, -5), (2, -4), (0, -5) and (1, -3).
- Item 5 has seven vertices: (0, 0), (5, -0), (5, -5), (4, -5), (3, -3), (2, -2) and (0, -1).
- Item 6 has three vertices: (0, 0), (2, -3) and (-2, -3).
- Item 7 has four vertices: (0, 0), (2, 0), (2, -2) and (0, -2).

Figure 33 – Illustration of the items.

Table 9 – Demand of the instances proposed by [Błażewicz, Hawryluk and Walkowiak \(1993\)](#).

Instance	Piece							# of item
	1	2	3	4	5	6	7	
Test1	8							8
Test2			10					10
Test3					10			10
Test4				10				10
Test5	2	2	3	1	1	1	2	12
Test6						10		10
Test7	1		5	2	2	2	1	13
Test8	5		1	2		4		12
Test9	5	5	5	5				20
Test10	4	4	4	4	4	4	4	28

A GENERAL HEURISTIC FOR TWO-DIMENSIONAL NESTING PROBLEMS WITH LIMITED-SIZE CONTAINERS

Two-dimensional (2D) nesting problems deal with the placement of irregular items inside a set of containers (otherwise boards) so as to optimize an objective. Although in real life problems the containers have limited dimensions and may contain holes, few papers propose heuristics to solve these, and so far no heuristic has been proposed to solve all 2D nesting problems with limited-size containers. In this chapter, we propose a new general heuristic for the Placement Problem, the Knapsack Problem, the Cutting Stock Problem, and the Bin Packing Problem. This heuristic makes random use of five placement rules inspired on the bottom left heuristic. In order to reduce the geometric complexity of these problems we resort to the concepts of no-fit raster and inner-fit raster.

Extensive computational experiments were run with a total of 1100 instances, and the results achieved exhibit a substantial improvement over those in the literature. This allows us to claim that H4NP, the proposed heuristic, is the most effective solution method published for 2D nesting problems with limited-size containers.

The solutions were improved for 39 out of 70 instances of different problems, we present two new sets of benchmark problems for the Placement Problem and for the Knapsack Problem. For the Bin Packing Problem, where the number of instances in the literature is very high, we propose a method to compare the results.

3.1 Introduction

Cutting and Packing (C&P) problems consist in placing one set of small objects (pieces) inside a set of large objects (containers) so as to optimize an objective. These problems have a

strong economic and ecological impact and their proper solving is essential for many sectors of the economy, such as the textile, footwear, automotive and shipbuilding industries, to mention only a few.

In this chapter we propose a general heuristic to solve the two-dimensional version of the C&P problems, with irregular pieces (convex and/or non-convex polygons) and limited-size containers, i.e. the basic problem types proposed by [Wäscher, Haußner and Schumann \(2007\)](#): Placement Problem (PP), where a set of weakly heterogeneous pieces is assigned to a container to maximize the value associated with the assigned pieces; Knapsack Problem (KP), where a set of strongly heterogeneous pieces is assigned to a container to maximize the occupation of the container; Cutting Stock Problem (CSP), where a weakly heterogeneous assortment of pieces is assigned to the minimum number of containers; and Bin Packing Problem (BPP), where a strongly heterogeneous assortment of pieces is assigned to the minimum number of containers (see Table 10).

Table 10 – Overview of the C&P problem variants studied in this chapter.

Number containers Pieces Assortment	One	Multiple
	PP	CSP
Weakly heterogeneous		
Strongly heterogeneous	KP	BPP

This paper presents the H4NP, a general heuristic based on placement rules, to find the best solutions for the PP, KP, CSP and BPP problems. We adopt the no-fit raster and inner-fit raster concepts to avoid piece overlapping and to position the pieces inside the container, be it rectangular or irregular.

The proposed general heuristic improved a lot the solutions in the literature. H4NP uses the no-fit raster and the inner-fit raster to reduce the geometric complexity. With this heuristic 39 results from the literature have been improved, and for 72.86% of the benchmark instances with which the H4NP was run better or equal results were found. In addition, we propose two new sets of instances and metrics of comparison for future papers.

This paper is organized as follows: Section 3.2 presents a brief literature review, while Section 3.3 provides some details on the geometry of the problem. Section 3.4 discusses the heuristic proposed, and Section 3.5 introduces the comparisons with benchmark problems from the literature. Section 3.6 provides a summary of the main contributions of this paper, and finally Section 3.7 introduces the reader to the conclusions thereof.

3.2 Review of the literature

This section presents a brief survey of the literature concerning the problems studied in this chapter, starting with the maximization problems PP and KP, and addressing later the minimization problems CSP and BPP. [Wäscher, Haußner and Schumann \(2007\)](#) classified 445 papers on cutting and packing problems, 64 of which deal with nesting problems. Of these, 49 papers deal with open dimension problems and 15 with problems with limited-size containers, such as those studied in this chapter. From these 15 papers, 12 deal with maximization problems, one with CSP and two with BPP problems.

Maximization problems aim at maximizing the use of a single container. The difference between the PP and KP maximization problems lies in the diversity of the types of pieces: whereas in the first set of pieces it is weakly heterogeneous, it is strongly heterogeneous in the latter.

[Valle *et al.* \(2012\)](#) propose a metaheuristic, based on a greedy randomized adaptive search procedure, to solve the PP and KP problems with a rectangular container. The authors present a method to address these problems for unlimited piece demand, based on the clustering of pieces into rectangles and on the optimal packing of these rectangles by means of an exact dynamic programming approach. [Dalalah, Khrais and Bataineh \(2014\)](#) propose a constructive and iterative algorithm for the maximization problems, with regular and irregular containers. [Birgin and Lobato \(2010\)](#) deal with the placement of the maximum number of identical rectangles into convex containers.

On the other hand, minimization problems consist in assigning a finite set of pieces to the least possible number of containers. The difference between the CSP and BPP problems lies in the heterogeneity of the respective piece sets, weak in the first and strong in the latter.

The CSP is traditionally solved by column generation, due to the seminal work of Gilmore and Gomory ([Gilmore and Gomory \(1961\)](#) and [Gilmore and Gomory \(1963\)](#)), and the first papers with irregular items, [Valle *et al.* \(2012\)](#) and [Song and Bennell \(2014\)](#) use this technique. To generate the columns in the sub-problem [Valle *et al.* \(2012\)](#) use a two-phase heuristic: the pieces are first clustered in pairs into rectangles, and then the exact dynamic programming approach of [Cintra *et al.* \(2008\)](#) is used to place these clusters into a container; [Song and Bennell \(2014\)](#) use an adaptation of the beam search of [Bennell and Song \(2010\)](#).

As to the BPP problems, in which a strongly heterogeneous set of pieces is to be assigned to multiple containers, the algorithms proposed in the literature are basically hyper-heuristics. The paper [Terashima-Marín *et al.* \(2010\)](#) presents a hyper-heuristic based on a genetic algorithm to solve the regular and irregular BPP. In the regular BPP the pieces are rectangular and in the irregular BPP the pieces are convex. The authors propose a set of 541 instances with convex pieces and use them to test the hyper-heuristic. [Lopez-Camacho *et al.* \(2013\)](#) propose an adaptation of the Djang and Finch heuristic proposed by [Ross *et al.* \(2002\)](#) for the one-dimensional BPP, to

the irregular BPP, and use the instances of [Terashima-Marín et al. \(2010\)](#) to test their approach. [López-Camacho et al. \(2014\)](#) present a hyper-heuristic framework to solve the BPP for the one- and two-dimensional cases with regular and irregular pieces, and propose a set of 480 instances with convex and non-convex pieces. [Terashima-Marín et al. \(2010\)](#), [Lopez-Camacho et al. \(2013\)](#) and [López-Camacho et al. \(2014\)](#) use a set of very similar heuristics, based on constructive heuristics, to solve the problems with irregular pieces. [Martinez-Sykora et al. \(2016\)](#) propose new algorithms for the BPP allowing for free rotation of the pieces. These algorithms are based in five construction algorithms, one MIP model and two local search procedures.

3.3 Geometry

Nesting problems are combinatorial optimization problems with geometrical constraints that must prevent the pieces from overlapping and must enforce them to be positioned inside the containers, i.e., the pieces must totally overlap the container. These geometrical constraints are the major challenges that must be overcome. [Bennell2008397](#) describe several approaches to deal with the geometry of the nesting problems: the raster method, direct trigonometry, the no-fit polygon and its generalization, the Φ -function.

[Toledo2013478](#) propose a combination of the no-fit polygon with the raster method to turn the problems solvable and computationally more efficient, and these ideas were merged in this chapter. The no-fit polygon (NFP) was transformed into a no-fit raster (NFR), that has been represented by a binary matrix. Recently, in [Mundim, Andretta and Queiroz \(2017\)](#) the no-fit raster concept is extended to deal with free form items.

Similarly, the inner-fit polygon (IFP), the locus of the admissible positioning points that guarantee that a piece is inside the board, was transformed into a inner-fit raster (IFR) and represented by a binary matrix. The quality of the solutions obtained for the nesting problems using the rasters is dependent on the discretizations used in the rasters.

In the Subsections [3.3.1](#) and [3.3.2](#) we present respectively the no-fit raster and the inner-fit raster.

3.3.1 No-fit raster (NFR)

In this section we present the no-fit raster (proposed in Chapter 2), represented by a binary matrix, as an efficient strategy for heuristic methods to guarantee that the pieces do not overlap.

The no-fit raster is built on the concept of no-fit polygon. The no-fit polygon between piece A and piece B , NFP_{AB} , is the path drawn by the reference point of piece B when it slides, with fixed orientation, around piece A , while the pieces touch but never overlap each other. Two examples of no-fit polygons are represented in Figure [34](#).

The transformation of the no-fit polygon into a no-fit raster depends on the resolution of the grid g , that we consider to be the same in the horizontal and in the vertical directions ($g = g_x = g_y$). The no-fit raster is represented as a mesh of booleans, with resolution g , over the no-fit polygon. The value of each position in the mesh is 1 for all the points inside the no-fit polygon, corresponding to forbidden positioning points for the orbital polygon, and 0 otherwise (see Figure 34).

MirHassani and Bashirzadeh (2015) propose a GRASP meta heuristic based on the no-fit raster and use the constraints of the dotted-board model defined Toledo2013478 to guarantee that the pieces do not overlap, but it takes up to 300 seconds to solve problems with 4 piece types (SHAPES9 and SHAPES15). This resolution time could be considerably reduced, considering the same solution space, by using directly the information provided by the no-fit raster.

Figure 34 – Examples of no-fit polygons and the corresponding no-fit rasters.

3.3.2 Inner-fit raster (IFR)

The inner-fit polygon of a piece is defined as the region inside the container where the reference point of the piece can be placed, so that the piece is totally inside the container. The transformation of the inner-fit polygon into an inner-fit raster depends on the resolution of the grid g . The inner-fit raster is a mesh of booleans over the inner-fit polygon where the value of a position in the mesh is 1 for all the points inside the inner-fit polygon, corresponding to positioning points such that the piece is totally inside the container, and 0 otherwise. Examples for inner-fit polygons and the corresponding inner-fit rasters are represented in Figure 35.

Figure 35 – Examples for inner-fit polygons and inner-fit rasters.

3.4 Solution method

The Heuristic for Nesting Problems (H4NP) is an iterative algorithm. In each iteration one placement rule, randomly chosen, defines a list of points to be used by a constructive heuristic to generate a new solution that is evaluated and can be discarded or saved. The heuristic returns the best solution obtained.

In Section 3.4.1 we present the constructive heuristic. The placement rules, that define the lists of points, are described in Section 3.4.2. The solution method proposed is shown in Section 3.4.3.

3.4.1 Constructive heuristic

The constructive heuristic proposed in this chapter is inspired on the bottom-left heuristic, where the pieces are placed one by one, in the leftmost lowest position available. In this constructive heuristic, the pieces ($p \in P$) are also allocated one by one, in the first available position $(x, y) \in LP$, where LP is a list of points. In the problems we are dealing with, the containers are

limited, therefore it may not be possible to position all the items in one container, i.e. if there is no available position in LP , the item is not placed in the container. In problems with only one container as the PP and KP, there are items that may not be placed. On the other hand, in multiple container problems as the CSP and BPP all the items must be positioned.

Algorithm 2 presents the framework for the constructive heuristic. In this algorithm, each piece $p \in P$, with a rotation $\theta \in \Theta$, is positioned in a container $c \in C$ in the first admissible positioning point $(x, y) \in LP$. To check if the piece p , with rotation θ , can be allocated in the container c , we use the Inner-fit Raster IFR (Section 3.3.2). Whenever we insert a piece in the solution S , the IFR is updated with the No-fit Raster NFR (Section 3.3.1).

Algorithm 2: Framework

Input: Set of pieces P , set of rotations Θ for pieces in P , No-fit Raster NFR for each pair of pieces in P , set of containers C , Inner-fit Raster IFR for each piece in P , list of points LP .

Output: S .

```

1 for  $p \in P$  do
2   for  $c \in C$  do
3     for  $(x, y) \in LP$  do
4       for  $\theta \in \Theta$  do
5         if  $IFR(p, c, x, y, \theta)$  is feasible then
6            $S \leftarrow S + \{p, c, x, y, \theta\}$ ;
7            $IFR \setminus NFR(p, c, x, y, \theta)$ ;
8           goto 1;
9 return  $S$ ;

```

Note that the framework is flexible enough to solve all the problem variants. In the PP and KP, the set C has only one container. In these problems the objective is to maximize the occupation of the container and there may be some items that can not be placed if, in the list LP , there are no admissible positioning points for any rotation. In the other problems, CSP and BPP, where the objective is to minimize the number of containers used to position all the items, we consider that the quantity of containers available is equal to the number of items.

3.4.2 Placement rules

The placement rules developed were inspired on the bottom left rule. For each placement rule, the path drawn by the sequence of points of the list LP is represented in Figure 36. The six placement rules are the Bottom Left (BL), where the items are positioned leftmost lowest position, the Left Bottom (LB), where the items are positioned in the lowest leftmost position, the Anti Spiral (S^{-1}), where the items are positioned in the first free position following an anti-clockwise spiral beginning at the bottom left corner of the container, the Spiral (S), where the items are positioned in the first free position following a clockwise spiral beginning at the

bottom left corner of the container and the Vertical Zig-Zag (V ZZ) and the Horizontal Zig-Zag (H ZZ) where the items are positioned following a vertical (horizontal) zig-zag path beginning at the bottom left corner of the container.

Figure 36 – The six placement rules used in the H4NP.

3.4.3 H4NP, the Heuristic for Nesting Problems

H4NP, the Heuristic for Nesting Problems proposed in this chapter, manages at random the placement rules presented in Section 3.4.2. The H4NP is general for any placement rule, allowing to apply H4NP in other contexts where other placement rules may be more interesting.

The H4NP is described in detail in Algorithm 3 and is limited to nIt iterations. In each iteration a placement rule is chosen randomly, returning the corresponding list of positioning points LP (line 2) and the pieces are ordered using a random function (line 3). The Algorithm 2 is called with list LP and the random sequence of pieces (line 4). Finally, if the current solution is better than the best solution so far, this solution is updated in line 6. The algorithm is finalized in line 7 if an optimal solution is found.

In this chapter for the problem variants with only one container, a solution is considered optimal if all the pieces are allocated or if the container is 100% occupied. In the problems with multiple containers an optimal solution is a solution where all the pieces fit in one container or where the amount of containers used is equal to $\lceil \frac{\sum_{i=1}^n area_{piece_i}}{area_{container}} \rceil$.

3.5 Computational experiments

This section is organized as follows. In Subsection 3.5.1 all the information needed to replicate the computational experiments is submitted. In the following subsections, the results obtained for each problem variant are presented: Placement Problem (PP) in Subsection 3.5.2,

Algorithm 3: H4NP

Input: Number of iterations nIt , set of placement rules PR , list of points LP_k for each $k \in PR$, set of pieces P , set of rotations Θ for pieces in P , No-fit Raster NFR for each pair of pieces in P , set of containers C , Inner-fit Raster IFR for each piece in P .

Output: S^* .

```

1 for  $i$  from 0 to  $nIt$  do
2    $k \leftarrow \text{rand}(k \in PR)$ ;
3    $P \leftarrow \text{shuffle}(P)$ ;
4    $S \leftarrow \text{framework}(P, \Theta, NFR, C, IFR, LP_k)$ ;
5   if ( $S$  is better than  $S^*$ ) then
6      $S^* \leftarrow S$ ;
7     if ( $S^*$  is optimal) return  $S^*$ ;
8 return  $S^*$ ;

```

Knapsack Problem (KP) in Subsection 3.5.3, Cutting Stock Problem (CSP) in Subsection 3.5.4 and Bin Packing Problem (BPP) in Subsection 3.5.5. Each subsection includes a description of the instances used, followed by an analysis of the computational results.

3.5.1 Parameter tuning

In this section, the computer environment, the methodology adopted to define the number of iterations to solve the various problems, and a selection of the best rules for the instances that were run are presented.

3.5.1.1 Computer and programming language

The computational experiments were performed in an Intel® Core™ i7-2600, 3.40 GHz, 16 GB RAM, using one thread and the Ubuntu 12.04 operating system. To code the algorithms the C programming language was used.

3.5.1.2 Piece types

To reduce computational memory and time, we adopted the concept of piece type, encompassing all pieces with the same shape. Thus, only one IFR and one NFR (see Section 3.3) for each piece type need to be kept in memory.

In the case of the PP and CSP problems, in which the set of the pieces is weakly heterogeneous, introducing the concept of piece type reduces the memory usage and the time needed to update the IFR and the NFR . Conversely, in the KP and BPP problems, with strongly heterogeneous sets of pieces, the adoption of that concept has no impact.

3.5.1.3 *Determining the number of iterations*

In order to determine the number of iterations for each problem, there are two features of the H4NP iteration process which need to be borne in mind.

- As the iterations of the H4NP are independent, increasing the number of iterations of the algorithm increases the probability of obtaining a better solution.
- The time spent in each iteration depends on the number of piece types, on the total number of items, and finally on the number of positioning points in the container, i.e. on the discretization used in the IFR.

Considering the above, we realize that the algorithm could either be limited by the elapsed iteration time or by the number of iterations. In this work the limit is imposed on the number of iterations since a time constraint would favor smaller instances with a shorter time per iteration and therefore a larger number of iterations.

After some preliminary computational experiments, considering that in a worst-case analysis the computational time is to be kept under 300 seconds, the number of iterations was fixed to 1000 for the maximization problems (PP and KP).

Regarding the CSP and BPP minimization problems, the number of iterations was fixed to 10 in order to keep the computational time low. The best results known from the literature for the CSP used, on average, 104 containers. As we fixed the limit to 1000 iterations for the problems with one container, for the problems with around 100 containers we used 10 iterations. As the H4NP is a random method, all the experiments were run 10 times.

3.5.1.4 *Selection of placement rules*

The H4NP is a generic heuristic and may be based on any type of placement rule. The quality of the placement rules depends solely on the instances, some preliminary computational tests allowed us to conclude that some placement rules led to better solutions for the instances found in the literature, and that the placement rules beginning at the bottom left corner did not lead to good solutions for some instances. These two conclusions allowed us to select the three best placement rules, to which we added a new placement rule to diversify the search space.

To compare the placement rules, 1000 random sequences of pieces were generated and used to build the solutions using the various placement rules. All the existing instances for the two variants of the placement problem were used: that with limited (LPP) and unlimited demand (UPP). These two variants were chosen on account of their representativity, i.e. they have characteristics of problems with one (KP, PP) and several containers (CSP, BPP), since the demand of the items is sufficiently large for more than one container to be needed.

In order to find the most appropriate placement rules to be used in the computational tests, the number of best solutions found for the thousand sequences for each problem was then computed for each placement rule. In the case of a tie, we considered all placement rules which produced a best solution. The computational time is not presented since it is similar for all the placement rules. The results obtained are represented in Figure 37.

Figure 37 – a) Number of best solutions for the LPP; (b) Number of best solutions for the UPP; (c) Percentage of best solutions for the PP (LPP+UPP); (d) Relative deviation to the best heuristic for the PP (LPP+UPP).

The general results represented in Figures 37(c) and 37(d) show a clear advantage of the Anti Spiral (S^{-1}), Bottom Left (BL) and Horizontal Zig-Zag (H ZZ) placement rules, wherefore these were chosen for the computational experiments.

In the case of the instance “Bottom Left Fails Puzzle” (BLFP) from Martins and Tsuzuki (2010), an additional placement rule was needed; it can be easily verified that a solution for this instance cannot be reached with any placement rule beginning at the bottom left corner.

To diversify the algorithm we thus added the random corner placement rule, according to which, for each piece to be positioned, the initial positioning corner is chosen randomly and the piece is positioned following the rules of the bottom left heuristic adapted to that random corner.

Figure 38 – Result for the BLFP instance obtained by the random corner placement rule.

3.5.2 Placement Problem

In this section we compare the best results published in the literature for benchmark problems of the Placement Problem with those achieved with the H4NP heuristic. For these problems, a set of weakly heterogeneous pieces are to be placed in a large container, so as to maximize the occupied area. In these experiments, we considered two variants of the Placement Problem, namely the Limited Placement Problem (LPP), in which the demand for each type of piece is limited, and the Unconstrained Placement Problem (UPP), where that demand is unlimited.

3.5.2.1 Instances for the Limited Placement Problem (LPP)

The nineteen instances existing in the literature for the LPP are described in Table 11. The containers used in these instances have different characteristics. The containers in the instances “Puzzle with a Irregular Shaped Container - PISC” and “Example5” are non-convex and in the instance “Container with a Hole Puzzle - CHP” the container has a hole. In all the other instances the containers are rectangular. The number in the column “Rotation step” represents the rotation step, in degrees, allowed for the pieces in the instance, zero meaning that no rotation is allowed.

Most instances from the literature share the characteristic that all the pieces fit inside the container, and pose therefore only a positioning problem, wherefore five new instances, based on benchmark instances from the ESICUP website (<http://www.fe.up.pt/esicup/>) are proposed, in order to add instances which present a selection problem in addition to the positioning problem. These new instances are based on the instances Han, Poly2A, Poly3A, Poly4A e Poly5A, where the container is defined as a square with side length $\lfloor \sqrt{\sum_{i=1}^n area_{piece_i}} \rfloor$, as proposed in Hu *et al.* (2015).

3.5.2.2 Instances for the Unconstrained Placement Problem (UPP)

For the problem with unlimited demand, Valle *et al.* (2012) proposed the fifteen instances presented in Table 12, in all of which the container is rectangular. Thirteen of these fifteen

Table 11 – Instances for the Limited Placement Problem (LPP).

Instance name	Original authors	Piece types	Total pieces	Rotation step	Container width	Container length
LFFP LPP	Martins and Tsuzuki (2010)	2	5	45	400.00	400.00
Tangram LPP	Martins and Tsuzuki (2010)	5	7	45	800.00	800.00
CHP LPP	Martins and Tsuzuki (2010)	5	7	45		
PISC LPP	Martins and Tsuzuki (2010)	6	7	45		
Fu LPP	Valle <i>et al.</i> (2012)	11	12	90	38.00	34.00
Jakobs1 LPP	Valle <i>et al.</i> (2012)	22	25	90	40.00	13.00
Jakobs2 LPP	Valle <i>et al.</i> (2012)	22	25	90	70.00	28.2
Shapes0 LPP	Valle <i>et al.</i> (2012)	4	43	0	40.00	63.00
Shapes1 LPP	Valle <i>et al.</i> (2012)	4	43	180	40.00	59.00
Shapes2 LPP	Valle <i>et al.</i> (2012)	7	28	180	15.00	27.30
Albano LPP	Valle <i>et al.</i> (2012)	8	24	180	4,900.00	10,122.63
Dagli LPP	Valle <i>et al.</i> (2012)	10	30	180	60.00	65.60
Mao LPP	Valle <i>et al.</i> (2012)	9	20	90	2,550.00	2,058.60
Marques LPP	Valle <i>et al.</i> (2012)	8	24	90	104.00	83.60
Shirts LPP	Valle <i>et al.</i> (2012)	8	99	180	40.00	63.13
Swim LPP	Valle <i>et al.</i> (2012)	10	48	180	5,752.00	6,568.00
Trousers LPP	Valle <i>et al.</i> (2012)	17	64	180	79.00	245.75
Example2 LPP	Dalalah, Khrais and Bataineh (2014)	4	13	45	6.00	10.00
Example5 LPP	Dalalah, Khrais and Bataineh (2014)	7	35	45		
Han LPP	New	20	23	90	44.00	44.00
Poly2A LPP	New	15	30	90	28.00	28.00
Poly3A LPP	New	15	45	90	34.00	34.00
Poly4A LPP	New	15	60	90	40.00	40.00
Poly5A LPP	New	15	75	90	45.00	45.00

instances were adapted from those proposed by the same authors for the LPP (Table 11), and two additional instances, Dighe1 and Dighe2, were added. It is however important to highlight that in Valle *et al.* (2012) the value of the length of the container was rounded down in the thirteen instances adapted from the LPP to the UPP.

Table 12 – Instances proposed by Valle *et al.* (2012) for the UPP.

Problem name	Piece types	Rotation step	Container width	Container length
Fu UPP	11	90	38.00	34.00
Jakobs1 UPP	22	90	40.00	13.00
Jakobs2 UPP	22	90	70.00	28.00
Shapes0 UPP	4	0	40.00	63.00
Shapes1 UPP	4	180	40.00	59.00
Shapes2 UPP	7	180	15.00	27.00
Albano UPP	8	180	4,900.00	10,122.00
Dighe1 UPP	16	0	100.00	138.00
Dighe2 UPP	10	0	100.00	134.00
Dagli UPP	10	180	60.00	65.00
Mao UPP	9	90	2,550.00	2,058.00
Marques UPP	8	90	104.00	83.00
Shirts UPP	8	180	40.00	63.00
Swim UPP	10	80	5,752.00	6,568.00

3.5.2.3 *Computational results for the Placement Problem*

In this section, the computational results achieved with the proposed H4NP are shown and compared with those obtained with the best existing algorithms for the Placement Problem. Tables 13 and 14 depict results obtained for the LPP and the UPP instances respectively.

Computational results for the LPP

For the LPP, the results obtained with the H4NP excel or equal those in the literature for all the instances. The results are strictly better for thirteen out of the sixteen benchmark instances. The solutions are optimal for the instances, LFFP|LPP, Mao|LPP and Marques|LPP, because in either case all the pieces are allocated into the container.

In what regards the five new instances, in the instance Han 19 out of the 23 pieces were positioned, in the instance Poly2A 19 out of the 30 pieces, in the instance Poly3A 28 out of the 45 pieces, Poly4A 39 out of the 60 pieces and Poly5A 49 out of the 75 pieces. The average occupancy for the instances Poly2A, Poly3A, Poly4A and Poly5A was between 63.58% and 66.73%, and for the Han instance 76.31%. The best results obtained are shown in Section 3.8 in Figure 42.

Table 13 – Results for the LPP instances

Instance name	Best result published		H4NP		Av. time (s)	Grid
	Occupied area	Time (s)	Occupied area			
			Best	Average		
LFFP LPP ^a	1.0000	4.7	1.0000	1.0000	0.4	1:1
Tangram LPP ^a	0.8670	8.6	1.0000	1.0000	0.5	1:1
CHP LPP			1.0000	1.0000	0.2	1:1
PISC LPP			1.0000	1.0000	0.1	1:1
Fu LPP ^a	0.8382	21.8	0.8382	0.8382	0.6	1:1
Jakobs1 LPP ^a	0.7538	8.3	0.7538	0.7538	0.1	1:1
Jakobs2 LPP ^a	0.6844	565.7	0.6844	0.6844	0.1	1:1
Shapes0 LPP ^a	0.6016	1552.5	0.6218	0.5858	0.1	1:1
Shapes1 LPP ^a	0.6424	3891.6	0.6496	0.6200	0.3	1:1
Shapes2 LPP ^a	0.7289	1048.1	0.7472	0.7067	0.1	1:1
Albano LPP ^a	0.8038	961.6	0.8073	0.7974	144.3	10:1
Dagli LPP ^a	0.7586	1106.4	0.7713	0.7400	0.8	1:1
Mao LPP ^a	0.7142	120.4	0.7142	0.7087	8.7	10:1
Marques LPP ^a	0.8270	217.8	0.8270	0.8069	6.33	1:1
Shirts LPP ^a	0.7702	14317.1	0.8182	0.8075	0.6	1:1
Swim LPP ^a	0.6427	39781.4	0.6610	0.6354	27.5	100:1
Trousers LPP ^a	0.7866	5796.6	0.7882	0.7672	3.7	1:1
Example2 LPP ^b	0.9000	11.5	1.0000	0.9522	0.2	1:1
Example5 LPP ^b	0.8971	31.2	0.9029	0.8617	3.0	1:1
Han LPP			0.7882	0.7631	8.5	1:1
Poly2A LPP			0.6888	0.6655	1.6	1:1
Poly3A LPP			0.6730	0.6673	4.9	1:1
Poly4A LPP			0.6606	0.6358	6.2	1:1
Poly5A LPP			0.7032	0.6605	11.0	1:1

^a Best result published in [Valle et al. \(2012\)](#).

^b Best result published in [Dalalah, Khrais and Bataineh \(2014\)](#).

Computational results for the UPP

As can be seen in Table 14, the H4NP performed well for the UPP, obtaining eight solutions that are better than those published in the literature and two solutions that are equal;

i.e., the H4NP outperformed or equaled the solutions published in the literature in 66.6% of the computational experiments. That said, we wish to emphasize that we added this problem to the computational tests only for the sake of completeness for it is our opinion that this problem is of no relevance to nesting problems, since e.g. some solutions with 100% occupancy are formed only by rectangles.

The best results obtained with the H4NP are shown in Section 3.8 in Figure 41.

Table 14 – Results for the UPP instances.

Instance name	Best result published ^a		H4NP		Av. time (s)	Grid
	Occupied area	Time (s)	Occupied area Best	Average		
Fu UPP	0.9892	5.0	0.9969	0.9152	6.7	1:1
Jakobs1 UPP	1.0000	49.6	1.0000	1.0000	5.5	1:1
Jakobs2 UPP	1.0000	66.1	0.9887	0.9657	18.8	1:1
Shapes0 UPP	0.6190	134.4	0.7806	0.6629	4.2	1:1
Shapes1 UPP	0.6949	248.2	0.7691	0.6801	7.2	1:1
Shapes2 UPP	0.9284	49.4	0.9086	0.8877	1.6	1:1
Dighe1 UPP	0.7631	7.6	0.8031	0.7414	30.5	1:1
Dighe2 UPP	0.7791	3.1	0.8040	0.7655	21.5	1:1
Albano UPP	0.9653	37.9	0.8609	0.8609	261.0	10:1
Dagli UPP	0.9256	51.0	0.9312	0.9291	9.7	1:1
Mao UPP	0.9812	71.5	0.9420	0.9312	87.5	10:1
Marques UPP	0.9606	43.8	0.9671	0.9606	88.3	1:1
Shirts UPP	1.0000	508.9	1.0000	0.9192	5.5	1:1
Swim UPP	0.7590	2206.4	0.8203	0.8138	258.2	100:1
Trousers UPP	1.0000	193.5	0.9914	0.9805	138.0	1:1

^a Best result published in [Valle et al. \(2012\)](#).

3.5.3 Knapsack Problem

In this section, we compare the results achieved with the H4NP with those published in the literature for benchmark instances of the Knapsack Problem. The Knapsack Problem consists in placing a set of different pieces into a large object so as to maximise the occupied area. This problem is still largely unexplored in the area of nesting problems, and only the first seven instances presented in Table 15 were to be found in the literature. The containers are rectangular in all instances, except for “Example4”, in which case the container is irregular.

To the instances existing in the literature we propose the addition of a new set of five instances, namely the last five instances in Table 15, PolyA, Poly2B, Poly3B, Poly4B e Poly5B. As in Section 3.5.2, the containers are squares of size $\lfloor \sqrt{\sum_{i=1}^n area_{piece_i}} \rfloor$.

Computational results for the KP

Table 16 displays the results achieved with the H4NP compared with those obtained with

Table 15 – Description of the KP instances.

Problem name	Original author	Number of pieces	Rotation step	Container width	Container length
Glass1 KP	Fischetti and Luzzi (2009)	5	0	45.00	45.00
Glass2 KP	Fischetti and Luzzi (2009)	7	0	45.00	45.00
Glass3 KP	Fischetti and Luzzi (2009)	9	0	100.00	100.00
Small puzzle KP	Martins and Tsuzuki (2010)	4	45	100.00	100.00
Dighe1 KP	Dighe and Jakiela (1996)	16	0	100.00	100.00
Dighe2 KP	Dighe and Jakiela (1996)	10	0	100.00	100.00
Example4 KP	Dalalah, Khrais and Bataineh (2014)	50	45		
Poly1A KP	New	15	90	20.00	20.00
Poly2B KP	New	30	90	29.00	29.00
Poly3B KP	New	45	90	32.00	32.00
Poly4B KP	New	60	90	39.00	39.00
Poly5B KP	New	75	90	42.00	42.00

the best existing algorithms for the KP. Furthermore, in Section 3.8 the Figure 43 depicts the best results obtained with the H4NP. The H4NP obtained four optimal solutions already reported in the literature, and for the instance “Example4”, which has a non-convex container, the solution was better than the best one so far reported in the literature.

As to the new instances proposed in this work, the number of pieces allocated in the best case is as follows: 10 out of 15 pieces in the instance Poly1A, 19 out of 29 pieces in the instance Poly2B, 28 out of 45 pieces in the instance Poly3B, 44 out of 60 pieces in the instance Poly4B, and 49 out of 75 pieces in the instance Poly5B. For these instances, the average occupancy rate varies between 65.78% and 70.72%.

Table 16 – Results for the KP instances.

Instance name	Best result published		H4NP		Av. time (s)	Grid
	Occupied area	Time (s)	Occupied area			
			Best	Average		
Glass1 KP ^a	1.0000		1.0000	1.0000	4.2	1:1
Glass2 KP ^a	1.0000		1.0000	1.0000	6.3	1:1
Glass3 KP ^b	1.0000		1.0000	1.0000	30.0	1:1
Small puzzle KP ^c	1.0000	7.3	1.0000	1.0000	0.0	1:1
Dighe1 KP ^d	1.0000	83	0.8156	0.7805	15.5	1:1
Dighe2 KP ^d	1.0000	22	0.8858	0.8633	8.4	1:1
Example4 KP ^e	0.9290	31.2	0.9559	0.9436	9.6	1:1
Poly1A KP			0.7225	0.6578	2.4	1:1
Poly2B KP			0.6920	0.6740	6.1	1:1
Poly3B KP			0.7148	0.7072	18.0	1:1
Poly4B KP			0.7068	0.6929	28.6	1:1
Poly5B KP			0.7083	0.6979	47.2	1:1

^a Best result published in [Fischetti and Luzzi \(2009\)](#) using a MIP model.

^b Best result published in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) using a MIP model.

^c Best result published in [Valle et al. \(2012\)](#).

^d Best result published in [Gomes and Oliveira \(2006\)](#).

^e Best result published in [Dalalah, Khrais and Bataineh \(2014\)](#).

3.5.4 Cutting Stock Problem

In this section, we compare the results obtained with the H4NP for benchmark instances with those published in the literature for CSP. For this problem we found twenty nine instances, all with rectangular containers, fifteen in Valle *et al.* (2012) and fourteen in Song and Bennell (2014). The instances are introduced in Table 17. The instances proposed by Valle *et al.* (2012) and Song and Bennell (2014) are based on instances available at the ESICUP website. Valle *et al.* (2012) use rectangular containers with the width of the original instance and the length inspired in Gomes and Oliveira (2006). In the instances proposed in Song and Bennell (2014), the containers are squares with the side equal to the width of the original instances. The demand of the pieces in the instances proposed by Valle *et al.* (2012) is random in the interval $[1, 100]$ (available at the url: <http://www.loco.ic.unicamp.br/instances/irregular2d/>) and the demand of the pieces in the instances proposed by Song and Bennell (2014) is 100 for all pieces.

Computational results for the CSP

Analising the results in Table 18 we conclude that the H4NP is competitive because in 16 out of the 29 instances the results obtained excel the best results published in the literature.

3.5.5 Bin Packing Problem

In this section, we present the instances adopted and the results obtained for the BPP. For the BPP two large sets of instances Terashima 1 and Terashima 2 are available at the ESICUP website. Terashima 1 is a set of 540 instances comprehending convex polygons, and Terashima 2 is a set of 480 instances comprising both convex and non-convex polygons. These instances were artificially created as described in Terashima-Marín *et al.* (2010) and López-Camacho *et al.* (2014).

Both sets, Terashima 1 and Terashima 2 are divided into classes of instances, each constituted by a number of pieces and an optimal number of containers, as presented in Table 19. The only class for which the optimal solution is not known is Conv G. Fifteen containers were used to build this class and then some pieces were removed from the instances, wherefore the optimal number containers is less than or equal to 15. All containers are squares of side 1000.

In addition to these instances, Terashima-Marín *et al.* (2010) propose to solve the Fu instance, available at the ESICUP website, scaled by a factor of 10, using a square container of side 300. The lower limit on the number of containers for this instance is set to two, as computed by dividing the sum of the areas of the pieces by the area of each container.

Computational results for the BPP

When dealing with a large set of instances, the main problem is establishing a fair comparison between results. To the best of our knowledge, no published work proposes a simple

Table 17 – Description of the CSP instances.

Instance name	Original author	Type pieces	Total pieces	Rotation step	Container width	Container length
Fu CSP	Valle <i>et al.</i> (2012)	11	775	90	38.00	34.00
Jakobs1 CSP	Valle <i>et al.</i> (2012)	22	1272	90	40.00	13.00
Jakobs2 CSP	Valle <i>et al.</i> (2012)	22	1070	90	70.00	28.00
Shapes0 CSP	Valle <i>et al.</i> (2012)	4	2025	0	40.00	63.00
Shapes1 CSP	Valle <i>et al.</i> (2012)	4	2132	180	40.00	59.00
Shapes2 CSP	Valle <i>et al.</i> (2012)	7	1699	180	15.00	27.00
Dighe1 CSP	Valle <i>et al.</i> (2012)	16	829	0	100.00	138.00
Dighe2 CSP	Valle <i>et al.</i> (2012)	10	517	0	100.00	134.00
Albano CSP	Valle <i>et al.</i> (2012)	8	1543	180	4900.00	10122.00
Dagli CSP	Valle <i>et al.</i> (2012)	10	1539	180	60.00	65.00
Mao CSP	Valle <i>et al.</i> (2012)	9	934	90	2550.00	2058.00
Marques CSP	Valle <i>et al.</i> (2012)	8	1290	90	104.00	83.00
Shirts CSP	Valle <i>et al.</i> (2012)	8	4441	180	40.00	63.00
Swim CSP	Valle <i>et al.</i> (2012)	10	2481	180	5752.00	6568.00
Trousers CSP	Valle <i>et al.</i> (2012)	17	3111	180	79.00	245.00
FU CSP	Song and Bennell (2014)	11	1200	90	38.00	38.00
JAKOBS1 CSP	Song and Bennell (2014)	22	2500	90	12.00	12.00
SHAPES1 CSP	Song and Bennell (2014)	4	4300	180	40.00	40.00
SHAPES2 CSP	Song and Bennell (2014)	7	2800	180	15.00	15.00
DIGHE1 CSP	Song and Bennell (2014)	16	1600	0	100.00	100.00
DIGHE2 CSP	Song and Bennell (2014)	10	1000	0	100.00	100.00
ALBANO CSP	Song and Bennell (2014)	8	2400	180	4900.00	4900.00
DAGLI CSP	Song and Bennell (2014)	10	3000	180	60.00	60.00
MAO CSP	Song and Bennell (2014)	9	2000	90	2550.00	2550.00
MARQUES CSP	Song and Bennell (2014)	8	2400	90	104.00	104.00
SHIRTS CSP	Song and Bennell (2014)	8	9900	180	40.00	40.00
SWIM CSP	Song and Bennell (2014)	10	4800	180	5752.00	5752.00
TROUSERS CSP	Song and Bennell (2014)	17	6400	180	79.00	79.00
POLY (5B) CSP	Song and Bennell (2014)	25	2500	90	40.00	40.00

way to compare the results obtained. In this chapter, we propose a simple method to compare the results obtained with different algorithms for the Terashima 1 and Terashima 2 classes of instances, i.e. to compare the performance of the algorithms for each class, given that each class has similar characteristics.

Tables 20 and 31 present the results obtained with the H4NP using two grids, namely 10:1 and 1:1, for the instances Terashima 1 and Terashima 2. The tables show that, as could be expected, the results were obtained faster with the grid 10:1, but the solutions are generally poorer.

In what concerns the Fu instance, it was solved with the H4NP in under one second in all ten runs, and an optimal two containers solution (see Figure 39) was achieved at all times.

Fourteen optimal solutions were found with the H4NP using grid 1:1, namely for the

Table 18 – Results for the CSP instances.

Instance name	Best result published Solution	Time (s)	H4NP Solution		Av. time (s)	Grid
			Best	Average		
Fu CSP ^a	74	228.9	70	71.3	26.7	1:1
Jakobs1 CSP ^a	50	6726.7	46	48.4	9.3	1:1
Jakobs2 CSP ^a	47	6897.0	43	45.2	31.9	1:1
Shapes0 CSP ^a	55	26701.1	50	51.6	34.1	1:1
Shapes1 CSP ^a	56	59601.6	49	49.9	59.4	1:1
Shapes2 CSP ^a	63	9413.4	69	70.5	11.2	1:1
Dighe1 CSP ^a	61	252.3	51	52.1	83.7	1:1
Dighe2 CSP ^a	46	45.2	43	44.5	48.4	1:1
Albano CSP ^a	84	6750.2	82	86.2	441.3	10:1
Dagli CSP ^a	58	9304.73	55	56.0	83.3	1:1
Mao CSP ^a	49	7073.2	47	48.0	131.0	10:1
Marques CSP ^a	53	8298.1	52	53.0	204.0	1:1
Shirts CSP ^a	45	1195677.0	47	55.3	46.5	1:1
Swim CSP ^a	60	466819.1	54	54.0	609.0	100:1
Trousers CSP ^a	53	273141.6	50	50.8	467.7	1:1
FU CSP ^b	108	580.0	92	93.8	57.5	1:1
JAKOBS1 CSP ^b	296	20524.0	329	333.0	34.8	1:1
SHAPES1 CSP ^b	140	335.0	160	163.7	205.3	1:1
SHAPES2 CSP ^b	174	476.0	209	210.6	22.4	1:1
DIGHE1 CSP ^b	114	11627.0	152	160.2	262.2	1:1
DIGHE2 CSP ^b	100	2002.0	141	154.0	160.8	1:1
ALBANO CSP ^b	205	2181.0	253	258.1	198.3	10:1
DAGLI CSP ^b	97	74449.0	108	110.9	193.1	1:1
MAO CSP ^b	136	51627.0	102	103.5	545.9	10:1
MARQUES CSP ^b	132	10713.0	101	102.0	646.2	1:1
SHIRTS CSP ^b	155	6983.0	168	168.8	228.4	1:1
SWIM CSP ^b	105	6373.0	121	126.5	5007.6	100:1
TROUSERS CSP ^b	348	17430.0	391	394.1	1611.1	1:1
POLY (5B) CSP ^b	60	152825.0	80	80.2	286.3	1:1

^a Best result published in [Valle et al. \(2012\)](#).^b Best result published in [Song and Bennell \(2014\)](#).

instances TO003C5 and TO009C5 of the NConvO class; for the instances TS001C5, TS009C5, TS012C8, TS029C8 and TS030C8 of the NConvS class; for the instances TT004C10, TT005C10 and TT009C10 of the NConvT class, and for the instances TV008C5, TV009C5, TV018C and TV023C8 of the NConvV class.

3.6 Contributions for the literature

In this section we present the major contributions of this paper.

Table 19 – Description of the Terashima 1 and Terashima 2 classes of instances.

Terashima 1			Terashima 2		
Class	Num. pieces	Optimal num. of containers	Class	Num. pieces	Optimal num. of containers
ConvA	30	3	NConvA	35 - 50	3
ConvB	30	10	NConvB	40 - 52	10
ConvC	36	6	NConvC	42 - 60	6
ConvD	60	3	NConvF	35 - 45	2
ConvE	60	3	NConvH	42 - 60	12
ConvF	30	2	NConvL	35 - 45	3
ConvG	36	≤ 15	NConvM	45 - 58	5
ConvH	36	12	NConvO	33 - 43	7
ConvI	60	3	NConvS	17 - 20	2
ConvJ	60	4	NConvT	30 - 40	10
ConvK	54	6	NConvU	20 - 33	5
ConvL	30	3	NConvV	15 - 18	5
ConvM	40	5	NConvW	24 - 28	4
ConvN	60	2	NConvX	25 - 39	3
ConvO	28	7	NConvY	40 - 50	6
ConvP	56	8	NConvZ	60	12
ConvQ	60	15			
ConvR	54	9

Figure 39 – Instance Fu adapted by [Terashima-Marín et al. \(2010\)](#).

The most significant contribution of this paper is the proposal for a general heuristic for nesting problems with limited-size containers, which improves the solutions for benchmark instances of the Limited Placement Problem (LPP), Unconstrained Placement Problem (UPP), Knapsack Problem (KP) and Cutting Stock Problem (CSP).

The contribution towards each one of these problems and the overview of all contributions are displayed in detail in Figure 40. The number of instances for which the solutions were better, equal, and worse than the best solution published in the literature are presented in pie charts in white, light gray and dark gray respectively.

The solutions achieved with the H4NP equaled or excelled the best solution published in the literature for 51 out of 70 benchmark instances (73%). H4NP does thus provide better results

Table 20 – Results obtained with H4NP, with grids 10:1 and 1:1, for the instances Terashima 1.

Class Name	Optimal num. of containers	Grid 10:1			Grid 1:1		
		Best	Average	Av. time (s)	Best	Average	Av. time (s)
ConvA	3	4	4.41	10.8	4	4.07	185.6
ConvB	10	13	13.59	27.0	11	12.50	280.5
ConvC	6	8	8.83	29.7	7	7.90	312.5
ConvD	3	4	4.86	61.9	4	4.37	603.8
ConvE	3	5	5.78	32.7	5	5.43	714.5
ConvF	2	3	3.06	20.6	3	3.03	171.9
ConvG	≤ 15	15	16.18	38.1	13	15.80	403.6
ConvH	12	16	16.05	48.8	13	15.07	389.7
ConvI	3	4	4.00	44.7	4	4.00	510.5
ConvJ	4	5	5.22	55.1	5	5.03	613.1
ConvK	6	8	8.45	35.5	7	7.97	380.4
ConvL	3	5	5.10	23.2	4	4.83	226.9
ConvM	5	7	7.88	33.0	6	7.53	392.4
ConvN	2	3	3.00	48.7	3	3.00	516.5
ConvO	7	9	10.14	23.3	8	8.73	245.2
ConvP	8	11	11.82	75.9	10	11.13	978.6
ConvQ	15	20	20.41	96.4	18	18.90	1227.0
ConvR	9	12	12.21	68.2	11	11.47	754.3

Table 21 – Results obtained with H4NP, with grids 10:1 and 1:1, for the instances Terashima 2.

Class Name	Optimal num. of containers	Grid 10:1			Grid 1:1		
		Best	Average	Time (s)	Best	Average	Av. time (s)
NConvA	3	4	4.61	22.9	4	4.17	307.8
NConvB	10	13	13.59	42.5	12	13.03	627.7
NConvC	6	8	8.83	38.1	7	8.17	536.8
NConvF	2	3	3.16	18.7	3	3.03	249.1
NConvH	12	15	16.15	47.4	13	15.53	703.4
NConvL	3	5	5.21	22.4	4	4.97	310.1
NConvM	5	7	7.99	38.1	7	7.63	535.1
NConvO	7	9	10.15	27.2	7	8.90	395.7
NConvS	2	4	3.75	5.7	2	3.20	84.1
NConvT	10	12	12.54	26.1	10	11.67	396.3
NConvU	5	7	7.99	10.2	6	7.37	198.3
NConvV	5	7	7.00	6.7	5	6.27	86.1
NConvW	4	5	5.96	12.8	5	5.47	161.3
NConvX	3	4	4.92	12.1	4	4.53	210.3
NConvY	6	8	8.45	34.9	7	8.13	472.8
NConvZ	12	14	14.77	90.4	13	14.70	983.8

than the algorithms published in the literature for a wide variety of problems.

Another contribution of this paper is the proposal of two new sets of instances, one for the Placement Problem and one for the Knapsack Problem. The proposed instances are more difficult than the existent ones, and can be used in future papers to compare algorithms. The proposed instances were adapted from a known set of instances available on the ESICUP website by modifying the width and length size of the container so that, the sum of the areas of the

Figure 40 – H4NP obtained equal or better solutions in 73% of the benchmark instances.

items being thus larger than the area of the container, a selection problem adds to a positioning problem.

The BPP differs from all other nesting problems in that it has many benchmark instances, such as those recently introduced by [Terashima-Marín *et al.* \(2010\)](#) and [López-Camacho *et al.* \(2014\)](#). As the set of instances for the BPP is very large, 1020 instances, an instance by instance comparison, as has been done for the other problems, is not possible. An alternative would be to report the total amount of optimal solutions, as is usually done in other papers, but this approach does not give insights on which types of instances the proposed algorithm solves better or worse. Accordingly, a comparison via classes of instances with similar characteristics is proposed in this chapter.

3.7 Conclusions

This paper presents the H4NP, a new general heuristic for two-dimensional nesting problems with limited-size containers, which makes random use of a set of placement rules. To deal with the geometry of the problems, the concepts of no-fit raster and the inner-fit raster were adopted.

The computational experiments show that the H4NP improved on the best solutions available in the literature for three problems: the Placement, the Knapsack, and the Cutting Stock Problems. For these, better or equal solutions were obtained with the H4NP for 51 of 70 instances. Moreover, we proposed 5 new instances for the Placement Problem and another 5 instances for the Knapsack Problem. As to the Bin Packing Problem, the H4NP was run with 1020 instances, and a simple comparison method via classes of instances was proposed. The

computational results for all 1100 instances indicate that the H4NP is highly competitive with state of the art algorithms, excelling some algorithms specifically designed for each problem and, improving on a set of best solutions for well known benchmark instances.

The results obtained with the H4NP show that working with different placement rules can be a very good alternative to solve cutting and packing problems. In the future, our focus will be primarily the specialization of the H4NP for the resolution of others problems by proposing new placement rules.

3.8 The draw of the solutions

Figure 41 – Best solutions computed with the H4NP for the UPP.

Figure 42 – Best solutions computed with H4NP for the LPP.

Figure 43 – Best solutions for the KP computed with the H4NP.

THE EXPLICIT ENUMERATION FOR IRREGULAR CUTTING STOCK PROBLEM UNDER DETERMINISTIC AND STOCHASTIC DEMAND

The cutting stock problem is present in many sectors and industries. In the 1960's it was proposed for the 1D case a revolutionary method called column generation, that obtained extraordinary results. The results were so interesting that, until recently, several papers for 1D and 2D rectangular (no guillotine, guillotine and guillotine in n -stages) cutting stock problem were based on this approach. For this reason, the first two papers dealing with the irregular cutting stock problem proposed column generation algorithms for the problem. In this chapter, we show that for the irregular cutting stock problem the explicit enumeration of patterns can be more efficient than column generation - improving the results for 18 out of 29 literature benchmark instances. This paradigm break generates new horizons, being a reference for the next studies. Finally, we extend the idea presented for a two-stage programming model and obtain results for the stochastic optimization problem, where the demand for each item is unknown.

4.1 Introduction

In the cutting stock problem a set of small items (each with demand greater than one) must be allocated in a subset of larger objects (identical or heterogeneous) in order to minimize the area of the large objects used. For identical larger objects this goal is equivalent to minimizing the number of larger objects used. This problem is found in the paper, wood, glass and ceramics industries, among others. This problem was proposed in 1939 by L. V. Kantorovich (the original article, written in Russian, has been translated into English: [Kantorovich \(1960\)](#)).

In the early 1960's Gilmore and Gomory's model ([Gilmore and Gomory \(1961\)](#)) and

[Gilmore and Gomory \(1963\)](#)) proposed the generation of columns for the one-dimensional case, based on [Dantzig and Wolfe \(1960\)](#). This method starts with a problem relaxation and, using an initial model, finds in each iteration a new column to be inserted in the master problem. The columns represent the cutting patterns of the problem, i.e., a feasible solution for the cutting problem, which are obtained by solving the subproblem for one larger object. Years later, the same authors, in [Gilmore and Gomory \(1965\)](#), emphasize that the efficiency obtained in the one-dimensional case (due to the subproblem being solved through a knapsack problem) is not present in the two-dimensional case. In this chapter, the authors present a method for the rectangular two-dimensional case with multi-stages.

The difficulty in solving cutting problems, especially for the two-dimensional case, motivated the development of new techniques. [Barnhart *et al.* \(1998\)](#) proposed a method called branch and price, which combines the branch and bound method (proposed by [Land and Doig \(1960\)](#)) with the column generation from Gilmore and Gomory. Other authors seek to improve column generation by accelerating convergence ([Alves and Carvalho \(2007\)](#)).

[Alvarez-Valdes, Parajon and Tamarit \(2013\)](#) solved the problem of two-dimensional cutting with guillotine. To do this, the authors used the column generation scheme from Gilmore and Gomory ([Gilmore and Gomory \(1961\)](#) and [Gilmore and Gomory \(1963\)](#)) and to solve the subproblem they used dynamic programming (classic idea for solving the knapsack problem) and three heuristic methods. [Vanderbeck \(2001\)](#) presented a method of column generation for the two-dimensional rectangular cutting problem, with three stages of orthogonal guillotine cutting.

[Cintra *et al.* \(2008\)](#) is a complete study about two-dimensional guillotine cutting stock problems and variants in which orthogonal rotations are allowed, which proposes some dynamic programming algorithms and a column generation method.

Other studies that use column generation are: [Furini *et al.* \(2012\)](#) (two-dimensional two-staged guillotine with multiple stock size), [Nonas and Thorstenson \(2008\)](#) (integrated with lot-size). A framework for meta-heuristic and column generation ([Alvelos, Sousa and Santos \(2013\)](#)), a strategy that combines the cutting plans from Chvatal–Gomory ([Chvátal \(CHVATAL1973305\)](#)) and column generation is generalized to the 1D multiple stock lengths cutting stock problem ([Belov and Scheithauer \(2002\)](#)). [Lübbecke and Desrosiers \(2005\)](#) presents an interesting review of papers that use column generation in more than 20 problems, such as vehicle routing, scheduling and graph coloring.

We can characterize the success of column generation in the cutting problems (1D, guillotine 2D and irregular 2D) by how easily the subproblem is solved. There is a gap between the results for the one-dimensional and the two-dimensional studies of regular objects (rectangles, circles, triangles, etc.), but if we consider irregular items, this gap is much larger.

As far as we know, the first two papers dealing with irregular cutting stock problem, where the items are convex and non-convex polygons, are: [Valle *et al.* \(2012\)](#) and [Song and](#)

[Bennell \(2014\)](#). In these papers, the authors proposed methods based on column generation.

In an earlier study, we proposed H4NP (see Chapter 3), the first constructive heuristic for the cutting stock problem. This method is a constructive heuristic that manages a set of positioning rules to obtain the best possible solution for all cutting and packing problems with limited containers. The main contribution of H4NP is the fact that it can be used for various problems: knapsack problem, placement problem, cutting stock problem and bin packing problem. Due to the generality of the method, to achieve good results in all problems, we did not explore each problem's characteristics to extract the best of the method.

Despite several attempts, the methods in the literature tend to go back to column generation. Is presenting a method based on column generation, especially in more complex problems, just an attempt to make the paper more elegant and publishable? Should we not question if in practice - where the most complex problems are - column generation is the best strategy? In this chapter, we show that the explicit enumeration is better, in several aspects, than the column generation methods proposed for irregular cutting stock problem.

The method proposed in this chapter is based on the explicit generation of patterns. We generate a large number of patterns and solve a mathematical programming model to find the best solution.

The proposed strategy is innovative because it shows that, for the irregular cutting stock problem, column generation is not the most interesting strategy, since it obtains much inferior results than the explicit generation. In the computational experiments, we show that the method proposed improves the results for a significant set of instances of the literature that use column generation. In the second part of this chapter, we present the main advantages of using the proposed method to solve a problem in the industry where the items demand is usually uncertain.

This chapter is organized as follows. In Section 4.2 we present how patterns are generated using H4NP. The generated patterns are used in the deterministic problem, where the items demands are known, and in the stochastic problem. The Section 4.3 presents the MIP models for find the best solution of the set of patterns is present. The two staged model for stochastic problem is proposed in Section 4.4. In Section 4.5 we show the framework with the solution method. The computational experiments are described in Section 4.6. Finally, in Section 4.7 we present the conclusions and future directions.

4.2 Pattern generation

For pattern generation using the H4NP, consider that we have only one bin. For this, it is enough to generate a limited number of random sequences and generate patterns with all the rules of positioning proposed in Chapter 3, that are presented in Figure 44.

Figure 44 – The placement rules used in the H4NP extracted from Chapter 3 .

The adapted version of H4NP for one bin is described in Algorithm 4, limited to nIt iterations. In each iteration, for each list of placement rules (shown in Figure 44), the items list I is ordered using a random function (line 3). Then, in line 4, solution S is built using the ordered items and placing them in bin B via placement rule j (for all the details see Chapter 3). Finally, the current solution is saved in a list of patterns in line 6.

Algorithm 4: H4NP adapted for one bin

Input: Number of iterations nIt , set of placement rules PR , set of items I and bin B .

Output: *Patterns*.

```

1 for  $i$  from 0 to  $nIt$  do
2   for  $j$  from 0 to  $PR$  do
3      $I \leftarrow \text{shuffle}(I)$ ;
4      $S \leftarrow \text{placement\_rule}_j(I, B)$ ;
5      $Patterns \leftarrow Patterns + S$ ;
6 return  $Patterns$ ;

```

4.3 MIP models for a set of patterns

To find the best solution for the cutting stock problem with a set of known patterns, we present in this section two MIP models: in Section 4.3.1 we show the model that determines a sub-set of patterns that produces either the required quantity of items or more than necessary and in Section 4.3.2 we present one model that produces exactly the demand of each item.

4.3.1 Simple MIP model for given patterns in cutting stock problem

For the deterministic case, we inserted the solutions generated by Algorithm 4 as columns in the a simple MIP model for cutting stock problem (called Simple Model or SM). This model can be found in papers from [Gilmore and Gomory \(1961\)](#) and [Gilmore and Gomory \(1963\)](#) to determine the best solution of the columns generated during the method column generation.

In this model we have:

- a set J of patterns;
- a set I of items.

The decision variables are

X_j = number of times the pattern j is used.

The constant related to generated patterns is given by

a_{ij} = number of items i present in pattern j .

The complete model is

$$z^{SM} = \text{minimize} \quad \sum_{j \in J} X_j \quad (4.1)$$

$$\text{subject to:} \quad \sum_{j \in J} a_{ij} X_j \geq d_i, \quad \forall i \in I, \quad (4.2)$$

$$X_j \in \mathbb{Z}^+, \quad \forall j \in J. \quad (4.3)$$

The objective (4.1) of the SM is to minimize the total number of patterns, subject to satisfying the items demand. Note that the demand constraints (4.2) state that the amount of item i used in the solution is greater than or equal to the demand of item i for all i . When solving problem (4.1)-(4.3) with equality constraints, there may not be a feasible solution. And even if it exists, it may not be the optimal solution to all the patterns. Thus, we solve the model using inequalities and then remove the items for which the demand given by the solution are extrapolated.

4.3.2 General MIP model for given patterns in cutting stock problem

Now we propose an adaptation of the SM to force the equality in Constraints (4.2). In this adaptation, called General Model or GM, we insert a new set of decision variables to ensure

that the exact demand of the items are cut. This can be done because they can remove items from the given patterns. This adaptation ensures that we do not lose the optimal solution of the SM and do not have an unfeasible search space.

Consider that we have a pattern (Figure 45 (a)) and the optimal solution for the SM with the Constraints (4.2) satisfied by equality. We need a subset of this pattern, for example, a pattern without both triangles (Figure 45 (b)) or a pattern without both rectangles (Figure 45 (c)). We define each of these subsets t of the pattern as a configuration t .

Figure 45 – Configuration examples.

The new set of variable indices used in the model to represent the different configurations is given by set T . The new decision variables are:

X_{jt} = number of times pattern j is used in configuration t ,

Z_{ijt} = number of items i removed from the pattern j in configuration t .

The complete General Model (GM) is given by

$$z^{GM} = \text{minimize} \quad \sum_{j \in J} \sum_{t \in T} X_{jt} \quad (4.4)$$

$$\text{subject to:} \quad \sum_{j \in J} \sum_{t \in T} (a_{ij} X_{jt} - Z_{ijt}) = d_i, \quad \forall i \in I, \quad (4.5)$$

$$a_{ij} X_{jt} - Z_{ijt} \geq 0, \quad \forall i \in I; \forall j \in J; \forall t \in T, \quad (4.6)$$

$$X_{jt} \in \mathbb{Z}^+, \quad \forall j \in J; \forall t \in T, \quad (4.7)$$

$$Z_{ijt} \in \mathbb{Z}^+, \quad \forall i \in I; \forall j \in J; \forall t \in T. \quad (4.8)$$

The objective (4.4) is to minimize the total number of patterns j in all configuration t , subject to satisfying the items demand. Constraints (4.5) ensure that the items demands are cut exactly. Constraints (4.6) ensure that we do not take from pattern j more items of type i than it has.

4.4 Two-stage MIP model for stochastic demand

Now consider that we can not determine the exact demand for items to solve the cutting stock problem. However, we can establish a finite set of scenarios $\Omega = \{1, \dots, S\}$ with their associated probabilities $\rho = \{\rho_1, \dots, \rho_s\}$, where $\sum_{s \in \Omega} \rho_s = 1$. We define a scenario as a demand vector with probability to happen ρ strictly greater than zero.

Consider the following first-stage variables, the same of the GM in Section 4.3.2, and Constraints (4.4) to (4.8).

The set of scenarios is given by S . Next, we present the parameters of the second stage:

- y_{is}^+ = number of times that item i was produced in excess in scenario s ,
- y_{is}^- = number of times that item i was not produced in scenario s ,
- ρ_s = probability of scenario s happening,
- d_{is} = demand of item i in scenario s ,
- δ_i^+ = penalty to produce more item i than necessary,
- δ_i^- = penalty to produce less item i than necessary.

The complete two-stage model (called SD-2SM, over Stochastic Demand - Two-Stage Model) is given by

$$z^{SD-2SM} = \text{minimize} \quad \sum_{j \in J} \sum_{t \in T} X_{jt} + \sum_{i \in I} \sum_{s \in S} \rho_s (\delta_i^+ y_{is}^+ + \delta_i^- y_{is}^-) \quad (4.9)$$

$$\text{subject to:} \quad \sum_{j \in J} \sum_{t \in T} (a_{ij} X_{jt} - Z_{ijt}) + y_{is}^+ - y_{is}^- = d_{is}, \quad \forall i \in I, s \in S, \quad (4.10)$$

$$a_{ij} X_{jt} - Z_{ijt} \geq 0, \quad \forall i \in I; \forall j \in J; \forall t \in T, \quad (4.11)$$

$$X_{jt}, Z_{jit}, y_{is}^+, y_{is}^- \in \mathbb{Z}^+, \quad \forall j \in J, i \in I, s \in S, t \in T. \quad (4.12)$$

The objective (4.9) is to minimize the total expected cost incurred in both stages, reduce the number of patterns and the penalties for the number of times that items were not produced in the necessary amount. Constraints (4.10) guarantee that the demand for each item i in scenario s is satisfied by the sum of the items produced $a_{ij} X_{jt}$ and removed Z_{ijt} from the patterns, and produced more y_{is}^+ or less y_{is}^- than necessary. Constraints (4.11) limit the number of items i

removed from pattern j in configuration t to the number of items i present in pattern j . All variables are integers and positive (Constraints (4.12)).

In the presented two-stage model (4.9)-(4.12), we can insert the following valid inequality:

- we can ensure that configuration $ij(t+1)$ can only be used if configuration ijt is used:

$$Z_{ijt} \geq Z_{ij(t+1)}, \quad \forall i \in I; \forall j \in J; \forall t \in T; \quad (4.13)$$

- in an analogous way, we can guarantee that there is only one configuration $j(t+1)$ for an X pattern if jt is active:

$$X_{jt} \geq X_{j(t+1)}, \quad \forall i \in I; \forall j \in J; \forall t \in T. \quad (4.14)$$

4.5 The solution method: Explicit Enumeration of Patterns

In this section we summarize the solution method in Algorithm 5. For this we need Algorithm 4 presented in Section 4.2 and the models presented in Section 4.3 and Section 4.4. In Algorithm 5 we have the same inputs of the Algorithm 4: number of iterations or patterns nIt , set of placement rules PR , set of items I and a bin B . In addition, we have the set of scenarios S that can be equal to one, when the problem is deterministic, or greater than one, when the problem is stochastic. We must emphasize that for the deterministic problem, we use the SM to find the best solution for the generated patterns because its relaxation is better than the relaxation of the GM.

In line 2 the list of patterns receives the one generated by Algorithm 4. In line 3 we define a solution variable to save the value of the MIP model that will be used. The conditional in line 4 checks if the problem is either deterministic or stochastic. If the problem has deterministic demands, solution receives the value of the SM solution (see Section 4.3.1) in line 5. The excess items of the deterministic solution, which exceed the demand are taken on line 6. If the problem has stochastic demand, in line 8 the solution receives the value of the SD-2SM solution (see

Section 4.4).

Algorithm 5: Explicit Enumeration of Patterns - EEP

Input: Number of iterations nIt , set of placement rules PR , set of items I , bin B and set of scenarios S .

Output: Solution.

```

1 Create a empty list of patterns PG;
2 PG  $\leftarrow$  Algorithm 4( $nIt, PR, I, B, S$ );
3 Create a empty Solution;
4 if  $|S| = 1$  then
5   Build SM using the patterns list PG and the items demands of scenario S;
6   Solve SM and store the solution found in Solution;
7   Update Solution, removing items for which the demand was exceeded;
8 else
9   Build SD-2SM using the patterns list PG and the items demands of scenarios S;
10  Solve SD-2SM and store the solution found in Solution;
11 return Solution;
```

4.6 Computational experiments

The instances used in the literature for the irregular cutting stock problem are based on the irregular strip packing problem (available at ESICUP ¹). In Song and Bennell (2014) these instances had their bin length fixed and the original items demands were multiplied by 100. In Valle *et al.* (2012) the bin length was also fixed and the items demands were multiplied by a random number between 1 and 100. In Section 4.6.1 we present the results for the instances proposed by Song and Bennell (2014). Then, in Section 4.6.2 we present the results for the instances proposed by Valle *et al.* (2012). To compare the proposed strategy EEP with the methods proposed in Song and Bennell (2014) and Valle *et al.* (2012), 100000 random patterns were generated (line 2 of EEP). In Sections 4.6.1 and 4.6.2, we also present the time spent in generating these patterns. In Section 4.6.3, we present how the proposed method can be used in practice, where demand is a stochastic value.

The computational experiments were performed on a Intel[®] CoreTM i7-2600, 3.40 GHz, 16 GB RAM and operational system Ubuntu 12.04. The pattern generation algorithm (Algorithm 4 - line 2 of EEP) was coded in C language and the MIP models (SM and SD-2SM) were coded in C++ programming language in the IBM ILOG CPLEX 12.7 solver with a 3600 seconds time limit.

¹ <http://www.fe.up.pt/esicup/>

4.6.1 Instances with proportional demand from [Song and Bennell \(2014\)](#)

The main feature of [Song and Bennell \(2014\)](#) instances was to multiply the demands of all items by 100 units. For example, Shapes0 and Shapes1 instances have 4 item types, with original demand $D_i = \{15, 7, 9, 12\}$ and total demand of 43. After the adaptation of [Song and Bennell \(2014\)](#), the demands become $D_i = \{1500, 700, 900, 1200\}$ and the total demand is 4300. Table 22 presents in detail all 14 instances proposed by the authors. The Poly (5B) instance is the only one that needs an additional explanation: for this instance, the authors consider only the first 25 items of the original Poly 5B instance.

Table 22 – Instances data from [Song and Bennell \(2014\)](#).

Problem name	Number items	Total items	Rotacional constraints	Container width	Container length
FU	12	1200	0, 90, 180, 270	38	38
JAKOBS1	25	2500	0, 90, 180, 270	12	12
SHAPES1	43	4300	0, 180	40	40
SHAPES2	28	2800	0, 180	15	15
DIGHE1	16	1600	0	100	100
DIGHE2	10	1000	0	100	100
ALBANO	24	2400	0, 180	4900	4900
DAGLI	30	3000	0, 180	60	60
MAO	20	2000	0, 90, 180, 270	2550	2550
MARQUES	24	2400	0, 90, 180, 270	104	104
SHIRTS	99	9900	0, 180	40	40
SWIM	48	4800	0, 180	5752	5752
TROUSERS	64	6400	0, 180	79	79
POLY (5B)	25	2500	0, 90, 180, 270	40	40

Table 23 presents the best results from [Bennell and Song \(2010\)](#) for the Column Generation (CG), Adapted Column Generation (ACG) and Sequential Heuristic Procedure (SHP). The solutions shown reflects the total number of containers used. For the proposed approach EEP, we present the solution and computational time (in seconds), which is divided into two parts: the time for the pattern generation (PG - line 2 of EEP) and the time to solve the Simple Model (SM - line 6 of EEP) with given patterns in cutting stock problem.

In Table 23 we can see that CG is the method that finds the higher number of best solution of all these methods. However, when EEP finds a worse solution than the best one found by the methods from [Bennell and Song \(2010\)](#) (CG/ACG/SHP), the difference between them is small (at most 20% in DIGHE2). On the other hand, when EEP finds the best solution among all these methods, the difference between its solution and the best one found by CG, ACG and SHP is much bigger (26, 63 and 55 bins for FU, MAO and MARQUES instances).

Table 23 – Comparison between EEP and the methods from Song and Bennell (2014).

Problem name	CG		ACG		SHP		EEP		
	Solution	Time (s)	Solution	Time (s)	Solution	Time (s)	Solution	Time (s) PG SM	
FU	108	580	119	1164	118	4	82	74.1	3.3
JAKOBS1	296	20524	327	7923	314	31	300	23.7	38.9
SHAPES1	140	355	141	2112	144	149	143	20.9	0.3
SHAPES2	174	476	186	430	186	23	182	4.4	0.4
DIGHE1	114	11627	124	9965	118	42	126	184.2	4.6
DIGHE2	108	2002	115	2343	100	2	120	96.7	0.5
ALBANO	207	2181	207	1507	205	30	209	1065.3	1.3
DAGLI	97	74449	98	34334	100	312	102	99.7	5.4
MAO	136	61627	143	1020	139	50	73	3812.8	3.2
MARQUES	132	10713	200	11229	138	721	77	430.2	2
SHIRTS	155	6983	157	7309	156	1029	168	30.9	1.6
SWIM	106	6373	109	28949	105	1145	113	873.2	3.1
TROUSERS	348	17430	353	1499	350	37	357	86.3	0.1
POLY(5B)	60	152825	71	79158	63	2169	70	65	7.3

Table 24 presents the results obtained by H4NP and EEP for the same set of instances. The columns present the same data as Table 23. We can see that the solution quality of the new approach EEP compared to H4NP never obtained a worse solution, being better for 13 out of 14 instances. In terms of CPU time, the sum of CPU time spent by PG and SM for EEP is lower than the CPU time spent by H4NP for 10 out of 14 instances. It is worth noting that on the results of H4NP are the best result of run 10 times from Chapter 3 and the time is the average time of the method.

Table 24 – Comparison between H4NP and EEP.

Problem name	H4NP		EEP			
	Solution	Time (s)	Solution	Time (s) PG SM		
FU	92	57.5	82	74.1	3.3	
JAKOBS1	329	34.8	300	23.7	38.9	
SHAPES1	160	205.3	143	20.9	0.3	
SHAPES2	209	22.4	182	4.4	0.4	
DIGHE1	152	262.2	126	184.2	4.6	
DIGHE2	141	160.8	120	96.7	0.5	
ALBANO	253	198.3	209	1065.3	1.3	
DAGLI	108	193.1	102	99.7	5.4	
MAO	102	545.9	73	3812.8	3.2	
MARQUES	101	646.2	77	430.2	2.0	
SHIRTS	168	228.4	168	30.9	1.6	
SWIM	121	5007.6	113	873.2	3.1	
TROUSERS	391	1611.1	357	86.3	0.1	
POLY(5B)	80	286.3	70	65.0	7.3	

4.6.2 Instances with deterministic demand from [Valle et al. \(2012\)](#)

In [Valle et al. \(2012\)](#) the authors simulate a random demand between 1 and 100 units for each item of the original instance. For example, instance Shapes0 have 4 item types, with original demand $D_i = \{15, 7, 9, 12\}$ and total demand of 43. For each item, a number between 1 and 100 was generated, and after the adaptation of [Valle et al. \(2012\)](#) the new demands are $D_i = \{719, 286, 439, 581\}$ and total demand for 2025. Table 25 presents in detail all 15 instances proposed by the authors. All the details of the instances of [Valle et al. \(2012\)](#), with the demands of each item and the no-fit polygon used, can be found in <http://www.loco.ic.unicamp.br/instances/irregular2d/>.

Table 25 – Instances data from [Valle et al. \(2012\)](#).

Problem name	Type items	Total items	Rotacional constraints	Container width	Container length
Fu	12	775	0, 90, 180, 270	38	34
Jakobs1	25	1272	0, 90, 180, 270	40	13
Jakobs2	25	1070	0, 90, 180, 270	70	28
Shapes0	43	2025	0	40	63
Shapes1	43	2132	0, 180	40	59
Shapes2	28	1699	0, 180	15	27
Dighe1	16	829	0	100	138
Dighe2	10	517	0	100	134
Albano	24	1543	0, 180	4900	10122
Dagli	30	1539	0, 180	60	65
Mao	20	934	0, 90, 180, 270	2550	2058
Marques	24	1290	0, 90, 180, 270	104	83
Shirts	99	4441	0, 180	40	63
Swim	48	2481	0, 180	5752	6568
Trousers	64	3111	0, 180	79	245

In Table 26, we report the solution from literature for the CGH from [Valle et al. \(2012\)](#), H4NP from Chapter 3 and EEP applied to this set of instances. For each algorithm we show the number of bins and the CPU time in seconds used to solve each instance. For the H4NP we presented the best result of the H4NP and the time is the average time of the method (for more details see Chapter 3). As the time of the EEP is divided into patterns generation (PG) and finding the best solution using a MIP model (SM), we present these two amounts separately.

The results of EEP, compared to CGH and H4NP, were better for all 15 instances. In terms of CPU time, both H4NP and EEP are much faster than CGH (even taking into account that these experiments were performed in different computers). For example, CGH took approximately 130 hours to solve instance Swim, while both H4NP and EEP do not exceed one hour for any instance.

Table 26 – Comparison between EEP, CGH from [Valle et al. \(2012\)](#) and H4NP from Chapter 3 .

Problem name	CGH		H4NP		EEP		
	Solution	Time (s)	Solution	Time (s)	Solution	Time (s)	
						PG	SM
Fu	74	228.9	70	26.7	62	64.5	2.8
Jakobs1	50	6726.7	46	9.3	45	93.9	222.2
Jakobs2	47	6897	43	31.9	41	224.5	68.1
Shapes0	55	26701.1	50	34.1	46	19.9	0.4
Shapes1	56	59601.6	49	59.4	44	38.2	0.4
Shapes2	63	9413.4	69	11.2	64	9.1	1.6
Dighe1	61	252.3	51	83.7	46	260.3	1.9
Dighe2	46	45.2	43	48.4	37	136.3	0.8
Albano	84	6750.2	82	441.3	77	2312.8	1.8
Dagli	58	9304.73	55	83.3	52	107.9	2.8
Mao	49	7073.2	47	131	43	1410.8	1.6
Marques	53	8298.1	52	204	50	330.8	2.8
Shirts	45	1195677	47	46.5	42	46.2	3.2
Swim	60	466819.1	54	609	50	2827.2	1.7
Trousers	53	273141.6	50	467.7	52	231.8	2.5

4.6.3 Instances with stochastic demand

Since we are presenting the first algorithm of the literature to consider the irregular cutting stock problem with stochastic demand, we took inspiration from the paper from [Alem et al. \(2010\)](#), which considers this problem for the one-dimensional case. To show the quality of the proposed method, we used two well-known measures of uncertainty effects in stochastic programming, to know: the expected value of perfect information (Section 4.6.3.1) and the value of stochastic solution (Section 4.6.3.2). Finally, with this measures, in Section 4.6.3.3 we present the results produced by EEP for solving instances with stochastic demand.

4.6.3.1 The expected value of perfect information

The maximum value that can be payed in order to have perfect information is the expected value of perfect information (*EVPI*). The *EVPI* is calculated by the difference between the value of the stochastic problem (here the optimal solution of the two-stage model: z^{2SM}) and the value of the solutions wait-and-see (*ws*), i.e. $EVPI = z^{2SM} - ws$. [Birge and Louveaux \(1997\)](#) state that the *EVPI* measures the maximum amount a decision maker would be ready to pay in return for complete (and accurate) information about the future.

For the irregular cutting stock problem under stochastic demand studied in this chapter, the *EVPI* is divide into 3 steps, defined below:

1. In this part we consider the model presented in Section 4.4 with one fixed scenario s . For

this adaptation, we calculate the optimal solutions for all scenarios s given by Z_s^{ws} :

$$Z_s^{ws} = \text{minimize} \quad \sum_{j \in J} \sum_{t \in T} X_{jt} + \sum_{i \in I} \rho_s (\delta_i^+ y_{is}^+ + \delta_i^- y_{is}^-) \quad (4.15)$$

$$\text{subject to:} \quad \sum_{j \in J} \sum_{t \in T} (a_{ij} X_{jt} - Z_{ijt}) + y_{is}^+ - y_{is}^- = d_{is}, \quad \forall i \in I, \quad (4.16)$$

$$a_{ij} X_{jt} - Z_{ijt} \geq 0, \quad \forall i \in I, \forall j \in J, \forall t \in T, \quad (4.17)$$

$$X_{jt}, Z_{jit}, y_{is}^+, y_{is}^- \in \mathbb{Z}^+, \quad \forall j \in J, i \in I, t \in T. \quad (4.18)$$

2. After solving the model present in Step 1 for all scenarios, we calculate the value of the objective function wait-and-see (called ws) assuming the probability of each scenario:

$$ws = \sum_{s \in S} \rho_s Z_s^{ws}. \quad (4.19)$$

3. Then, we calculate the $EVPI$, which is given by the difference between the value of the solution of the complete two-stage model (Section 4.4) and the value the wait-and-see ws :

$$EVPI = z^{2SM} - ws. \quad (4.20)$$

4.6.3.2 The value of stochastic solution

The value of the stochastic solution (VSS) is how much better is the solution for the stochastic model than the solution for the deterministic model. For [Birge and Louveaux \(1997\)](#) (page 165), the amount VSS is the cost of ignoring uncertainty in choosing a decision. [Escudero, Garín and Pérez \(2007\)](#) presents a general procedure for obtaining VSS for multi staged stochastic models, as well as the two-stage model proposed here. We adapted the VSS proposed by [Alem et al. \(2010\)](#) for one-dimensional cutting stock problem to determine the VSS for the model proposed in this chapter. The procedure for calculating VSS , that is divided into 4 steps, is detailed below:

1. We calculate the average demand for each item and consider that we have only the average scenario. The we can consider the relaxed model, i.e., solve the expected value problem model (EVP) with $\bar{d}_i = \lfloor \sum_{s \in S} \rho_s d_{is} \rfloor, \forall i \in I$:

$$EVP = \text{minimize} \quad \sum_{j \in J} \sum_{t \in T} X_{jt} \quad (4.21)$$

$$\text{subject to:} \quad \sum_{j \in J} \sum_{t \in T} (a_{ij} X_{jt} - Z_{ijt}) = \bar{d}_i, \quad (4.22)$$

$$a_{ij} X_{jt} - Z_{ijt} \geq 0, \quad \forall j \in J, \forall t \in T, \quad (4.23)$$

$$X_{jt}, Z_{jit} \in \mathbb{Z}^+, \quad \forall j \in J, t \in T. \quad (4.24)$$

2. We calculate the values of variables y_{is}^+ and y_{is}^- for the solution of *EVP* (4.21)-(4.24).
3. After calculate the Step 1 and 2, we can calculate the expected objective value obtained using the deterministic problem *EVP*: $EEV = EVP + \sum_{i \in I} \sum_{s \in S} \rho_s (\delta_i^+ y_{is}^+ + \delta_i^- y_{is}^-)$.
4. The value of the stochastic solution is given by $VSS = EEV - z^{2SM}$.

4.6.3.3 Numerical results

Since we are interested in modeling real life applications, in this section, we consider the instances of ESICUP inspired in textile industry: Albano, Dagli, Mao, Marques, Shirts and Trousers. The main characteristics of the cutting stock problem with stochastic demands instances are presented in Table 27. These instances are available in <https://web.fe.up.pt/~esicup/>. For simplification, we used the adaptation of Song and Bennell (2014) to determine the length of the bins. The patterns used in this section are the same as the ones calculated in Section 4.6.1.

Table 27 – Instances stochastic for CSP.

Problem name	Number items	Rotacional constraints	Container width	Container length
ALBANO	8	0,180	4900	4900
DAGLI	10	0,180	60	60
MAO	9	0, 90,180, 270	2550	2550
MARQUES	8	0, 90,180, 270	104	104
SHIRTS	8	0,180	40	40
SWIM	10	0,180	5752	5752
TROUSERS	17	0,180	79	79

To adapt the problem to the stochastic problem, where each item have a limited number of scenarios, we define the following parameters, based on Alem *et al.* (2010):

- number of scenarios: $S = \{5, 10\}$;
- penalty to produce more or less of item i : $\delta_i^+ = \delta_i^- = 1$ for all $i \in I$;
- the demand of item i in scenario s is given by a normal distribution: $d_{is} = N(\mu, \sigma^2)$, for all $i \in I$ and $s \in S$;
- $\sigma = \{2, 10\}$.

In addition, for the two-stage model presented in Section 4.4 we need to define the set of patterns configurations T . In preliminary experiments we noted that in the optimal solution for GM (see Section 4.3.2) it is common to be used one or two configurations (the number of t), so we limit the number of configurations by the floor of half the number of items.

Table 28 – Results for stochastic demand with 1000 patterns and $d_{is} = N(1000, \sigma^2)$.

Instance	σ	Number of scenarios	m	z^{2SM}	$EVPI$	VSS	GAP	Time (s)
ALBANO	2	5	999	1012.0	225.5	21797611.6	0	2
	2	10	999	1012.3	471.5	34140430.6	0	2
	10	5	994	1062.6	275.4	19797892.8	0	1
	10	10	998	1064.5	523.5	33049099.6	0	3
DAGLI	2	5	350	364.8	14.2	7180050.4	0	2086
	2	10	350	364.8	14.5	39475311.2	11	TL
	10	5	351	426.0	75.4	5980594.4	8	TL
	10	10	349	423.7	73.6	42790444.2	5	TL
MAO	2	5	382	395.0	12.4	5365374.2	0	74
	2	10	382	395.8	13.4	42152808.6	0	77
	10	5	382	448.0	65.0	9487876.8	0	2195
	10	10	382	449.0	66.1	12753583	0	5
MARQUES	2	5	305	317.2	12.6	5027317.4	24	TL
	2	10	305	317.7	13.5	12060344.2	31	TL
	10	5	305	367.0	62.2	6472545.4	0	2584
	10	10	305	368.9	64.5	11878938.2	25	TL
SHIRTS	2	5	176	188.2	12.2	2568053.4	21	TL
	2	10	176	188.7	12.9	6364351.2	34	TL
	10	5	176	237.8	61.6	4179847.6	0	139
	10	10	175	238.9	63.2	6666800.2	0	67
SWIM	2	5	283	297.8	15.0	4241566.4	34	TL
	2	10	282	296.8	14.2	32574805.2	9	TL
	10	5	283	358.0	75.2	4843954.4	14	TL
	10	10	282	356.7	74.1	30810317.2	17	TL
TROUSERS	2	5	667	689.4	49.1	17486745.2	0	1332
	2	10	667	690.9	197.2	68609271.2	10	TL
	10	5	666	780.0	139.6	17043521.8	3	TL
	10	10	666	793.3	299.9	68524042.6	0	527

Table 28 presents the results for stochastic demand with a normal distribution with 1000 patterns and mean of the distribution $\mu = 1000$. In this table, we present the name of the instance, the demand variance (σ), the number of scenarios, the number of bins used (m), the value of the complete two stage model (z^{2SM}), the expected value of perfect information ($EVPI$) and the value of stochastic solution (VSS). As we can see, in Table 28, the positive values of VSS and $EVPI$ indicate that solving the problem considering stochastic demand is the best option.

In these experiments, the values of the instances with 5 scenarios are contained in the instances with 10 scenarios. For this reason, it is obvious that increasing σ or the number of scenarios increases the value of the stochastic solution, and also increases the value of VSS and $EVPI$. However, in Table 28 we can find some counter-examples, such as Dagli and Swim, for which we have lower solution value for 10 scenarios than for 5 scenarios (considering $\sigma = 10$). This happened because, in these cases, model optimality was not proven and thus cannot find a

better solution.

4.7 Conclusion

In this chapter we studied the explicit pattern generation for the irregular cutting stock problem, showing that randomly generated patterns may be better than column generation because of the difficulty to solve its subproblems. In addition, we have shown that the proposed method is very interesting for the case where items demands are uncertain, which happens much more frequently in practice.

For the irregular cutting stock problem with deterministic demand the proposed method improved the results for 18 out of 29 instances of the literature. In addition, the explicit patterns generation method improved the results obtained by H4NP (see Chapter 3).

We studied for the first time the irregular cutting stock problem with stochastic demand. For the stochastic problem, we extended the Gilmore and Gomory model used to propose a new two-stage programming model for the stochastic irregular cutting stock problem. To evaluate the proposed method with stochastic demands, we used two well-known measures: the value of stochastic solution (*VSS*) and the expected value of perfect information (*EVPI*). The *VSS* average of 107 units says how much optimality is being lost when ignoring uncertainty, while the average *EVPI* of 20,475,839 units shows how much we could win if we had the perfect foresight of uncertainty. These values of *VSS* and *EVPI* indicate that it is recommended to solve the problem with the multi-stage model.

IMPROVING NON-OVERLAP PARTITIONS FOR IRREGULAR STRIP PACKING MODELS

The irregular strip packing problem consists in cutting convex and non-convex pieces from a board with fixed height and infinite length. The objective is to minimize the length of the board used to cut all the pieces. In the last decade several mixed integer programming models have been proposed for the problem. However, these models may allow a large number of symmetric solutions, variables and constraints. The most recent mixed integer programming models describe the feasible placement positions for the pieces by covering (NFP-CM) or partitioning (HS2) nofit-polygons. In this chapter, we combine the best characteristics of the literature models and new symmetry breaking constraints into a single model. The results obtained by the model outperforms the literature in terms of lower bounds, run-time and number of nodes explored to prove optimality.

5.1 Introduction

The irregular strip packing problem (also known as irregular open dimensional problem) consists in cutting convex and non-convex pieces from a rectangular board with fixed width and infinite length. In the problem, pieces cannot overlap and each piece must be inside the board. The objective is to minimize the board length used to cut the pieces. The problem is both economical and environmental relevant, since reducing material consumption decreases production costs and minimizes wastes. Such problems appear in several industries, e.g., shoes manufacturing, metalworking, clothing and furniture manufacturing.

The irregular strip packing problem is NP-hard and the methods proposed to solve it are mainly heuristics. The best solutions for large benchmark instances were found in [Elkeran \(2013\)](#). He proposed a guided cuckoo search algorithm to solve the problem. In his method, the best fit between two pieces is found using a pairwise clustering and then the guided cuckoo

search algorithm is applied to find complete solutions.

Some mixed integer programming (MIP) models were proposed to represent the problem. [Fischetti and Luzzi \(2009\)](#) were the first to propose a MIP model to the problem considering that the pieces can be continuously placed inside of the board. [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) defined some structures used in [Fischetti and Luzzi \(2009\)](#) and developed a branch-and-bound algorithm to solve the model. In addition, the authors propose a mixed integer programming model derived from a compaction model proposed by [Gomes and Oliveira \(2006\)](#). [Santoro and Lemos \(2015\)](#) proposed a MIP model to convex polygons with some limitations: the polygons can have up to eight edges and their opposite edges must be parallel. In [Cherri et al. \(2016\)](#) we proposed two models that allow rotation of pieces. cap:apmod:cap:apmod: The models differ in the geometric structures used to avoid overlap among pieces. [Toledo et al. \(2013\)](#) proposed the dotted-board model where the pieces were placed in a finite set of dots inside the board. The optimality of the model is subject to the discretization used to generate the dots. In order to combine the accuracy of continuous models with the advantages of the dotted-board model, [Leao et al. \(2016\)](#) proposed a semi-continuous model where the pieces can be placed in continuous positions in x-axis and in discrete positions in y-axis.

While [Fischetti and Luzzi \(2009\)](#) model (HS1) and [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) model (HS2) emphasize the elimination of symmetries of the model, in [Cherri et al. \(2016\)](#) we proposed a lightweight model (NFP-CM) with less constraints which invokes less geometric information for the pieces and its interactions. Here, the best characteristics of MIP models presented from the literature are combined originating a model that emphasize the elimination of symmetries as in the models presented in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and still invoking simple geometries as in the model presented in [Cherri et al. \(2016\)](#). Furthermore, new symmetric breaking constraints are proposed for the problem. Computational experiments were performed with a set of convex and non-convex instances. The performance of the proposed model is improved in various aspects such as lower bounds, number of nodes explored and run-time.

This paper is organized as follows: Section 5.2 presents an analysis geometry used HS1, HS2 and NFP-CM models. In Section 5.3, a new model that mixes the ideas of the models in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and [Cherri et al. \(2016\)](#) is proposed. New symmetry breaking constraints are introduced in Section 5.4. Section 5.5 presents the computational results obtained by solving the proposed model. Finally, the conclusions are presented in Section 5.6.

5.2 Revisiting mixed integer programming models for strip packing problems

Given a set of N pieces and a board with fixed width W , the irregular strip packing problem consists in minimizing the length z of the board used to cut all pieces. Each piece is

defined by a set of vertices in clockwise orientation and one of the vertices is chosen to be its reference point. In a feasible solution for this problem, all the pieces must be placed inside the board and the pieces can not overlap. The containment of the pieces inside the board is ensured by the same constraints in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and [Cherri *et al.* \(2016\)](#). [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) proposed lifted bound constraints to the problem which makes HS2 model slightly better than HS1 model as presented in their computational experiments.

To depict these constraints, let $w_i^{\min}$ ($w_i^{\max}$) be the vertical distance between the reference point of piece i and the vertex of the piece that is nearest (farthest) to the origin. Likewise, let $\ell_i^{\min}$ ($\ell_i^{\max}$) be the distance of the reference point of piece i and the leftmost (rightmost) vertex of the piece. Consider (x_i, y_i) the variables that define the translation of the reference point of piece i to place it in the board. Figure 46 illustrates the distance parameters of a piece.

Figure 46 – Representation of a piece and its distance parameters in NFP-CM.

Therefore, the pieces are inside the board if the Constraints(5.1) and (5.2) are satisfied.

$$\ell_i^{\min} \leq x_i \leq z - \ell_i^{\max}, \quad i = 1, \dots, N, \quad (5.1)$$

$$w_i^{\min} \leq y_i \leq W - w_i^{\max}, \quad i = 1, \dots, N. \quad (5.2)$$

However, the main difference of HS1/HS2 and NFP-CM models is how they handle the non-overlapping constraints. To explain these constraints, the concept of no-fit polygon must be understood. The no-fit polygon between pieces i and piece j (NFP_{ij}) defines the region whether if these pieces are place, they overlap.

While HS1 and HS2 use the outside area of the no-fit polygon partitioned in convex polygons to avoid the overlap among pieces, NFP-CM model use analysis on each edge of the no-fit polygon. This characteristic makes HS1 and HS2 to consider less symmetric solutions in their solution space than NFP-CM model. Figure 47a shows the partitioning of the exterior region of a no-fit polygon as performed in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#). Figure 47b illustrates how the nofit polygon is used in NFP-CM.

The partitions presented in Figure 47a do not overlap while in Figure 47b there are several intersection areas in the exterior part of the no-fit polygon which mean more symmetric

Figure 47 – Outer no-fit polygon partitions. (a) presents the partition used in [Cherri et al. \(2016\)](#) and (b) presents the partition used in [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#)

solutions in the model. Specifically, in Figure 47b, piece j reference point can be placed in the dotted (hatched) region by being on the right side of two (three) distinct no-fit polygon edges.

In the next section, a new partitioning of outer part of the nofit polygon is presented. This partitioning originate a new set of constraints in NFP-CM model that eliminates some symmetric solutions.

5.3 New partition model

As stated in Section 5.2, the main difficulty to solve the irregular strip packing problem is the non-overlap constraints.

As in [Cherri et al. \(2016\)](#), here, a covering of the no-fit polygons by convex polygons is used, that is, each no-fit polygon is composed by a set of convex parts (polygons) that if united become the no-fit polygon. Specifically, $NFP_{ij} = \bigcup_{p \in P_{ij}} NFP_{ijp}$, where P_{ij} is the set of parts that compose the NFP_{ij} . The NFP_{ijp} is composed by K_{ijp} edges.

Also in [Cherri et al. \(2016\)](#), the following binary variables to prevent overlap: ξ_{ij}^{pk} which is one if the piece j reference point is in the right side of edge k of part p of polygon i , and zero otherwise. If piece i is on the right side of at least one edge $k = 1, \dots, K_{ijp}$ of each NFP_{ijp} , there is no overlap between pieces i and j .

NFP-CM is given by:

$$\min \quad z \tag{5.3}$$

s.t.

$$\ell_i^{\min} \leq x_i \leq z - \ell_i^{\max}, \quad i = 1, \dots, N, \quad (5.4)$$

$$w_i^{\min} \leq y_i \leq W - w_i^{\max}, \quad i = 1, \dots, N, \quad (5.5)$$

$$a_{ij}^{pk}(x_i - x_j) + b_{ij}^{pk}(y_i - y_j) \leq c_{ij}^{pk} + M_{ij}^{kp}(1 - \xi_{ij}^k), \quad 1 \leq i < j \leq N, \quad (5.6)$$

$$p \in P_{ij}, k = 1, \dots, K_{ijp},$$

$$\sum_{k=1}^{K_{ijp}} \xi_{ij}^{pk} = 1, \quad 1 \leq i < j \leq N, p \in P_{ij}, \quad (5.7)$$

$$\xi_{ij}^{pk} \in \{0, 1\}, \quad 1 \leq i < j \leq N, \quad (5.8)$$

$$p \in P_{ij}, k = 1, \dots, K_{ijp}.$$

Objective function (5.3) minimizes the length used to cut all pieces. Constraints (5.4) and (5.5) impose the pieces are placed inside the board. Constraints (5.6) prevent overlap among pieces, where $a_{ij}^{pk}(x_i - x_j) + b_{ij}^{pk}(y_i - y_j) = c_{ij}^{pk}$ is the equation of the line including the k -th edge of part p of the NFP_{ij} . Constraints (5.7) ensure each piece is on the right side of an edge of the no-fit polygon while constraints (5.8) define the domain of binary variables.

As performed by Alvarez-Valdes, Martinez and Tamarit (2013), the branching priority proposed by Fischetti and Luzzi (2009) was used to solve this model is higher for the pieces with larger area.

5.3.1 Partitioning the solution space

The main drawback of NFP-CM is non-overlap constraints (5.6). The definition of variables ξ_{ij}^{pk} generates overlap among various regions, as shown in Figure 47(b). Such redundancies on ξ_{ij}^{pk} values can be avoided by adding constraints (5.9) to the model.

Consider now a constraint that ensures the reference point of piece j is on the left side of the line defined by edge k' of part p of NFP_{ijp} , where edge k' precedes edge k in the no-fit polygon and that this constraint is activated by the same ξ_{ij}^{pk} variable. Therefore, whether the reference point of piece j is placed in the outside part of the no-fit polygon, only one ξ_{ij}^{pk} can be activated. Figure 48 illustrates the partitioning obtained from a no-fit polygon using this strategy.

These behavior is represented in Constraints (5.9).

$$\bar{a}_{ij}^{pk'}(x_i - x_j) + \bar{b}_{ij}^{pk'}(y_i - y_j) \leq \bar{c}_{ij}^{pk'} + \bar{M}_{ij}^{pk'}(1 - \xi_{ij}^{pk}), \quad 1 \leq i < j \leq N, \quad (5.9)$$

$$p \in P_{ij}, k = 1, \dots, K_{ij}.$$

where $\bar{a}_{ij}^{pk'}(x_i - x_j) + \bar{b}_{ij}^{pk'}(y_i - y_j) = \bar{c}_{ij}^{pk'}$ is the equation of the line including the k' -th edge of part p of the NFP_{ij} .

Figure 48 – Non-overlap regions using the proposed partitioning.

5.3.2 Variable eliminations

In this section, the variable reduction applied in NFP-CM by [Cherri *et al.* \(2016\)](#) is adapted to be valid with the new proposed constraints. Each time that two co-linear edges of different parts of the same no-fit polygons have the same orientation, they can be represented by the same variable and constraint in the model. Figure 49 illustrates a case of co-linear lines.

Figure 49 – Colinear lines.

For non-convex polygons, these variable reduction can eliminate several variables and constraints from the model. Once the set of colinear lines is defined, only one of them must be kept in the model, however, each of these lines has an associated support line and one of these support lines must be chosen to compose the model. If the wrong support line is chosen, some feasible solutions can be considered unfeasible and the optimal solution could be lost.

In fact, to define the set of colinear lines two conditions must be satisfied to ensure that the variable elimination is valid.

- The lines must be co-linear and have the same direction;
- Their support lines must have the same rotation angle.

Figure 50 illustrates that the colinear lines shown in Figure 49 fits the new definition. Ensuring these conditions, the support lines will never cross, thus, one support line contain the domain of all other support lines. The support line the cover all the other support lines is the one which pass by the leftmost starting point of the co-linear edges. Therefore, from a set of colinear lines, one variable, one non-overlap constraint and one supporting line constraint is created.

Figure 50 – Co-linear lines. The hatched area is the placement region if the co-linear lines are activated

5.4 Symmetry breaking constraints

NFP-CM (5.3)-(5.8) may have a huge number symmetric solutions. Therefore, identical or equivalent solutions can be found several times by the enumeration scheme. In this section symmetric breaking constraints are proposed. The constraints can improve the linear relaxation of the model and reduce the number of identical solutions.

5.4.1 Connecting pieces

In a feasible solution for the irregular strip packing problem, some pieces may not touch each other. However, one can translate the pieces to make they touch (connect them), obtaining another solution with the same quality. Consider the following variables:

$$\phi_{ij}^p = \begin{cases} 1, & \text{if piece } j \text{ touches the boundary of part } p \text{ of piece } i; \\ 0, & \text{otherwise.} \end{cases} \quad (5.10)$$

Constraints (5.11) combined with (5.6) ensure if ϕ_{ij} is equal to 1, the pieces i and j are connected.

$$a_{ij}^{pk}(x_i - x_j) + b_{ij}^{pk}(y_i - y_j) \geq c_{ij}^{pk} - \underline{M}_{ij}^{pk}(1 - \phi_{ij}^p), \quad 1 \leq i < j \leq N, \\ p \in P_{ij}, k = 1, \dots, K_{ijp}. \quad (5.11)$$

To ensure each piece is connected with another piece, the following constraints are imposed:

$$\sum_{j=1}^{i-1} \sum_{p \in P_{ij}} \phi_{ji}^p + \sum_{j=i+1}^N \sum_{p \in P_{ij}} \phi_{ij}^p \geq 1, \quad i = 1, \dots, N. \quad (5.12)$$

Note that constraints (5.12) do not ensure all pieces are connected as a single component. For example, it may have two groups of connected pieces with no interconnection. To enforce

all pieces are connected, one can impose all cut sets of pieces are connected, as defined by constraints (5.13).

$$\sum_{(i,j,p) \in \delta(S) | i < j} \phi_{ij}^p \geq 1, \quad \forall S \subset N, S \neq \emptyset \quad (5.13)$$

where $\delta(S)$ is the cutset of S .

Although constraints (5.13) impose connectivity, they introduce an exponential number of constraints in the model and therefore may hamper its resolution.

5.4.2 Connecting pieces with the board

In the optimal solution, at least one piece can touch the bottom most position of the board. In fact, if there is a solution that does not satisfy this statement, one can translate a piece down until it touches the board, obtaining a solution with the same quality.

Let ω_i be a binary variable, defined as follows:

$$\omega_i = \begin{cases} 1, & \text{if piece } i \text{ is in the bottommost position of the board;} \\ 0, & \text{otherwise.} \end{cases}$$

Constraints (5.14) and (5.15) ensure at least one piece is placed in the bottommost position in the board.

$$y_i \leq w_i^{\min} + (W - w_i^{\max} - w_i^{\min})(1 - \omega_i), \quad i = 1, \dots, N, \quad (5.14)$$

$$\sum_{i=1}^N \omega_i = 1. \quad (5.15)$$

5.4.3 Pieces with same shape

Symmetric solutions can be found if there are multiple pieces with same type (shape). For example, if there are two pieces with same type, all the solutions can be duplicated in an enumeration scheme. To avoid such problem, the pieces of same type can be ordered. Consider T_i , the type of piece i , to eliminate the symmetry on the placement of pieces of same type, the following constraints are imposed.

$$y_i \leq y_j, \quad i = 1, \dots, N, j = \min\{k = i + 1, \dots, N | T_k = T_i\}. \quad (5.16)$$

Note that constraints (5.16), except in the case where the pieces are placed in the same y -position, eliminate symmetry of identical pieces.

Some values of ω_i can be inferred for pieces with same type using constraints (5.14)-(5.16). Specifically, for a piece with several copies, only the smallest index constraint (5.14) can be active. Constraints (5.17) ensure only the smallest index copy of each piece type can activate constraints (5.14).

$$\omega_j = 0, \quad i = 1, \dots, N, j = \min\{k = i + 1, \dots, N | T_k = T_i\}. \quad (5.17)$$

5.5 Results

In this section we present computational experiments to evaluate the performance of the proposed symmetry breaking constraints. The experiments were run on a computer with Intel® Core™ i7-2600 3.40 GHz CPU and 16 GB RAM under Ubuntu 16.04 LTS 64 bits operating system. The instances were solved by IBM ILOG CPLEX 12.6 solver using default parameters and a time limit of 3,600 seconds.

Table 29 presents the main characteristics of the evaluated instances: the name, the author(s) who proposed it, the total number of pieces, the number of piece types, the number of non-convex pieces, the average number of vertices of the pieces, and the upper bound for the board length (UB). The total number of pieces ranges between 3 and 17, with an average of 9.9 pieces, and an average number of 50.42 vertices per instance. Additionally, we present the upper bound (UB) used to estimate the big-M coefficients in constraints (5.6) and (5.9), for more details see Cherri *et al.* (2016). The upper bound for each instance was calculated using the bottom-left heuristic proposed in Gomes and Oliveira (2006) with one thousand random input sequences.

Table 30 reports the computational experiments of NFP-CM proposed in Cherri *et al.* (2016) and the other partition model, i.e., NFP-CM with symmetry breaking constraints (5.9) and (5.14)-(5.17) with the name of improved NFP-CM (or iNFP-CM). iNFP-CM obtained better or equal lower bounds (LB) for all instances compared to NFP-CM. Moreover, iNFP-CM proved optimality for 37 instances, five more instances than NFP-CM. In average, iNFP-CM was 5.1 times more faster than NFP-CM to prove optimality. However, iNFP-CM was slower for instance santoro4 and do not find a feasible solution for rco2, poly1a and dighe1. Considering nodes explored by CPLEX, five instances were solved at the root node by iNFP-CM, two instances the more than the NFP-CM. For the remaining instances, iNFP-CM explored less nodes for 93.02% of the instances, an average reduction for instance of three million nodes. The biggest reduction of nodes explored to prove optimality was obtained for instance santoro1, where NFP-CM explored 185.36 times the amount of nodes of iNFP-CM. Furthermore, despite the proposed model is more general, it solved all santoro instances faster than the original paper Santoro and Lemos (2015).

We also highlight iNFP-CM is the first MIP model to prove optimality for instances

Table 29 – Features of benchmark instances.

Instance	Author(s)	Pieces	Piece types	Non-convex pieces	Avg. vertices	UB
three	Alvarez-Valdes, Martinez and Tamarit (2013)	3	3	0	3.67	6.67
shapes_4	Alvarez-Valdes, Martinez and Tamarit (2013)	4	4	3	8.75	24.00
jones1	Jones (2014)	4	4	0	3.25	5
fu5	Alvarez-Valdes, Martinez and Tamarit (2013)	5	4	0	3.20	37.00
glass1	Alvarez-Valdes, Martinez and Tamarit (2013)	5	5	3	5.00	45.00
fu6	Alvarez-Valdes, Martinez and Tamarit (2013)	6	5	0	3.33	17.89
threep2	Alvarez-Valdes, Martinez and Tamarit (2013)	6	3	0	3.67	9.67
threep2w9	Alvarez-Valdes, Martinez and Tamarit (2013)	6	3	0	3.67	8.67
jones2	Jones (2014)	6	6	0	3.83	10
fu7	Alvarez-Valdes, Martinez and Tamarit (2013)	7	6	0	3.43	24.00
glass2	Alvarez-Valdes, Martinez and Tamarit (2013)	7	7	5	5.43	45.00
rco1	Ribeiro, Carravilla and Oliveira (1999)	7	7	0	5.14	8.00
fu8	Alvarez-Valdes, Martinez and Tamarit (2013)	8	7	0	3.50	24.00
shapes_8	Alvarez-Valdes, Martinez and Tamarit (2013)	8	4	6	8.75	26.98
fu9	Alvarez-Valdes, Martinez and Tamarit (2013)	9	8	0	3.44	24.00
threep3	Alvarez-Valdes, Martinez and Tamarit (2013)	9	3	0	3.67	14.00
threep3w9	Alvarez-Valdes, Martinez and Tamarit (2013)	9	3	0	3.67	12.00
glass3	Alvarez-Valdes, Martinez and Tamarit (2013)	9	9	5	5.33	100.00
fu10	Alvarez-Valdes, Martinez and Tamarit (2013)	10	9	0	3.50	28.00
dighe2	Alvarez-Valdes, Martinez and Tamarit (2013)	10	10	1	4.70	133.00
J1-10-20-0	Alvarez-Valdes, Martinez and Tamarit (2013)	10	10	4	6.20	19.00
J1-10-20-1	Alvarez-Valdes, Martinez and Tamarit (2013)	10	10	3	4.60	18.00
J1-10-20-2	Alvarez-Valdes, Martinez and Tamarit (2013)	10	10	4	5.80	20.00
J1-10-20-3	Alvarez-Valdes, Martinez and Tamarit (2013)	10	10	3	5.80	22.00
J1-10-20-4	Alvarez-Valdes, Martinez and Tamarit (2013)	10	10	5	6.30	14.00
santoro1	Santoro and Lemos (2015)	11	11	0	8.00	28.00
santoro2	Santoro and Lemos (2015)	11	10	0	4.55	6.20
J1-12-20-0	Alvarez-Valdes, Martinez and Tamarit (2013)	12	12	5	6.00	12.00
J1-12-20-1	Alvarez-Valdes, Martinez and Tamarit (2013)	12	12	5	5.83	11.00
J1-12-20-2	Alvarez-Valdes, Martinez and Tamarit (2013)	12	12	4	5.83	14.00
J1-12-20-3	Alvarez-Valdes, Martinez and Tamarit (2013)	12	12	4	5.83	9.00
J1-12-20-4	Alvarez-Valdes, Martinez and Tamarit (2013)	12	12	4	6.08	14.00
fu	Alvarez-Valdes, Martinez and Tamarit (2013)	12	11	0	3.42	30.00
santoro3	Santoro and Lemos (2015)	12	8	0	4.33	6.50
J1-14-20-0	Alvarez-Valdes, Martinez and Tamarit (2013)	14	14	6	5.78	14.00
J1-14-20-1	Alvarez-Valdes, Martinez and Tamarit (2013)	14	14	5	5.50	13.00
J1-14-20-2	Alvarez-Valdes, Martinez and Tamarit (2013)	14	14	5	6.00	16.24
J1-14-20-3	Alvarez-Valdes, Martinez and Tamarit (2013)	14	14	6	6.21	12.00
J1-14-20-4	Alvarez-Valdes, Martinez and Tamarit (2013)	14	14	6	6.04	16.00
rco2	Ribeiro, Carravilla and Oliveira (1999)	14	7	0	5.14	15.00
poly1a	Alvarez-Valdes, Martinez and Tamarit (2013)	15	15	5	4.60	20.00
dighe1	Alvarez-Valdes, Martinez and Tamarit (2013)	16	16	1	3.87	139.80
santoro4	Santoro and Lemos (2015)	17	11	0	4.00	19.00

threep3, threep3w9, J1-10-20-3 and fu in less than 1,200 seconds (see Figure 51). To the instance fu, for example, the solution gap (given by $\frac{UB-LB}{UB}$) for other MIP models in 3,600 seconds using CPLEX is: 29% for the HS1 and HS2, 17% for GO (see Alvarez-Valdes, Martinez and Tamarit (2013)), and 16% for DTM, 14% for NFP-CMnc and NFP-CM (see the Cherri *et al.* (2016)).

In terms of the comparison with the literature, in Table 31, we present the results of the

Table 30 – Comparison between MIP models.

Instance	NFP-CM				iNFP-CM			
	LB	UB	Time (s)	Nodes	LB	UB	Time (s)	Nodes
three	6.00	6.00	0	0	6.00	6.00	0	0
shapes_4	24.00	24.00	0	577	24.00	24.00	0	0
jones1	5.00	5.00	0	0	5.00	5.00	0	0
fu5	17.89	17.89	0	336	17.89	17.89	0	235
glass1	45.00	45.00	0	1,582	45.00	45.00	0	0
fu6	23.00	23.00	0	428	23.00	23.00	0	452
threep2	9.33	9.33	2	21,360	9.33	9.33	1	12,963
threep2w9	8.00	8.00	6	107,113	8.00	8.00	1	15,217
jones2	10.00	10.00	0	0	10.00	10.00	0	0
fu7	24.00	24.00	0	1,324	24.00	24.00	0	1,481
glass2	45.00	45.00	64	348,943	45.00	45.00	1	448
rco1	8.00	8.00	1	14,127	8.00	8.00	1	12,008
fu8	24.00	24.00	0	851	24.00	24.00	0	3,127
shapes_8	26.00	26.00	479	631,791	26.00	26.00	238	226,520
fu9	25.00	25.00	7	83,709	25.00	25.00	4	32,911
threep3	10.06	13.53	TL	24,408,906	13.53	13.53	928	9,711,358
threep3w9	8.75	11.00	TL	33,442,609	11.00	11.00	1,113	9,036,051
glass3	100.00	100.00	377	1,080,163	100.00	100.00	8	12,999
fu10	28.69	28.69	301	2,653,576	28.69	28.69	33	253,314
dighe2	100.00	100.00	38	158,237	100.00	100.00	15	38,580
J1-10-20-0	18.00	18.00	8	28,692	18.00	18.00	5	12,235
J1-10-20-1	17.00	17.00	4	15,412	17.00	17.00	1	4,288
J1-10-20-2	20.00	20.00	5	19,180	20.00	20.00	4	9,355
J1-10-20-3	20.00	20.75	TL	10,457,796	20.75	20.75	281	1,094,528
J1-10-20-4	12.00	12.50	150	434,544	12.50	12.50	60	146,350
santoro1	26.85	28.00	TL	49,058,509	28.00	28.00	41	264,666
santoro2	6.00	6.00	474	6,958,452	6.00	6.00	19	120,888
J1-12-20-0	12.00	12.00	40	88,034	12.00	12.00	11	27,713
J1-12-20-1	10.00	10.00	39	85,278	10.00	10.00	18	44,033
J1-12-20-2	12.00	12.00	51	136,646	12.00	12.00	18	26,242
J1-12-20-3	8.00	8.00	160	272,438	8.00	8.00	11	18,914
J1-12-20-4	13.00	13.00	1,619	3,000,250	13.00	13.00	173	307,237
fu	28.00	33.14	TL	18,761,708	33.14	33.14	511	2,539,903
santoro3	6.00	6.00	5	30,938	6.00	6.00	5	17,955
J1-14-20-0	12.00	12.00	844	2,017,123	12.00	12.00	264	479,236
J1-14-20-1	9.92	12.00	TL	3,967,673	11.00	11.33	TL	2,567,398
J1-14-20-2	12.00	14.00	TL	7,165,315	12.00	14.00	TL	5,119,634
J1-14-20-3	10.00	10.00	295	384,586	10.00	10.00	39	54,031
J1-14-20-4	11.90	14.00	TL	7,318,967	12.00	14.00	TL	3,480,851
rco2	6.67	-	TL	17,116,136	9.00	-	TL	10,534,103
poly1a0	13.00	16.37	TL	7,355,762	13.00	-	TL	4,350,937
dighe1	100.00	122.75	TL	7,387,732	100.00	-	TL	7,391,089
santoro4	15.00	15.00	24	178,258	15.00	15.00	52	131,688
Optimal			32				37	

iNFP-CM against the best known results. An overview about this results is show in Figure 52, where we can observed than the iNFP obtained results are better or equal to 95.12% in terms of lower bound and upper bound. The computational time is less or equal to all instances.

Figure 51 – Optimal solutions proved by iNFP-CM and not proved for NFP-CM in less than 3,600 seconds.

Table 31 – Comparison with the literature.

Instance	iNFP-CM			best solution of literature			Author
	LB	UB	Time (s)	LB	UB	Time (s)	
three	6.00	6.00	0	6.00	6.00	1	Alvarez-Valdes, Martinez and Tamarit (2013)
shapes_4	24.00	24.00	0	24.00	24.00	0	Alvarez-Valdes, Martinez and Tamarit (2013)
jones1	5.00	5.00	0	4.99	5.00	22671	Jones (2014)
fu5	17.89	17.89	0	17.89	17.89	0	Alvarez-Valdes, Martinez and Tamarit (2013)
glass1	45.00	45.00	0	45.00	45.00	0	Alvarez-Valdes, Martinez and Tamarit (2013)
fu6	23.00	23.00	0	23.00	23.00	0	Alvarez-Valdes, Martinez and Tamarit (2013)
threep2	9.33	9.33	1	9.33	9.33	4	Alvarez-Valdes, Martinez and Tamarit (2013)
threep2w9	8.00	8.00	1	8.00	8.00	8	Alvarez-Valdes, Martinez and Tamarit (2013)
jones2	10.00	10.00	0	5.44	11.44	2596 ^a	Jones (2014)
fu7	24.00	24.00	0	24.00	24.00	1	Alvarez-Valdes, Martinez and Tamarit (2013)
glass2	45.00	45.00	1	45.00	45.00	3	Alvarez-Valdes, Martinez and Tamarit (2013)
fu8	24.00	24.00	0	24.00	24.00	1	Alvarez-Valdes, Martinez and Tamarit (2013)
shapes_8	26.00	26.00	238	26.00	26.00	272	Alvarez-Valdes, Martinez and Tamarit (2013)
fu9	25.00	25.00	4	25.00	25.00	70	Alvarez-Valdes, Martinez and Tamarit (2013)
threep3	13.53	13.53	928	13.53	13.53	3394	Alvarez-Valdes, Martinez and Tamarit (2013)
threep3w9	11.00	11.00	1,113	11.00	11.00	2144	Cherri <i>et al.</i> (2016)
glass3	100.00	100.00	8	100.00	100.00	324	Alvarez-Valdes, Martinez and Tamarit (2013)
fu10	28.69	28.69	33	28.69	28.69	3064	Alvarez-Valdes, Martinez and Tamarit (2013)
dighe2	100.00	100.00	15	100.00	100.00	177	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-10-20-0	18.00	18.00	5	18.00	18.00	45	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-10-20-1	17.00	17.00	1	17.00	17.00	34	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-10-20-2	20.00	20.00	4	20.00	20.00	304	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-10-20-3	20.75	20.75	281	20.67	20.75	TL	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-10-20-4	12.50	12.50	60	12.50	12.50	628	Alvarez-Valdes, Martinez and Tamarit (2013)
santoro1	28.00	28.00	41	28.00	28.00	0	Santoro and Lemos (2015)
santoro2	6.00	6.00	19	6.00	6.00	2400	Santoro and Lemos (2015)
J1-12-20-0	12.00	12.00	11	12.00	12.00	509	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-12-20-1	10.00	10.00	18	10.00	10.00	2430	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-12-20-2	12.00	12.00	18	12.00	12.00	2332	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-12-20-3	8.00	8.00	11	8.00	8.00	214	Alvarez-Valdes, Martinez and Tamarit (2013)
J1-12-20-4	13.00	13.00	173	13.00	13.00	1619	Cherri <i>et al.</i> (2016)
fu	33.14	33.14	511	28.50	33.14	TL	Cherri <i>et al.</i> (2016)
santoro3	6.00	6.00	5	6.00	6.00	389	Santoro and Lemos (2015)
J1-14-20-0	12.00	12.00	264	12.00	12.00	295	Cherri <i>et al.</i> (2016)
J1-14-20-1	11.00	11.33	TL	9.92	12	TL	Cherri <i>et al.</i> (2016)
J1-14-20-2	12.00	14.00	TL	12.00	14.00	TL	Cherri <i>et al.</i> (2016)
J1-14-20-3	10.00	10.00	39	10.00	10.00	295	Cherri <i>et al.</i> (2016)
J1-14-20-4	12.00	14.00	TL	11.90	14.00	TL	Cherri <i>et al.</i> (2016)
poly1a0	-	-	TL	13.00	16.35	TL	Cherri <i>et al.</i> (2016)
dighe1	-	-	TL	100.00	122.75	TL	Cherri <i>et al.</i> (2016)
santoro4	15.00	15.00	52	15.00	15.00	2401	Santoro and Lemos (2015)
Best result	39	39	36	35	39	5	

^aInsufficient memory.

5.6 Conclusions

The model proposed in this chapter merge the simplicity of geometric structures used in [Cherri *et al.* \(2016\)](#) and the ideas of eliminate the symmetric solutions used by [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#). Thus, new constraints that divides the external area of no-fit polygon into convex disjoint regions is proposed, leading to a new set of non overlap constraints. Using this new set of constraints the variable elimination presented in [Cherri *et al.* \(2016\)](#) were no longer valid, thus, additional conditions were introduced to make these eliminations valid. Furthermore, a new set of symmetry breaking constraints for the irregular strip packing problem were proposed. Although this new set of constraints was added to iNFP-CM, they can be applied to the [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and [Cherri *et al.* \(2016\)](#) models.

Computational experiments show the relevance of the proposed model. In general, iNFP-CM model has improved lower bounds and reduced run-time and the number of nodes explored to prove optimality, proving the optimality of 86% of instances. NFP-CM, which is the model in the literature that reports more optimums for this set of instances, prove the a optimality of 74% (five instances less).

In terms of the comparison with the literature the Figure 52 presents an overview about the iNFP-CM results. We can see than the iNFP-CM is better or equal to 95.12% in terms of lower bound and upper bound, while the computational time is less or equal to all instances.

Figure 52 – Overview about the results in comparasion with the literature.

MIXED-INTEGER LINEAR PROGRAMMING MODELS FOR THE IRREGULAR STRIP PACKING PROBLEM: ANALYZING THE USE OF CONTINUOUS MODELS IN SEMI-CONTINUOUS AND DISCRETE DOMAINS

The irregular strip packing problem consists in placing a set of polygons into a strip, minimizing the used length of this strip. This problem appears in many industries, such as the textile industry. In the literature, this problem is mostly solved with heuristic methods. However, mixed integer linear programming models for irregular strip packing problem have recently been proposed. These models can be classified in three classes: 1) discrete models, 2) semi-continuous models and 3) continuous models. In this chapter, an adaptation to the state of the art continuous models are proposed, moving it to the class of discrete and semi-continuous models. Computational experiments were performed and the results show that the solutions of the adapted continuous models is competitive with those from specialized models. The optimality of 22.58% of benchmark of the literature was proven for the first time. For 96.77% of the instances, better or equal results than the specialized models were computed.

6.1 Introduction

The strip packing problem is a combinatorial optimization problem that belongs to the class of the cutting and packing problems. This problem is defined in the typology of [Wäscher, Haußner and Schumann \(2007\)](#) as a minimization problem, where a set of small objects (or

items) should be placed inside of a strip of fixed height aiming to minimize the length used to place all the items.

When the items that must be placed in the strip are represented by convex and non-convex polygons, the problem is called irregular strip packing problem, also known as nesting problems. Many industries consider the irregular strip packing problem, such as the textile or furniture industry.

The solution methods proposed for the irregular strip packing problem are predominantly hybrid algorithms ([Gomes and Oliveira \(2006\)](#) and [Elkeran \(2013\)](#)), matheuristics ([Martinez-Sykora et al. \(2016\)](#) and [Cherri, Carravilla and Toledo \(2016\)](#)), and metaheuristic, such as the first genetic algorithm for the irregular strip packing problem from [Jakobs \(1996\)](#), the guided local search from [Umetani et al. \(2009\)](#), the fast neighborhood search from [Egeblad, Nielsen and Odgaard \(2007\)](#). Others authors proposed constructive algorithms based on the bottom-left heuristic from [Baker, Jr and Rivest \(1980\)](#) and more recently improved by [Dowsland, Dowsland and Bennell \(1998\)](#), [Dowsland, Vaid and Dowsland \(2002\)](#) and [Burke et al. \(2006\)](#).

Mixed integer linear programming models for the irregular strip packing problem have been recently proposed. These models can be classified in three classes: 1) discrete models, where the items were placed in a finite set of dots inside the strip; 2) semi-continuous models, where the items can be placed in continuous positions in the x -axis and in discrete positions in the y -axis; and 3) continuous models, where the items can be placed in continuous positions in the strip.

In the class of discrete models, [Toledo et al. \(2013\)](#) developed the dotted board model, the first discrete model considering that the items can be placed on a finite number of positions on the board. Inspired by [Toledo et al. \(2013\)](#), [Rodrigues and Toledo \(2016\)](#) proposed heuristics to generate stronger non-overlap constraints for the dotted board model.

Continuous models were first explored by [Fischetti and Luzzi \(2009\)](#), which also proposed a heuristic to solve large scale problems. [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) formally define the partitions needed to construct the non-overlap constraints in [Fischetti and Luzzi \(2009\)](#) model and define a particular partition which allows them to perform lifting in the bound constraints. The authors also proposed a mixed integer linear programming model derived from a compaction model proposed by [Gomes and Oliveira \(2006\)](#). [Cherri et al. \(2016\)](#) proposed two models which allow fixed rotations of the items. Both models differ on how the overlap among items is avoided. Specifically, one model uses only direct trigonometry to ensure the non-overlap between items while the other model uses the no-fit polygon to perform this task.

A semi-continuous model was recently proposed by [Leao et al. \(2016\)](#), intending to combine the best characteristics of continuous and discrete models.

Each class of mixed integer linear programming models described above have different variables and domains. Thus, the optimal solutions may differ when using models of different

classes to solve the same instance. Despite the different model characteristics, there is no study analyzing the impact of changing the variable domains of the best continuous models (HS2 from [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and NFP-CM from [Cherri *et al.* \(2016\)](#)) into discrete and semi-continuous. Such domain changes allow a fair comparison among the class of continuous model with the other two classes and deliver optimal and feasible solutions for problems where the items can not be placed in some positions of the strip.

In this chapter, the variables domain in the mixed integer linear programming models for the irregular strip packing problem are analyzed. Specifically, an adaptation of the continuous models is proposed in order to convert their domains into discrete and semi-continuous domains. These adaptations are very important in special cases of industry where the items can be cut only in restricted positions from a discrete domain or semi-continuous domain. This special cases appear, for example, in the cutting of striped fabrics in textile industry. Consider a textile industry problem where the objective is to cut pants from a color striped fabric as illustrated in [Figure 53](#). The right and left trouser's legs should be cut in a region where, when the trouser is finished, the stripes of different parts of the pant fit perfectly.

(a) a color striped fabric.

(b) pants from a color striped.

Figure 53 – Example of a striped trouser cut .

After this brief introductory section, in [Section 6.2](#) we present the continuous mixed integer programming models for irregular strip packing problem and the adaptation of these models for discrete and semi-continuous domains. In [Section 6.3](#) we present the numerical results obtained by solving these adapted models and compare them to the ones from the specialized

models. The conclusions and future studies are presented in Section 6.4.

6.2 Continuous MIP models for nesting problems

Recently, two mixed integer programming models for irregular strip packing problems were proposed, HS2 from [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and NFP-CM from [Cherri et al. \(2016\)](#). Both models use the no-fit polygon structure to avoid the overlap between items. However, these models have small differences that will be highlighted in this section.

Consider that item i has a reference point given by (x_i, y_i) and length and width given by l_i and w_i , respectively. The reference point is used in continuous models to translate the item to the locus that they should be placed on the strip. Figure 54 illustrates a item with its reference point.

Figure 54 – Item i and its reference point. Extracted from [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#).

Usually, the objective in a continuous model for irregular strip packing problem is to minimize the maximum length of the strip used to perform the cut of all items (Constraint 6.1). This objective must be supported by Constraints 6.2 that account for the length used by the rightmost item in the strip.

Furthermore, a set of constraints must be respected. Specifically, the items must respect the strip dimensions (Constraints 6.3, 6.4 and 6.5), the non-overlap between items (Constraints 6.6) and the reference point domain, which can be placed in a real point of the feasible region (Constraints 6.7). Constraints 6.6 are the only difference between HS2 and NFP-CM.

$$\min \quad L \quad (6.1)$$

$$\text{subject to: } x_i \leq L - l_i, \quad i = 1, \dots, N, \quad (6.2)$$

$$x_i \geq 0, \quad i = 1, \dots, N, \quad (6.3)$$

$$y_i \leq H - h_i, \quad i = 1, \dots, N, \quad (6.4)$$

$$y_i \geq 0, \quad i = 1, \dots, N, \quad (6.5)$$

$$\text{non-overlap}_{ij}, \quad \forall (i, j) \text{ with } i \neq j, \quad (6.6)$$

$$(x_i, y_i) \in \mathbb{R}^+, \quad i = 1, \dots, N, \quad (6.7)$$

The non-overlap constraints of the HS2 and NFP-CM are both based in the concept of the no-fit polygon, i.e. the region where the items do not overlap. Succinctly, to avoid the overlap between two items (i and j), the reference point of item i must be out of the no-fit polygon of items i and j . For this, HS2 and NFP-CM use the D-function (a mathematical function that verifies whenever a point is on the right, on the left or over a line).

The difference between HS2 and NFP-CM is on how the no-fit polygon is used to avoid the items overlap. While HS2 consider the no-fit polygon as a unique polygon, NFP-CM divide the no-fit polygon into convex partitions. As consequence, the holes and concavities in HS2 must be closed to construct the non-overlap constraints and NFP-CM handles the overlap using each convex part.

Figure 55 – Different no-overlap regions of NFP-CM and HS2.

The Figure 55 illustrates a specific case, where we show the difference of two no-fit polygons in approaches of HS2 and NFP-CM. The partition of the NFP-CM is shown in Figure 55(a). This model considers that the no-fit polygon of items i and j is composed by two parts (NFP_{ij}^1 and NFP_{ij}^2). Therefore, if the reference point of item j (x_j, y_j) is placed outside or over a line of both of these parts, there is no overlap involving these items. To ensure the non-overlap between items i and j , Cherri *et al.* (2016) uses the D-function to force the reference

point be placed on the right of at least one vector b_k^1 for $k = 1, \dots, 6$ from NFP_{ij}^1 and the right of at least one vector b_k^2 for $k = 1, \dots, 7$ from NFP_{ij}^2 . The use of the no-fit polygon for HS2 is illustrated in Figure 55(b). In this case, the model from [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) forces the reference point to be placed inside region p_l , for $l = 1, \dots, 9$.

Note that, using these formulations as they were proposed, it is not possible to obtain solutions for striped or plaid clothes. However, changing the domain of variables x_i and y_i (reference points) we can have a discrete or a semi-continuous version of these models that can properly model the clothing industry problems. This domain change is performed by modifying Constraints 6.7. Specifically, for the discrete models, Constraints 6.7 must be replaced by Constraints 6.8.

$$(x_i, y_i) \in \mathbb{Z}^+, \quad i = 1, \dots, N. \quad (6.8)$$

For the semi-continuous models, only the domain of variable y_i should be modified, that is, Constraints 6.7 must be replaced by Constraints 6.9.

$$x_i \in \mathbb{R}^+ \text{ and } y_i \in \mathbb{Z}^+, \quad i = 1, \dots, N. \quad (6.9)$$

6.3 Computational results

This section presents computational experiments to evaluate the performance of solving HS2 and NFP-CM in semi-continuous (HS2^{sc} and $\text{NFP-CM}^{\text{sc}}$) and discrete domains (HS2^d and NFP-CM^d). The experiments were run on a computer with Intel® Core™ i7-2600 @ 3.40 GHz CPU and 16 GB RAM under Ubuntu 16.04 LTS 64 bits operating system. The models were coded in C++ programming language in IBM ILOG CPLEX 12.7 solver with a time limit of 3600 seconds and using default parameters. The results of Semi-continuous model from [Leao et al. \(2016\)](#) were extracted from the original paper.

6.3.1 Instances

To perform the experiments, a set of 31 instances proposed by [Alvarez-Valdes, Martinez and Tamarit \(2013\)](#) and used by [Leao et al. \(2016\)](#) and [Cherri et al. \(2016\)](#) were considered. These instances are presented in Table 32. The columns of this table present the name of each instance, the number of different types of items, the total number of items and the strip height. The last table column presents a strip length than is used by dotted-board model from [Toledo et al. \(2013\)](#). This parameter is important for the discrete models, since the number of variables in the dotted-board model is directly proportional to this value. The instances used here are the large instances used on experiments from [Leao et al. \(2016\)](#), where the authors selected instances

to make a comparison of the between the Semi-continuous model, dotted-board model and HS2 considering domains different.

Table 32 – Instances used in the computational experiments.

Instances	Number of item types	Total number of items	Strip height	Strip length
Three	3	3	7	15
Threep2	3	6	7	20
Threep2w9	3	6	9	20
Threep3	3	9	7	25
Threep3w9	3	9	9	20
Shapes4	4	4	13	40
Shapes8	4	8	20	30
Fu5	4	5	38	20
Fu6	5	6	38	26
Fu7	6	7	38	27
Fu8	7	8	38	27
Fu9	8	9	38	29
Fu10	9	10	38	33
Fu	11	12	38	50
Dighe1	16	16	100	150
Dighe2	10	10	100	150
J1-10-20-0	10	10	10	25
J1-10-20-1	10	10	10	25
J1-10-20-2	10	10	10	25
J1-10-20-3	10	10	10	25
J1-10-20-4	10	10	10	25
J1-12-20-0	12	12	20	25
J1-12-20-1	12	12	20	25
J1-12-20-2	12	12	20	25
J1-12-20-3	12	12	20	25
J1-12-20-4	12	12	20	25
J1-14-20-0	14	14	20	25
J1-14-20-1	14	14	20	25
J1-14-20-2	14	14	20	25
J1-14-20-3	14	14	20	25
J1-14-20-4	14	14	20	25

6.3.2 Discrete domain

This section presents the results of the dotted-board model from [Toledo *et al.* \(2013\)](#) and the adaptation of HS2 and NFP-CM to consider discrete domains, these models are called as HS2^d and NFP-CM^d. It is important to highlight that this adaptation makes every solution found by HS2^d and NFP-CM^d feasible to the dotted-board model and every solution found by dotted-board model feasible to HS2^d and NFP-CM^d. The models have the same mesh of points, that is, the same solution space.

Table 33 presents the computational experiments comparing the dotted-board model, HS2^d and NFP-CM^d. In the table, the first and second columns present the instance name and the number of items in the instance, respectively. Columns three and four show the solution found by solving the instance by the dotted-board model and its computational time (in seconds), respectively. Columns five and six (seven and eight) present the same content from columns three and four but for HS2^d (NFP-CM^d).

Table 33 – Comparing the results with the dotted-board model from Toledo *et al.* (2013).

Instances	Items	Dotted-board		HS2 ^d		NFP-CM ^d	
		Solution	Time	Solution	Time	Solution	Time
Three	3	6.00	0	6.00	0	6.00	0
Threep2	6	10.00	0	10.00	1	10.00	2
Threep2w9	6	8.00	0	8.00	2	8.00	3
Threep3	9	14.00	0	14.00	1447	14.00	TL
Threep3w9	9	12.00	0	12.00	TL	12.00	TL
Shapes4	4	24.00	3	24.00	0	24.00	0
Shapes8	8	26.00	173	26.00	2	26.00	357
Fu5	5	18.00	9	18.00	0	18.00	1
Fu6	6	23.00	566	23.00	0	23.00	1
Fu7	7	24.00	1344	24.00	0	24.00	1
Fu8	8	24.00	1386	24.00	0	24.00	1
Fu9	9	28	TL	25.00	7	25.00	5
Fu10	10	—	—	29.00	722	29.00	1050
Fu	12	—	—	34.00	TL	34.00	TL
Dighe1	16	—	—	130.00	TL	114.00	TL
Dighe2	10	—	—	100.00	26	100.00	19
J1-10-20-0	10	18.00	19	18.00	5	18.00	6
J1-10-20-1	10	17.00	19	17.00	8	17.00	7
J1-10-20-2	10	20.00	5	20.00	41	20.00	6
J1-10-20-3	10	21.00	28	21.00	2395	21.00	TL
J1-10-20-4	10	13.00	8	13.00	1007	13.00	98
J1-12-20-0	12	12.00	37	12.00	339	12.00	33
J1-12-20-1	12	10.00	46	10.00	84	10.00	27
J1-12-20-2	12	12.00	92	12.00	3569	12.00	73
J1-12-20-3	12	8.00	51	9.00	TL	8.00	203
J1-12-20-4	12	13.00	68	13.00	TL	13.00	2042
J1-14-20-0	14	12.00	61	14.00	TL	12.00	2922
J1-14-20-1	14	12.00	100	12.00	TL	12.00	TL
J1-14-20-2	14	14.00	719	15.00	TL	15.00	TL
J1-14-20-3	14	10.00	41	10.00	3092	10.00	1187
J1-14-20-4	14	14.00	407	15.00	TL	15.00	TL
#Optimal		26		22		23	
#Best solution		26		26		29	

#Optimal: number of instances for which optimality was proven.

#Best solution: number of instances than have the best solution.

Analyzing the results, NFP-CM^d found better or equal solutions for 29 instances out of

the 31 instances, while $HS2^d$ and the dotted-board model found better or equal solutions for 26 instances. The dotted-board model found strictly better solutions for the instances J1-14-20-1, J1-14-20-2 and J1-14-20-4 and was able to prove optimality for 26 instances. On the other hand, the memory cost for solve the dotted-board model was much higher, spending more than 16 GB RAM for instances Fu10, Fu, Dighe1 and Dighe1. Both $HS2^d$ and $NFP-CM^d$ have a much lower memory cost, not exceeding the computational limit. $HS2$ proved optimality for 22 instances, while $NFP-CM$ proved optimality for 23 instances. Figure 56 illustrates these numbers.

Figure 56 – Overview about the results in the discrete domain.

The dotted-board model is the only model that can take advantage of identical items, that is, while $HS2^d$ and $NFP-CM^d$ consider identical items as several copies of different items, the dotted-board model unifies identical items by the variable definition, reducing both the number of variables and constraints in the model. For example, consider instance BLAZP2P4_28_28, with 58 items of 2 different types, presented in Toledo *et al.* (2013). In their paper, the authors solve this instance, which is the largest instance with known optimal solution in the literature. CPLEX can barely build the model for an instance of this size for adapted models $HS22^d$ and $NFP-CM2^d$.

6.3.3 Semi-continuous domain

This section presents the results of the semi-continuous model proposed by Leao *et al.* (2016) and the adapted $HS2$ and $NFP-CM$ models (or $HS2^{sc}$ and $NFP-CM^{sc}$). Table 34 shows the computational experiments involving these three models. The columns of this table have the same type of content as Table 33, this same structure or content was used by Leao *et al.* (2016).

$HS2^{sc}$ and $NFP-CM^{sc}$ models proved optimality for 22 instances while the semi-continuous model proved optimality for 13 instances. From the set of instances that $HS2^{sc}$ and $NFP-CM^{sc}$ proved optimality, they only differ on instances J1-10-20-3 (whose optimality was proved by $HS2$) and J1-14-20-3 (whose optimality was proved by $NFP-CM^{sc}$). In terms of solution quality, $NFP-CM^{sc}$ was better or equal for all instances, while $HS2^{sc}$ and the semi-continuous model was better or equal for 29 and 24 out of 31 instances, respectively. Figure 57 illustrates this data.

Figure 58 shows the solutions found by $NFP-CM^{sc}$ that were strictly better than the ones found by semi-continuous model. The lines in this figure represent the domain in which the

Table 34 – Comparing the semi-continuous model with HS2 and NFP-CM adapted to semi-continuous domains.

Instances	Items	Semi-continuous*		HS2 ^{sc}		NFP-CM ^{sc}	
		Solution	time	Solution	time	Solution	time
Three	3	6.00	0	6.00	0	6.00	0
Threep2	6	9.67	1	9.67	2	9.67	2
Threep2w9	6	8.00	3	8.00	5	8.00	3
Threep3	9	14.00	853	14.00	TL	14.00	TL
Threep3w9	9	11.33	TL	11.33	TL	11.33	TL
Shapes4	4	24.00	1	24.00	0	24.00	0
Shapes8	8	26.00	49	25.00	13	26.00	577
Fu5	5	17.89	76	17.89	0	17.89	0
Fu6	6	23.00	443	23.00	0	23.00	1
Fu7	7	24.00	TL	24.00	0	24.00	1
Fu8	8	24.00	TL	24.00	0	24.00	1
Fu9	9	25.00	TL	25.00	8	25.00	5
Fu10	10	30.00	TL	29.00	334	29.00	293
Fu	12	34.44	TL	33.33	TL	33.33	TL
Dighe1	16	140.29	TL	127.87	TL	126.29	TL
Dighe2	10	130.75	TL	100.00	12	100.00	12
J1-10-20-0	10	18.00	18	18.00	12	18.00	4
J1-10-20-1	10	17.00	27	17.00	7	17.00	11
J1-10-20-2	10	20.00	48	20.00	30	20.00	17
J1-10-20-3	10	20.75	111	20.75	1677	20.75	TL
J1-10-20-4	10	13.00	71	13.00	592	13.00	639
J1-12-20-0	12	12.00	TL	12.00	261	12.00	47
J1-12-20-1	12	10.00	TL	10.00	370	10.00	38
J1-12-20-2	12	12.00	TL	12.00	100	12.00	29
J1-12-20-3	12	8.00	TL	8.00	501	8.00	520
J1-12-20-4	12	14.00	TL	13.00	3573	13.00	2339
J1-14-20-0	14	14.00	TL	13.00	TL	13.00	TL
J1-14-20-1	14	12.00	TL	12.00	TL	12.00	TL
J1-14-20-2	14	14.00	TL	14.00	TL	14.00	TL
J1-14-20-3	14	11.00	TL	10.00	TL	10.00	931
J1-14-20-4	14	14.00	TL	15.00	TL	14.00	TL
#Optimal		13		22		22	
#Best solution		24		29		31	

*: Results taken from [Leao et al. \(2016\)](#)

items can be placed. Note that for Dighe1 and Dighe2 instances, a hundred lines were needed due the size of bin. Moreover, 38 lines were needed for Fu10 and Fu instances and 20 lines were needed for J1-12-20-4, J1-14-20-0 and J1-14-20-3 instances.

Figure 57 – Overview about the results in the semi-continuous domain.

Figure 58 – Solutions strictly better than the semi-continuous model leao .

6.4 Conclusions

This paper presented adaptations in two mixed integer linear programming models for irregular strip packing, changing the domains of its variables for discrete and semi-continuous. This domain change is relevant in situations where the material to be cut is striped or plaid.

Computational experiments were performed comparing the solutions of the proposed adapted models and models that only consider striped and plaid strips. The results show that using the adapted continuous models can be advantageous in many cases, leading to new perspective about using of continuous models for nesting problems. For the benchmark of 62 tests: NFP-CM got better or equal solutions for 60 (96.77%) while HS2 got better or equal solutions for 55 (88.71%). Considering only the tests for semi-continuous domains, the proposed approach proved

optimality for 11 new instances and improved the best known solution for 7 instances from the literature. Looking the tests for discrete domains the proposed models proved the optimality for 3 new instances and improved the best know solution for 5 instances.

To validate the approach proposed in this chapter, computational experiments were performed on a benchmark of the literature with 62 instances, where HS2 and the NFP-CM were better or equal the specialized models for 88.71% and 96.77%. The proposed models proved the optimality for 14 new instances and improved the best known solutions of 12 instances.

Furthermore, the proposed approach encourages the study of irregular cutting and packing problems where material has different patterns, as diagonal stripes.

CONCLUSION

In this thesis we study the two dimensional nesting problems, to know: placement problem, knapsack problem, cutting stock problem, bin packing problem and two open dimensional problems (with one open dimension and two open dimension). The simpler variants of these problems are considered NP-Hard (see [Nielsen and Odgaard \(2003\)](#)). For each of these problems we propose and test several heuristic algorithms and mathematical programming models, which solve the problem or sub-problems of the problem.

Among the proposed algorithms we have significantly improved all the specialized literature, with contribution in quantitative terms (number of studies problems and methods proposed) and in terms qualitative (original results and papers published). We have improved for example: presenting the first results for the problem with two open dimensions, some of the best with an open dimension, finally we have found better solutions for all closed dimension problems: placement problem, knapsack problem, cutting stock problem and bin packing problem.

Although this thesis provides the pseudo-code for all the developed algorithms, we do not show the effort required for its implementation. It should be noted that thousands of lines of code in C language have been implemented for the calculation of no-fit polygon, no-fit raster, heuristics, meta-heuristics, and for mathematical models. All models, besides the proposed cuts and inequalities, were implemented in C ++ using the package IBM CPLEX Optimizer¹. In addition to these code, we developed a triangle decomposition algorithm in language C and we use the CGAL package to decompose nonconvex polygons with holes in the least number of possible convex polygons.

Then, in the Figure 59 we present an overview of the thesis. Introducing organization of thesis inside to first rectangle, within this rectangle we have five parts the these thesis, to know: 1) the introduction, 2) the heuristics methods, 3) the mathematical programming, 4) the conclusion and 5) the appendix. Inside of these parts we have the chapters, each chapter have breview details

¹ <https://goo.gl/chkMVj>

in order to remember the reader.

Figure 59 – My PhD document.

In Figure 59 inside of **Heuristic methods** we have 3 chapters, where each chapter represent one paper with one heuristic method. In Chapter 2, we have the metaheuristics Biased Random-Key Genetic Algorithm (BRKGA) for the open dimensional problem. The Chapter 3 proposes a Heuristic For Nesting Problem (H4NP), the algorithm is a generic heuristic for the problems with rectangular and irregular bins of fixed dimensions. In Chapter 4 we proposes a hybrid method for the irregular cutting stock problem with known demand and stochastic demand.

The part of **Mathematical programming** is the Chapter 5 and the Chapter 6. These chapters proposed new mixed integer programming models for open dimensional problems. In Chapter 5 we proposed a new constraints that divides the external area of no-fit polygon into convex disjoint regions. This chapter improved the model proposed by us in Cherri *et al.* (2016), this paper was published during the writing period of this doctoral thesis. In Chapter 6

we proposed change the domain of the continuous models for semi-discrete and discrete domain, obtain better or equal results in comparasion with the specialized literature in 96.77%.

7.0.1 Future work

For future work, some ideas are very promising and will be presented in this section. We intend to propose a matheuristic for two-dimensional nesting problems. The idea is to allocate one item at a time in the container until all items are allocated. The metaheuristic BRKGA is very good for determining the sequence of items to be allocated. Thus, the idea is to apply the BRKGA to determine the sequence in which the items will be allocated using mathematical models, in which case the BRKGA would determine the sequence that the items would be fixed in the container by means of some mathematical model.

With regard to mathematical models the big limitation is the lower bound. We have an idea to decompose the items in small items and use these new items to improved the lower bound. In addition to the study of lower bounds, another determinant factor for the quality of continuous position mathematical models is that currently no model takes advantage of the demand for items, i.e. all continuous models do not take advantage of the fact that some items have the same shape. The idea, in this sense, is to tie the demand of the items like it was done in the work of [Toledo *et al.* \(2013\)](#).

As for heuristic methods, we intend to explore the use of mathematical models to solve subproblems within heuristics. In other words, we intend to solve to optimality the problem of clustering items and to allow that these clusters be used in the problem, instead of the items, reducing its complexity. The idea it the use of mathematical models in structure of cluster of items, as can be seen in [Elkeran \(2013\)](#) for clusters of two items.

BIBLIOGRAPHY

ADAMOWICZ, M.; ALBANO, A. Nesting two-dimensional shapes in rectangular modules. **Computer-Aided Design**, v. 8, n. 1, p. 27 – 33, 1976. ISSN 0010-4485. Available in: <http://www.sciencedirect.com/science/article/pii/0010448576900063>. Citation on page 46.

ALBANO, A.; SAPUPPO, G. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. **IEEE Transactions on Systems, Man, and Cybernetics**, SMC-10, n. 5, p. 242–248, 1980. Citation on page 26.

ALEM, D. J.; MUNARI, P. A.; ARENALES, M. N.; FERREIRA, P. A. V. On the cutting stock problem under stochastic demand. **Annals of Operations Research**, v. 179, n. 1, p. 169–186, 2010. ISSN 1572-9338. Available in: <http://dx.doi.org/10.1007/s10479-008-0454-7>. Citations on pages 113, 114 e 115.

ALVAREZ-VALDES, R.; MARTINEZ, A.; TAMARIT, J. A branch & bound algorithm for cutting and packing irregularly shaped pieces. **International Journal of Production Economics**, v. 145, n. 2, p. 463 – 477, 2013. Citations on pages 16, 17, 37, 41, 42, 44, 89, 120, 121, 122, 123, 128, 130, 131, 134, 135, 136 e 138.

ALVAREZ-VALDES, R.; PARAION, A.; TAMARIT, J. A computational study of LP-based heuristic algorithms for two-dimensional guillotine cutting stock problems. **OR Spectrum**, v. 24, n. 2, p. 179 – 192, 2002. Citation on page 102.

ALVELOS, F.; SOUSA, A. de; SANTOS, D. Hybrid metaheuristics. In: _____. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. cap. Combining Column Generation and Metaheuristics, p. 285–334. ISBN 978-3-642-30671-6. Available in: http://dx.doi.org/10.1007/978-3-642-30671-6_11. Citation on page 102.

ALVES, C.; BRAS, P.; CARVALHO, J. M. V.; PINTO, T. New constructive algorithms for leather nesting in the automotive industry. **Computers and Operations Research**, v. 39, p. 1487–1505, 2012. Citations on pages 15, 35, 36 e 44.

ALVES, C.; CARVALHO, J. V. de. Accelerating column generation for variable sized bin-packing problems. **European Journal of Operational Research**, v. 183, n. 3, p. 1333 – 1352, 2007. ISSN 0377-2217. Available in: <http://www.sciencedirect.com/science/article/pii/S0377221706003080>. Citation on page 102.

ART, R. C. **An approach to the two dimensional irregular cutting stock problem**. Cambridge, Massachusetts, USA, 1966. Citation on page 26.

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**, 33, n. 12, p. 879–891, 2001. Citation on page 29.

BAKER, B.; JR, E.; RIVEST, R. Orthogonal packings in two dimensions. **SIAM Journal on Computing**, SIAM, v. 9, n. 4, p. 846–855, 1980. Citations on pages 32 e 134.

BALDACCI, R.; BOSCHETTI, M. A.; GANOVELLI, M.; MANIEZZO, V. Algorithms for nesting with defects. **Discrete Applied Mathematics**, v. 163, Part 1, n. 0, p. 17 – 33, 2014. Citations on pages 35, 36 e 44.

BARNHART, C.; JOHNSON, E. L.; NEMHAUSER, G. L.; SAVELSBERGH, M. W. P.; VANCE, P. H. Branch-and-price: Column generation for solving huge integer programs. **Operations Research**, v. 46, n. 3, p. 316–329, 1998. Available in: <<http://pubsonline.informs.org/doi/abs/10.1287/opre.46.3.316>>. Citation on page 102.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. **INFORMS Journal on Computing**, v. 6, n. 2, p. 154–160, 1994. Available in: <<http://dblp.uni-trier.de/db/journals/informs/informs6.html#Bean94>>. Citation on page 52.

BELOV, G.; SCHEITHAUER, G. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. **European Journal of Operational Research**, v. 141, n. 2, p. 274 – 294, 2002. ISSN 0377-2217. Available in: <<http://www.sciencedirect.com/science/article/pii/S037722170200125X>>. Citation on page 102.

BENNELL, J.; SCHEITHAUER, G.; STOYAN, Y.; ROMANOVA, T.; PANKRATOV, A. Optimal clustering of a pair of irregular objects. **Journal of Global Optimization**, Springer US, v. 61, n. 3, p. 497–524, 2015. ISSN 0925-5001. Available in: <<http://dx.doi.org/10.1007/s10898-014-0192-0>>. Citations on pages 46 e 66.

BENNELL, J. A.; OLIVEIRA, J. F. The geometry of nesting problems: A tutorial. **European Journal of Operational Research**, v. 184, n. 2, p. 397 – 415, 2008. Citations on pages 28, 30 e 47.

_____. A tutorial in irregular shape packing problems. **Journal of the Operational Research Society**, 60, p. S93–S105, 2009. Citations on pages 32, 44 e 45.

BENNELL, J. A.; SONG, X. A comprehensive and robust procedure for obtaining the no-fit-polygon using minkowski sums. **Computers and Operations Research**, v. 35, n. 1, p. 267–281, 2008. Citation on page 31.

_____. A beam search implementation for the irregular shape packing problem. **Journal of Heuristics**, v. 16, p. 167–188, 2010. Citations on pages 33, 62, 68, 75 e 110.

BIRGE, J. R.; LOUVEAUX, F. Introduction to stochastic programming. **New York: Springer**, ISBN: 978-1-4614-0236-7 (Print) 978-1-4614-0237-4 (Online). Citations on pages 113 e 114.

BIRGIN, E.; SOBRAL, F. Minimizing the object dimensions in circle and sphere packing problems. **Computers & Operations Research**, v. 35, n. 7, p. 2357 – 2375, 2008. ISSN 0305-0548. Available in: <<http://www.sciencedirect.com/science/article/pii/S0305054806002942>>. Citation on page 46.

BIRGIN, E. G.; LOBATO, R. D. Orthogonal packing of identical rectangles within isotropic convex regions. **Comput. Ind. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 59, n. 4, p. 595–602, nov. 2010. ISSN 0360-8352. Available in: <<http://dx.doi.org/10.1016/j.cie.2010.07.004>>. Citation on page 75.

BURKE, E.; HELLIER, R.; KENDALL, G.; WHITWELL, G. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. **Operations Research**, 54, n. 3, p. 587–601, 2006. Citations on pages 32, 45, 62, 64, 68 e 134.

_____. Complete and robust no-fit polygon generation for the irregular stock cutting problem. **European Journal of Operational Research**, v. 179, p. 27 – 49, 2007. Citations on pages 15 e 31.

_____. Irregular packing using the line and arc no-fit polygon. **Operational Research**, v. 58, p. 948–970, 2010. Citation on page 31.

BIAZEWICZ, J.; HAWRYLUK, P.; WALKOWIAK, R. Using a tabu search approach for solving the two-dimensional irregular cutting problem. **Annals of Operations Research**, Baltzer Science Publishers, Baarn/Kluwer Academic Publishers, v. 41, n. 4, p. 313–325, 1993. ISSN 0254-5330. Available in: <<http://dx.doi.org/10.1007/BF02022998>>. Citations on pages 19, 22, 65, 70 e 71.

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**, , p. 113–127, 2005. No citation.

CARRAVILLA, M. A.; RIBEIRO, C.; OLIVEIRA, J. F. Solving nesting problems with non-convex polygons by constraint logic programming. **International Transactions in Operational Research**, 10, p. 651–663, 2003. Citation on page 37.

CGAL, Computational Geometry Algorithms Library. [Http://www.cgal.org](http://www.cgal.org). No citation.

CHEN, D.; LIU, J.; FU, Y.; SHANG, M. An efficient heuristic algorithm for arbitrary shaped rectilinear block packing problem. **Computers & Operations Research**, v. 37, n. 6, p. 1068 – 1074, 2010. ISSN 0305-0548. Available in: <<http://www.sciencedirect.com/science/article/pii/S0305054809002299>>. Citation on page 44.

CHERRI, L. H.; CARRAVILLA, M. A.; TOLEDO, F. M. B. A model-based heuristic for the irregular stip packing. **Pesquisa Operacional**, p. 447–468, 2016. Citations on pages 37 e 134.

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**, v. 253, n. 3, p. 570 – 583, 2016. ISSN 0377-2217. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0377221716301370>>. Citations on pages 16, 41, 42, 45, 120, 121, 122, 124, 127, 128, 130, 131, 134, 135, 136, 137, 138 e 146.

CHVÁTAL1973 CHVÁTAL, v. Edmonds polytopes and a hierarchy of combinatorial problems. **Discrete Mathematics**, v. 4, n. 4, p. 305 – 337, 1973. ISSN 0012-365X. Available in: <<http://www.sciencedirect.com/science/article/pii/0012365X73901672>>.

CINTRA, G.; MIYAZAWA, F.; WAKABAYASHI, Y.; XAVIER, E. Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation. **European Journal of Operational Research**, v. 191, n. 1, p. 61 – 85, 2008. ISSN 0377-2217. Available in: <<http://www.sciencedirect.com/science/article/pii/S0377221707008831>>. Citations on pages 75 e 102.

COSTA, M. T.; GOMES, A. M.; OLIVEIRA, J. F. Heuristic approaches to large-scale periodic packing of irregular shapes on a rectangular sheet. **European Journal of Operational Research**, v. 192, n. 1, p. 29 – 40, 2009. ISSN 0377-2217. Available in: <<http://www.sciencedirect.com/science/article/pii/S0377221707009320>>. Citation on page 46.

CRISPIN, A.; CLAY, P.; TAYLOR, G.; BAYES, T.; REEDMAN, D. Genetic algorithm coding methods for leather nesting. **Applied Intelligence**, v. 23, p. 9–20, 2005. Citations on pages 35 e 36.

CUNINGHAME-GREEN, R. Geometry, shoemaking and the milk tray problem. **New Scientist** 1677, p. 50–53, 1989. Citations on pages 26 e 31.

DALALAH, D.; KHRAIS, S.; BATAINEH, K. Waste minimization in irregular stock cutting. **Journal of Manufacturing Systems**, v. 33, p. 27–40, 2014. Citations on pages 75, 85, 87 e 89.

DANTZIG, G. B.; WOLFE, P. Decomposition principle for linear programs. **Operations Research**, INFORMS, v. 8, n. 1, p. 101–111, 1960. ISSN 0030364X, 15265463. Available in: <<http://www.jstor.org/stable/167547>>. Citation on page 102.

DIGHE, R.; JAKIELA, M. J. Solving pattern nesting problems with genetic algorithms employing task decomposition and contact detection. **Evolutionary Computation**, v. 3, p. 239 – 266, 1996. Citation on page 89.

DOWSLAND, K. A.; DOWSLAND, W. B.; BENNELL, J. A. Jostle for position: Local improvement for irregular cutting patterns. **The Journal of the Operational Research Society**, v. 49, p. 647–658, 1998. Citations on pages 32 e 134.

DOWSLAND, K. A.; VAID, S.; DOWSLAND, W. B. An algorithm for polygon placement using a bottom-left strategy. **European Journal of Operational Research**, 141, p. 371–381, 2002. Citations on pages 15, 32, 45 e 134.

DYCKHOFF, H.; FINKE, U. **Cutting and packing in production and distribution: Typology and bibliography**. [S.l.]: Heidelberg: Springer-Verlag, 1992. Citation on page 26.

EGEBLAD, J.; NIELSEN, B. K.; ODGAARD, A. Fast neighborhood search for two- and three-dimensional nesting problems. **European Journal of Operational Research**, v. 183, n. 3, p. 1249–1266, 2007. Citations on pages 15, 34, 35 e 134.

ELKERAN, A. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. **European Journal of Operational Research**, v. 231, n. 3, p. 757 – 769, 2013. Citations on pages 15, 35, 36, 45, 62, 68, 119, 134 e 147.

ESCUDERO, L. F.; GARÍN, A.; PÉREZ, G. The value of the stochastic solution in multistage problems. **TOP**, v. 15, n. 1, p. 48–64, 2007. Citation on page 114.

FISCHETTI, M.; LUZZI, I. Mixed-integer programming models for nesting problems. **Journal of Heuristics**, v. 15, n. 3, p. 201–226, 2009. Citations on pages 37, 44, 45, 89, 120, 123 e 134.

FOWLER, R. J.; PATERSON, M.; TANIMOTO, S. L. Optimal packing and covering in the plane are np-complete. **Inf. Process. Lett.**, v. 12, n. 3, p. 133–137, 1981. Citation on page 26.

FURINI, F.; MALAGUTI, E.; DURÁN, R. M.; PERSIANI, A.; TOTH, P. A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. **European Journal of Operational Research**, v. 218, n. 1, p. 251 – 260, 2012. ISSN 0377-2217. Available in: <<http://www.sciencedirect.com/science/article/pii/S0377221711009374>>. Citation on page 102.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability; A Guide to the Theory of NP-Completeness**. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN 0716710455. Citation on page 44.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. **Operations Research**, v. 9, n. 6, p. 849–859, 1961. Available in: <<http://dx.doi.org/10.1287/opre.9.6.849>>. Citations on pages 26, 40, 75, 101, 102 e 105.

_____. A linear programming approach to the cutting-stock problem, part ii. **Operations Research**, v. 11, p. 863–888, 1963. Citations on pages 40, 75, 102 e 105.

_____. Multistage cutting stock problems of two and more dimensions. **Operations Research**, v. 13, n. 1, p. 94–120, 1965. Available in: <<http://dx.doi.org/10.1287/opre.13.1.94>>. Citation on page 102.

GOMES, M. A.; OLIVEIRA, J. F. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. **European Journal of Operational Research**, 171, n. 3, p. 811–829, 2006. Citations on pages 34, 37, 45, 89, 90, 120, 127 e 134.

_____. A 2-exchange heuristic for nesting problems. **European Journal of Operational Research**, v. 141, n. 2, p. 359 – 370, 2002. Citations on pages 15, 32, 34 e 45.

GONCALVES, J.; RESENDE, M. A biased random-key genetic algorithm for the unequal area facility layout problem. **European Journal of Operational Research**, v. 246, n. 1, p. 86 – 107, 2015. Citation on page 57.

GREENE, D. H. The decomposition of polygons into convex parts. **Computational Geometry**, v. 1, p. 235–259, 1983. No citation.

HEISTERMANN, J.; LENGAUER, T. The nesting problem in the leather manufacturing industry. **Annals of Operations Research**, v. 57, p. 147–173, 1995. Citation on page 35.

HOLLAND, J. **Adaptation in Natural and Artificial Systems**. Ann Arbor, MI, USA: University of Michigan Press, 1975. Citation on page 51.

HU, Y.; HASHIMOTO, H.; IMAHORI, S.; YAGIURA, M. Efficient implementations of construction heuristics for the rectilinear block packing problem. **Computers & Operations Research**, v. 53, p. 206 – 222, 2015. ISSN 0305-0548. Available in: <<http://www.sciencedirect.com/science/article/pii/S0305054814001798>>. Citation on page 84.

HU-YAO, L.; YUAN-JUN, H. Algorithm for 2d irregular-shaped nesting problem based on the nfp algorithm and lowest-gravity-center principle. **Journal of Zhejiang University SCIENCE A**, v. 7, p. 570–576, 2006. Citation on page 31.

IMAMICHI, T.; YAGIURA, M.; NAGAMOCCHI, H. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. **Discrete Optimization**, v. 6, n. 4, p. 345 – 361, 2009. Citations on pages 35 e 45.

JAKOBS, S. On genetic algorithms for the packing of polygons. **European Journal of Operational Research**, v. 88, n. 1, p. 165–181, jan. 1996. Citations on pages 34, 45 e 134.

JONES, D. R. A fully general, exact algorithm for nesting irregular shapes. **Journal of Global Optimization**, v. 59, p. 367–404, 2014. Citations on pages 15, 31, 47, 128 e 130.

KALLRATH, J. Cutting circles and polygons from area-minimizing rectangles. **Journal of Global Optimization**, Springer US, v. 43, n. 2-3, p. 299–328, 2009. ISSN 0925-5001. Available in: <http://dx.doi.org/10.1007/s10898-007-9274-6>. Citations on pages 46 e 66.

KALLRATH, J.; REBENNACK, S.; KALLRATH, J.; KUSCHE, R. Solving real-world cutting stock-problems in the paper industry: Mathematical approaches, experience and challenges. **European Journal of Operational Research**, v. 238, n. 1, p. 374 – 389, 2014. ISSN 0377-2217. Available in: <http://www.sciencedirect.com/science/article/pii/S0377221714002562>. No citation.

KANTOROVICH, L. V. Mathematical methods of organizing and planning production. **Management Science**, v. 6, n. 4, p. 366–422, 1960. Available in: <http://dx.doi.org/10.1287/mnsc.6.4.366>. Citation on page 101.

LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. **Econometrica**, v. 28, n. 3, p. 497–520, 1960. Available in: <http://www.jstor.org/stable/1910129>. Citation on page 102.

LEAO, A. A.; TOLEDO, F. M.; OLIVEIRA, J. F.; CARRAVILLA, M. A. A semi-continuous mip model for the irregular strip packing problem. **International Journal of Production Research**, v. 54, n. 3, p. 712–721, 2016. Citations on pages 37, 120, 134, 138, 141 e 142.

LEUNG, S. C.; LIN, Y.; ZHANG, D. Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. **Computers & Operations Research**, v. 39, n. 3, p. 678 – 686, 2012. Citation on page 35.

LOPEZ-CAMACHO, E.; OCHOA, G.; TERASHIMA-MARÍN, H.; BURKE, E. K. An effective heuristic for the two-dimensional irregular bin packing problem. **Annals of Operations Research**, v. 206, n. 1, p. 241–264, 2013. Citations on pages 75 e 76.

LÓPEZ-CAMACHO, E.; TERASHIMA-MARIN, H.; ROSS, P.; OCHOA, G. A unified hyper-heuristic framework for solving bin packing problems. **Expert Systems with Applications**, v. 41, n. 15, p. 6876 – 6889, 2014. ISSN 0957-4174. Available in: <http://www.sciencedirect.com/science/article/pii/S0957417414002668>. Citations on pages 76, 90 e 95.

LÜBBECKE, M. E.; DESROSIERS, J. Selected topics in column generation. **Operations Research**, v. 53, n. 6, p. 1007–1023, 2005. Citation on page 102.

MARTELLO, S.; MONACI, M. Models and algorithms for packing rectangles into the smallest square. **Computers & Operations Research**, v. 63, p. 161 – 171, 2015. ISSN 0305-0548. Available in: <http://www.sciencedirect.com/science/article/pii/S0305054815001161>. Citation on page 46.

MARTELLO, S.; MONACI, M.; VIGO, D. An exact approach to the strip-packing problem. **INFORMS Journal on Computing**, v. 15, n. 3, p. 310–319, 2003. No citation.

MARTINEZ-SYKORA, A.; ALVAREZ-VALDES, R.; BENNELL, J.; RUIZ, R.; TAMARIT, J. Matheuristics for the irregular bin packing problem with free rotations. **European Journal of Operational Research**, p. –, 2016. ISSN 0377-2217. Available in: <http://www.sciencedirect.com/science/article/pii/S0377221716307950>. Citations on pages 76 e 134.

MARTINS, T.; TSUZUKI, M. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. **Expert Systems with Applications**, v. 37, n. 3, p. 1955 – 1972, 2010. ISSN 0957-4174. Available in: <http://www.sciencedirect.com/science/article/pii/S0957417409006575>. Citations on pages 83, 85 e 89.

MIRHASSANI, S.; BASHIRZADEH, A. J. A grasp meta-heuristic for two-dimensional irregular cutting stock problem. **The International Journal of Advanced Manufacturing Technology**, Springer London, v. 81, n. 1-4, p. 455–464, 2015. ISSN 0268-3768. Available in: <http://dx.doi.org/10.1007/s00170-015-7107-1>. Citations on pages 59 e 77.

MUNDIM, L. R.; ANDRETTA, M.; QUEIROZ, T. A. A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. **Expert Systems with Applications**, v. 81, p. 358–371, 2017. ISSN 0957-4174. Available in: <http://dx.doi.org/10.1016/j.eswa.2017.03.059>. Citations on pages 39 e 76.

MUNDIM, L. R.; ANDRETTA, M.; OLIVEIRA, J. F.; CARRAVILLA, M. A. H4np, a general heuristic for two-dimensional nesting problems with fixed size containers. 13th ESICUP Meeting, 2016. Citation on page 40.

MUNDIM, L. R.; ANDRETTA, M.; QUEIROZ, T. A. Aplicando o algoritmo genético de chaves aleatórias viciadas em um problema de corte com itens irregulares. **Blucher Mathematical Proceedings**, v. 1, n. 1, p. 195 – 203, 2015. ISSN em b-reve. Available in: www.proceedings.blucher.com.br/article-details/11901. Citation on page 39.

MUNDIM, L. R.; CHERRI, L. H.; ANDRETTA, M.; TOLEDO, F. M. B.; CARRAVILLA, M. A.; OLIVEIRA, J. F. Um modelo inteiro misto com trigonometria direta para o corte de polígonos convexos aplicado à indústria. **XLVII SBPO - Simpósio Brasileiro de Pesquisa Operacional**, p. 4075–4086, 2015. Citation on page 41.

NIELSEN, B.; ODGAARD, A. Fast neighborhood search for the nesting problem. **Relatório técnico, DIKU**, p. 9–10, 2003. Citation on page 145.

NONÅS, S. L.; THORSTENSON, A. Solving a combined cutting-stock and lot-sizing problem with a column generating procedure. **Computers & Operations Research**, v. 35, n. 10, p. 3371 – 3392, 2008. ISSN 0305-0548. Part Special Issue: Search-based Software Engineering. Available in: <http://www.sciencedirect.com/science/article/pii/S0305054807000706>. Citation on page 102.

OLIVEIRA, J. F.; FERREIRA, J. Algorithms for nesting problems, applied simulated annealing. In Vidal, R.V.V. (Ed., **Lecture Notes in Economics and Maths Systems**), 396, p. 255–274, 1993. Citations on pages 15, 28, 29 e 30.

OLIVEIRA, J. F.; GOMES, A. M.; FERREIRA, J. S. Topos - a new constructive algorithm for nesting problems. **OR Spektrum**, v. 22, p. 263–284, 2000. Citations on pages 15, 32, 33, 34 e 45.

PINHEIRO, P. R.; JÚNIOR, B. A.; SARAIVA, R. D. A random-key genetic algorithm for solving the nesting problem. **International Journal of Computer Integrated Manufacturing**, v. 0, n. 0, p. 1–7, 2015. Available in: <http://dx.doi.org/10.1080/0951192X.2015.1036522>. Citations on pages 45, 62, 64 e 68.

QUEIROZ, T. A. de; MIYAZAWA, F. K.; WAKABAYASHI, Y.; XAVIER, E. C. Algorithms for 3d guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. **Computers & Operations Research**, v. 39, n. 2, p. 200 – 212, 2012. ISSN 0305-0548. Citation on page 44.

RIBEIRO, C.; CARRAVILLA, M. A.; OLIVEIRA, J. F. Applying constraint logic programming to the resolution of nesting problems. **Pesquisa Operacional**, 19, n. 2, p. 239–247, 1999. Citation on page 128.

ROCHA, P.; RODRIGUES, R.; GOMES, A. M.; TOLEDO, F. M.; ANDRETTA, M. Two-phase approach to the nesting problem with continuous rotations. **IFAC-PapersOnLine**, v. 48, n. 3, p. 501 – 506, 2015. ISSN 2405-8963. Available in: <<http://www.sciencedirect.com/science/article/pii/S2405896315003705>>. Citation on page 47.

RODRIGUES, M. O.; TOLEDO, F. M. A clique covering MIP model for the irregular strip packing problem. **Computers & Operations Research**, p. –, 2016. ISSN 0305-0548. Available in: <<http://www.sciencedirect.com/science/article/pii/S0305054816302702>>. Citations on pages 37 e 134.

RODRIGUES, M. O.; CHERRI, L. H.; MUNDIM, L. R. MIP models for the irregular strip packing problem: new symmetry breaking constraints. **Applied mathematical programming and Modelling**, 2016, Brno, Czech Republic. APMOD conference proceedings, 2016. Citation on page 41.

ROSS, P.; SCHULENBURG, S.; MARÍN-BLÁZQUEZ, J. G.; HART, E. Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002. (GECCO '02), p. 942–948. ISBN 1-55860-878-8. Citation on page 75.

SANTORO, M. C.; LEMOS, F. K. Irregular packing: Milp model based on a polygonal enclosure. **Annals of Operations Research**, v. 235, n. 1, p. 693–707, 2015. Citations on pages 120, 127, 128 e 130.

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**, v. 44, n. 8, p. 766 – 777, 2012. No citation.

SATO, A. K.; MARTINS, T. de C.; TSUZUKI, M. de S. G. A pairwise exact placement algorithm for the irregular nesting problem. **International Journal of Computer Integrated Manufacturing**, v. 0, n. 0, p. 1–13, 2015. Available in: <<http://dx.doi.org/10.1080/0951192X.2015.1033018>>. Citations on pages 46, 62 e 68.

SEGENREICH, S.; BRAGA, L. Optimal nesting of general plane figures - A Monte-Carlo heuristic approach. **Computers & Graphics**, 10, n. 3, p. 229–237, 1986. Citations on pages 15, 21, 29, 48, 64, 65 e 70.

SONG, X.; BENNELL, J. A. Column generation and sequential heuristic procedure. **J Oper Res Soc**, Palgrave Macmillan, v. 65, n. 7, p. 1037–1052, Jul 2014. ISSN 0160-5682. Available in: <<http://dx.doi.org/10.1057/jors.2013.44>>. Citations on pages 19, 23, 75, 90, 91, 92, 103, 109, 110, 111 e 115.

SPEARS, V. M.; JONG, K. A. D. On the virtues of parameterized uniform crossover. In: **In Proceedings of the Fourth International Conference on Genetic Algorithms**. [S.l.: s.n.], 1991. p. 230–236. Citation on page 52.

STOYAN, Y.; SCHEITHAUER, G.; GIL, N.; ROMANOVA, T. ϕ -function for complex 2D-objects. **4OR Quarterly Journal of the Belgian, French and Italian Operations Research Societies**, 2, p. 69–84, 2004. Citation on page 29.

STOYAN, Y.; TERNO, J.; SCHEITHAUER, G.; GIL, N.; ROMANOVA, T. ϕ -function for primary 2D-objects. **Studia Informatica Universalis**, 2, n. 1, p. 1–32, 2001. Citation on page 29.

STOYAN, Y. G.; PATSUK, V. A method of optimal lattice packing of congruent oriented polygons in the plane. **European Journal of Operational Research**, v. 124, n. 1, p. 204 – 216, 2000. ISSN 0377-2217. Available in: <http://www.sciencedirect.com/science/article/pii/S0377221799001150>. Citation on page 46.

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**, Springer US, v. 179, n. 1, p. 369–392, 2010. ISSN 0254-5330. Available in: <http://dx.doi.org/10.1007/s10479-008-0475-2>. Citations on pages 16, 75, 76, 90, 93 e 95.

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**, v. 145, n. 2, p. 478 – 487, 2013. Citations on pages 19, 20, 21, 31, 37, 45, 47, 48, 49, 50, 59, 60, 120, 134, 138, 139, 140, 141 e 147.

TOSO, R.; RESENDE, M. A c++application programming interface for biased random-key genetic algorithms. **Optimization Methods and Software**, v. 30, n. 1, p. 81–93, 2015. Available in: <http://dx.doi.org/10.1080/10556788.2014.890197>. Citations on pages 52, 55 e 57.

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**, v. 16, n. 6, p. 661–683, 2009. Citations on pages 35, 45 e 134.

VALLE, A. M. D.; 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**, v. 39, n. 16, p. 12589 – 12598, 2012. ISSN 0957-4174. Available in: <http://www.sciencedirect.com/science/article/pii/S0957417412007233>. Citations on pages 19, 23, 46, 75, 84, 85, 87, 88, 89, 90, 91, 92, 102, 109, 112 e 113.

VANDERBECK, F. A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem. **Manage. Sci.**, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 47, n. 6, p. 864–879, jun. 2001. ISSN 0025-1909. Available in: <http://dx.doi.org/10.1287/mnsc.47.6.864.9809>. Citation on page 102.

WÄSCHER, G.; HAUßNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, v. 183, n. 3, p. 1109 – 1130, 2007. Citations on pages 15, 26, 27, 74, 75 e 133.

YANG, H.-H.; LIN, C.-L. On genetic algorithms for shoe making nesting – a taiwan case. **Expert Systems with Applications**, v. 36, n. 2, Part 1, p. 1134 – 1141, 2009. ISSN 0957-4174. Available in: <<http://www.sciencedirect.com/science/article/pii/S0957417407005192>>. Citation on page 44.

YUPING, Z.; SHOUWEI, J.; CHUNLI, Z. A very fast simulated re-annealing algorithm for the leather nesting problem. **International Journal of Advanced Manufacturing Technology**, p. 1113–1118, 2005. Citation on page 35.

DRAWING PATTERNS IN C&P PROBLEMS USING $\text{\LaTeX}$ /TIKZ

A.1 Introduction

Cutting and Packing (C&P) problems have been studied for many years now and all researchers have faced the difficulty of depicting the cutting and packing patterns in a way that is at one time accurate and with high graphic quality. In fact, geometry is a distinctive characteristic of almost all C&P problems when compared with other combinatorial optimization problems. The first papers represented cutting patterns by combinations of characters, hand-made drawings or, more rarely, by drawings generated by computer codes programmed from scratch. Nowadays the available technology is very different, and high quality C&P solutions can be quickly depicted with generic software tools, allowing for the easy analysis of these solutions and therefore for the easy debugging of computer codes. Moreover, in addition to style, a good design brings credibility to your work.

This tutorial aims to provide a simple template for authors dealing with C&P problems who use $\text{\LaTeX}$ to write their manuscripts. Therefore, we will show how to draw 1D, 2D and 3D shapes with TikZ, a LaTeX package for drawing and creating graphics available at (<https://goo.gl/6kt5YP>).

A.2 How to use $\text{\LaTeX}$ /TikZ

To use $\text{\LaTeX}$ We may resort to an online editor (Codecogs.com, Overleaf.com, Latexlab.org, Sharelatex.com, etc...) or install an integrated $\text{\LaTeX}$ environment on the computer (e.g. Linux: <https://goo.gl/re9H34>, Windows: <https://goo.gl/ZMfB0R>, Mac: <https://goo.gl/Z3gJYq>).

$\text{\LaTeX}$ documents are commonly written inside a block like the one below.

```

1: \documentclass{article}
2: \begin{document}
   ...
3: \end{document}

```

To use TikZ we just have to add the respective package (line 2).

```

1: \documentclass{article}
2: \usepackage{tikz}
3: \begin{document}
   ...
4: \end{document}

```

To use TikZ commands an environment as the one below has to be used. TikZ commands can then be used inside such an environment:

```

\begin{tikzpicture}
   ...
\end{tikzpicture}

```

For a gentle introduction to the use of TikZ an interactive editor may be used. With this editor the user may both write TikZ code and see the result automatically displayed on the screen and draw the figures and see the TikZ code generated on the fly. For this purpose we recommend TikzEdt (<https://goo.gl/VmDJ08>) for Windows, KtikZ (<https://goo.gl/6Rg41k>) for Ubuntu/Debian, and TikZit (<https://goo.gl/PqL2XYt>) for Windows/Linux/iOS.

A.3 Basic components

Line:

```

\begin{tikzpicture}
  \draw (0,0)--(2,1);
\end{tikzpicture}

```

Circles and arrows. The % symbol is used for comments.

```

\begin{tikzpicture}
  \draw (0,0) circle[radius=4pt]; %empty circle
  \draw[fill=red] (1,0) circle[radius=8pt]; %red circle
  \draw[->] (2,0)--(3,2); %arrow
  \draw[<->, shift={(1,0)}] (2,0)--(3,2); %arrow
\end{tikzpicture}

```

You have direct access to the following colors: red ■, green ■, blue ■, gray ■, white, etc.

A.4 Drawing primitive structures

Drawing polygons is the basis to represent solutions in the majority of the C&P problems. In TikZ to draw a convex or non-convex polygon we need to provide the list of clockwise or counterclockwise vertices.

The following polygon is a rectangle.

```
\begin{tikzpicture}
  \draw (0,0)--(1,0)--(1,1)--(0,1)--cycle;
\end{tikzpicture}
```

We can draw a container:

```
\begin{tikzpicture}
  \draw (1,1)--(0,1) - - (0,0)--(1,0); % container lines
  \draw[dashed] (1,1)--(1,0); % limit line
\end{tikzpicture}
```

A gray non-convex polygon.

```
\begin{tikzpicture}
  \draw[fill = gray] (0, 5)--(2, 5)--(2, 2)--(12, 2)--(12, 5)
    --(14, 5)--(14, 0)--(0, 0) --cycle;
\end{tikzpicture}
```

Another important concept is opacity.

```
\begin{tikzpicture}
  \draw[fill=gray,opacity = 0.2]
    (0, 0)--(6, -6)--(12, 0)--(6, 6)--cycle;
\end{tikzpicture}
```

Opacity is a fantastic tool to visually verify overlap.

```
\begin{tikzpicture}
  \draw[fill=, fill opacity=0.3] (0.074, -0.305)--(0.870, -0.186)
    --(1.666, -0.305)--(1.740, -0.180)--(0.870, -0.000)--(0.000, -0.180)--cycle;
  \draw[fill=, fill opacity=0.3, shift={(0,0.1)}] (3.000, -0.296)--(2.034, -0.352)
    --(1.017, -0.21) --(0.815, -0.448)--(0.266, -0.427)--(0.000, -0.977)--(0.181, -1.11)
    --(0.181, -1.570)--(0.000, -1.703)--(0.266, -2.253)--(0.815, -2.232)
    --(1.017, -2.47)--(2.034, -2.328)--(3.000, -2.384)--cycle;
\end{tikzpicture}
```

With the concepts presented until now, we can represent one-dimensional and two-dimensional solutions of C&P in a simple way:

```
\begin{tikzpicture}
  \draw (0,0)--(13,0)--(13,4)--(0,4)--cycle;
  \draw[fill = green, dashed, opacity=0.5]
    (0,0)--(4,0)--(4,4)--(0,4)--cycle;
  \draw[fill = blue, dashed, opacity=0.5, shift={(2,0)}]
    (0,0)--(4,0)--(4,4)--(0,4)--cycle;
\end{tikzpicture}
```

Or most elaborate drawings:

```
\begin{tikzpicture}
  \draw (0,4.5) ellipse (1 and 0.5);
  \draw (-1,4.5)--(-1,-2);
  \draw (1,-2)--(1,4.5);
  \draw (-1,-2) arc (180:360:1 and 0.5);
  \draw [dashed] (-1,-2) arc (180:360:1 and -0.5);
  \fill [opacity=0.1] (-1,4.5)--(-1,-2) arc (180:360:1 and 0.5)
    --(1,4.5) arc (0:180:1 and -0.5);
\end{tikzpicture}
```

To draw box in three-dimensional case:

```
\begin{tikzpicture}
  \draw[fill=gray, fill opacity=0.3] (20, 0, 50) --(20, 0, 0)--(20, 30, 0)
    --(0, 30, 0) --(0, 30, 50) --(0, 0, 50) --(20, 0, 50)
    --(20, 30, 50) --(20, 30, 0) --(20, 30, 50) --(0, 30, 50) ;
  \draw[dashed] (0, 0, 0)--(20, 0, 0)--(0, 0, 0)
    --(0, 30, 0)--(0, 0, 0) --(0, 0, 50) ;
\end{tikzpicture}
```

A.5 Conclusions

We have presented practical examples of how to draw objects in 1D, 2D and 3D in the C&P problems context. For more details about this tool: <https://goo.gl/GhPXPW>.

A.6 Drawings and code

One dimensional:

```

\begin{tikzpicture}
  \begin{scope}[rotate=270, shift={(0,0)}]
    \draw (0,4.5) ellipse (1 and 0.5);
    \draw (-1,4.5)--(-1,-4);
    \draw (1,-4)--(1,4.5);
    \draw (-1,-4) arc (180:360:1 and 0.5);
    \draw [dashed] (-1,-4) arc (180:360:1 and -0.5);
    \fill [opacity=0.1] (-1,4.5)--(-1,-4) arc (180:360:1 and 0.5)--(1,4.5) arc (0:180:1 and -0.5);
  \end{scope}
  \begin{scope}[rotate=270]
    \draw[fill = blue, opacity=0.2] (0,4) ellipse (1 and 0.5);
    \draw [dashed] (-1,4) arc (180:360:1 and -0.5);
    \draw (-1,0)--(-1,-2);
    \draw (1,-2)--(1,0);
    \fill [blue,opacity=0.5] (-1,0)--(-1,-2) arc (180:360:1 and 0.5)--(1,4) arc (0:180:1 and -0.5);
  \end{scope}
  \begin{scope}[rotate=270, shift={(0,-2)}]
    \draw[dashed] (0,0) ellipse (1 and 0.5);
    \draw (-1,0)--(-1,-2);
    \draw (1,-2)--(1,0);
    \fill [green,opacity=0.5] (-1,0)--(-1,-2) arc (180:360:1 and 0.5)--(1,0) arc (0:180:1 and -0.5);
    \draw (-1,0) arc (180:360:1 and 0.5);
    \draw (-1,6) arc (180:360:1 and 0.5);
  \end{scope}
  \begin{scope}[rotate=270, shift={(3,0)}]
    \draw (0,4.5) ellipse (1 and 0.5);
    \draw (-1,4.5)--(-1,-2);
    \draw (1,-2)--(1,4.5);
    \draw (-1,-2) arc (180:360:1 and 0.5);
    \fill [opacity=0.1] (-1,4.5)--(-1,-2) arc (180:360:1 and 0.5)--(1,4.5) arc (0:180:1 and -0.5);
    \draw (-1,2) arc (180:360:1 and 0.5);
  \end{scope}
  \begin{scope}[rotate=270, shift={(3,-2)}]
    \draw (-1,0)--(-1,-2);
    \draw (1,-2)--(1,0);
    \draw (-1,-2) arc (180:360:1 and 0.5);
    \draw [dashed] (-1,-2) arc (180:360:1 and -0.5);
    \fill [blue,opacity=0.5] (-1,0)--(-1,-2) arc (180:360:1 and 0.5)--(1,0) arc (0:180:1 and -0.5);
  \end{scope}
  \begin{scope}[rotate=270, shift={(3,0)}]
    \draw[fill = red, opacity=0.2] (0,2) ellipse (1 and 0.5);
    \draw [dashed] (-1,2) arc (180:360:1 and -0.5);
    \draw (-1,1)--(-1,-2);
    \draw (1,-2)--(1,1);
    \draw (-1,-2) arc (180:360:1 and 0.5);
    \draw [dashed] (-1,-2) arc (180:360:1 and -0.5);
    \fill [red,opacity=0.5] (-1,2)--(-1,-2) arc (180:360:1 and 0.5)--(1,2) arc (0:180:1 and -0.5);
  \end{scope}
\end{tikzpicture}

```

Two-dimensional:

```
\begin{tikzpicture}
  \foreach \x in {0,...,7}{
    \draw[shift={(\x,0)}, fill=, opacity=0.7] (0,2)--(1,2)--(1,0)--(0,0)--cycle;
  }
  \foreach \x in {2,4,6}{
    \draw[shift={(\x,5)}, fill=, opacity=0.1, rotate=90] (0,2)--(0.7,2)--(0.7,0)--(0,0)--cycle;
    \foreach \y in {2,3,5}{
      \draw[shift={(\x,\y)}, fill=, opacity=0.4, rotate=90] (0,2)--(1.5,2)--(1.5,0)--(0,0)--cycle;
    }
  }
  \foreach \x in {6,6.7,7.4}{
    \foreach \y in {2,4}{
      \draw[shift={(\x,\y)}, fill=, opacity=0.1] (0,2)--(0.7,2)--(0.7,0)--(0,0)--cycle;
    }
  }
  \draw[draw= red] (0,0)--(8.5,0)--(8.5,6)--(0,6)--(0,0);
  \draw[line width=0.6mm,shift={(0,2)},draw= blue, dashed] (-1,0)--(9.5,0);
  \draw[line width=0.3mm,shift={(4,0)},draw= blue, dashed] (2,-1)--(2,6.5);
  \draw[line width=0.1mm,shift={(0,5)},draw= blue, dashed] (-1,0)--(9.5,0);
\end{tikzpicture}
```

[illegible]

Three-dimensional:

```

\begin{tikzpicture}
  \draw[fill=gray, fill opacity=0.3] (20, 0, 50) --(20, 0, 0)--(20, 30, 0)
    --(0, 30, 0) --(0, 30, 50) --(0, 0, 50) --(20, 0, 50)
    --(20, 30, 50) --(20, 30, 0) --(20, 30, 50) --(0, 30, 50) ;
  \draw[dashed ] (0, 0, 0)--(20, 0, 0)--(0, 0, 0)
    --(0, 30, 0)--(0, 0, 0) --(0, 0, 50) ;
  \draw[fill=green, fill opacity=0.2] (10, 0, 10)--(10, 0, 0)--(10, 10, 0)
    --(0, 10, 0) --(0, 10, 10) --(0, 0, 10) --(10, 0, 10)
    --(10, 10, 10) --(10, 10, 0) --(10, 10, 10) --(0, 10, 10) ;
  \draw[fill=red, fill opacity=0.2, shift={(10,0)}] (10, 0, 10)--(10, 0, 0)--(10, 10, 0)
    --(0, 10, 0) --(0, 10, 10) --(0, 0, 10) --(10, 0, 10)
    --(10, 10, 10) --(10, 10, 0) --(10, 10, 10) --(0, 10, 10) ;
  \draw[fill=blue, fill opacity=0.2, shift={(0,10)}] (10, 0, 10)--(10, 0, 0)--(10, 10, 0)
    --(0, 10, 0) --(0, 10, 10) --(0, 0, 10) --(10, 0, 10)
    --(10, 10, 10) --(10, 10, 0) --(10, 10, 10) --(0, 10, 10) ;
\end{tikzpicture}

```