

Efficient Heuristics for Two-Dimensional Cutting and Packing Problems

Óscar António Maia de Oliveira

Supervisor

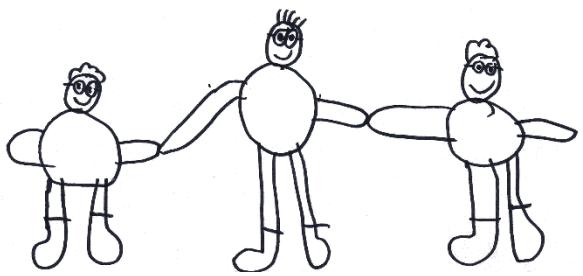
Professor Elsa Marília da Costa Silva

Co-supervisor

Professor Dorabela Regina Chiote Ferreira Gamboa

Submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Engineering and Industrial Management.

2019



To Rúben and Martim.

For endless comprehension, thank you Mónica.

For endless joy, thank you Rúben and Martim.

For endless kindness, thank you mother.

*For endless patience, support and guidance, I express my gratitude
to my advisors, Professors Dorabela Gamboa and Elsa Silva.*

Abstract

Cutting and packing problems are complex combinatorial optimization problems that have gained a lot of attention from the scientific community. These problems aim to assign to one or more objects a set of items to either be cut or packed. From this class of problems, this thesis focusses on those aiming to achieve the layout that maximizes the value of the items assigned to one object, and on those aiming to assign all items to objects using as fewer objects as possible.

The importance given to the solution of these problems is, mainly, due to their relevance in diversified areas, such as economics, industry, health, among many others.

Exact methods ensure the achievement of the problem's optimal solution at the expense of, usually, high computational resources, justifying the exploration of alternative approaches, such as heuristics, that can obtain high-quality solutions with lower resources.

This thesis is divided into three parts. The first part, aside from a comprehensive introduction to solution methods, discusses the relevant contributions given by the papers included in the other two parts of this thesis.

The second part is composed by three papers. The first paper gives a deep dive into the cutting and packing subject and reviews the solution methods for the two-dimensional guillotine and non-guillotine problems. The second paper is a survey on the datasets used by researchers when evaluating the solution methods for the two-dimensional rectangular cutting and packing problems. The third paper presents the resources that were developed aiming to help the development of our research and that we have made available for the scientific community.

The third part contains two papers presenting the solution methods proposed for two-dimensional cutting and packing problems. The first paper presents heuristics to solve non-guillotine problems, while the second deals with problems that consider guillotine cut constraints.

The proposed heuristics to solve guillotine and non-guillotine two-dimensional cutting and packing problems share the same base concept. The idea behind the design of these heuristics

was that, if an ordering (to cut or pack items) generates a good solution, it may be the case that slight changes to this ordering generate a better solution. And, if the changes do not improve the current solution, the incremental introduction of more changes to the base ordering could allow the exploration of more diverse regions of the solution space.

The computational results demonstrate that the proposed heuristics attain good results when compared with other solution methods, obtaining constantly good quality solutions in reduced computational times, validating their effectiveness and robustness.

Keywords: Two-dimensional, Rectangular, Guillotine, Non-guillotine, Cutting Problem, Packing Problem, Heuristics

Resumo

Os problemas de corte e empacotamento são problemas complexos de otimização combinatória que têm sido foco de muita atenção por parte da comunidade científica. Estes problemas visam atribuir a um ou mais objetos um conjunto de itens de modo a serem cortados ou empacotados. Desta família de problemas, esta tese concentra-se naqueles que procuram atingir o *layout* que maximiza o valor dos itens atribuídos a um objeto, e nos problemas que pretendem atribuir todos os itens minimizando o número de objetos utilizados.

A importância dada à resolução desses problemas deve-se, principalmente, à sua relevância em áreas diversificadas, como economia, indústria, saúde, entre muitas outras.

Os métodos exatos garantem a obtenção da solução ótima do problema, geralmente, à custa de elevados recursos computacionais, justificando deste modo a investigação de abordagens alternativas, como heurísticas, que possam obter soluções de alta qualidade utilizando menos recursos.

Esta tese é dividida em três partes. A primeira parte, além de uma introdução aos métodos de resolução, discute as contribuições relevantes dadas pelos artigos incluídos nas outras duas partes desta tese.

A segunda parte é composta por três artigos. O primeiro artigo explora o tópico de corte e empacotamento e os métodos de resolução para problemas bidimensionais guilhotinados e não guilhotinados. O segundo artigo apresenta os conjuntos e os geradores de instâncias que podem ser encontrados na literatura e que foram usados pelos investigadores para avaliar os métodos de resolução propostos para os problemas de corte e empacotamento. O terceiro artigo apresenta os recursos que foram criados para suportar a nossa investigação e que disponibilizamos para serem usados pela comunidade científica.

A terceira parte é constituída por dois artigos apresentando os métodos de resolução propostos para problemas de corte e empacotamento bidimensionais considerando cortes guilhotinados e não guilhotinados. A conceito por detrás das heurísticas propostas é que, se uma sequência de itens (a serem cortados ou empacotados) gera uma boa solução, pequenas mudanças nessa sequência podem levar a uma melhor solução. E que, se essas mudanças não

melhorarem a solução, a introdução de incrementalmente mais mudanças na sequência de itens pode permitir a exploração de regiões mais diversificadas do espaço de soluções.

Os resultados computacionais demonstram que as heurísticas propostas alcançam bons resultados quando comparadas com outros métodos de solução obtendo soluções de boa qualidade em tempos computacionais reduzidos, validando sua eficácia e robustez.

Palavras-chave: Bidimensional, Retangular, Guilhotinados, Não Guilhotinados, Problema da Corte, Problemas de Empacotamento, Heurísticas

Contents

Abstract	vii
Resumo	ix
Contents.....	xi
List of Tables	xiii
List of Figures	xix
Part I – Thesis	1
1. Introduction	2
1.1. Solution Methods.....	3
1.2. Background and Motivation	7
2. Thesis Contribution	7
3. Conclusions	15
References	16
Part II – Research Resources	19
A Review of Solution Approaches for Two-dimensional Cutting and Packing Problems	21
1. Introduction	21
2. Cutting and Packing Problems	22
3. Solution Approaches	30
4. Conclusion.....	53
Appendix A.....	54
References	56
Datasets and Generators for Two-dimensional Cutting and Packing Problems	69
1. Introduction	69
2. Literature Benchmarks.....	70

3. Instance Generators.....	85
References	86
Resources for Two-dimensional (and Three-dimensional) Cutting and Packing Solution Methods Research	93
1. Introduction	93
2. Datasets	95
3. GUI for instance and cutting plan visualisation	97
4. Website	103
5. Conclusion.....	110
References	110
Part III – Heuristics	113
Adaptive Sequence-based Heuristic for Two-Dimensional Non-Guillotine Packing Problems	115
1. Introduction	115
2. Literature Review.....	116
3. Adaptive Sequence-based Heuristic (ASH)	119
4. Computational Results.....	121
5. Conclusion.....	127
References	127
Adaptive Sequence-based Heuristic for Two-Dimensional Guillotine Cutting Problems..	131
1. Introduction	131
2. Literature Review.....	133
3. Adaptive Sequence-based Heuristic (ASH)	138
4. Computational Results.....	141
5. Conclusion.....	150
References	151

List of Tables

Part II – Research Resources

A Review of Solution Approaches for Two-dimensional Cutting and Packing Problems

Table 1. Typology of Dyckhoff - Criteria.	29
Table 2. Typology of Wäscher <i>et al.</i> (2007) - Criteria.	29
Table 3. Typology of Wäscher <i>et al.</i> (2007).	30
Table 4. Solution approaches for cutting and packing problems.	54

Datasets and Generators for Two-dimensional Cutting and Packing Problems

Table 1. Datasets for cutting and packing problems.	70
Table 2. Features of the instances in AA.....	72
Table 3. Features of the instances in AB.....	72
Table 4. Features of the instances in ABM.	73
Table 5. Features of the instances in ABMR.	73
Table 6. Features of the instances in AH.	73
Table 7. Features of the instances in ASSORT.	73
Table 8. Features of the instances in ATP.	73
Table 9. Features of the instances in B.	73
Table 10. Features of the instances in BABU.	74
Table 11. Features of the instances in BABU2.	74
Table 12. Features of the instances in BENG.	74
Table 13. Features of the instances in BKW.	74
Table 14. Features of the instances in BRPB.	74
Table 15. Features of the instances in CGCUT.	74
Table 16. Features of the instances in CH.	74
Table 17. Features of the instances in CHL.	75
Table 18. Features of the instances in CJCM.	75
Table 19. Features of the instances in CLASS.	75

Table 20. Features of the instances in CMWX.....	75
Table 21. Features of the instances in CUI.	75
Table 22. Features of the instances in CWL.....	75
Table 23. Features of the instances in CY.....	75
Table 24. Features of the instances in CZ.	76
Table 25. Features of the instances in D.....	76
Table 26. Features of the instances in DOWSLAND.....	76
Table 27. Features of the instances in EL-AAL.....	76
Table 28. Features of the instances in EP2.	76
Table 29. Features of the instances in FHZ	76
Table 30. Features of the instances in FO.....	77
Table 31. Features of the instances in GARD.....	77
Table 32. Features of the instances in GCUT.....	77
Table 33. Features of the instances in HADCHR.....	77
Table 34. Features of the instances in HERZ.....	77
Table 35. Features of the instances in HIFI1997a.....	77
Table 36. Features of the instances in HIFI1997b.....	78
Table 37. Features of the instances in HIFI2001.....	78
Table 38. Features of the instances in HOPPER.....	78
Table 39. Features of the instances in HT2001a.....	78
Table 40. Features of the instances in HT2001b.	78
Table 41. Features of the instances in HZ1.....	78
Table 42. Features of the instances in HZ2.....	79
Table 43. Features of the instances in IS.	79
Table 44. Features of the instances in IYUAI.	79
Table 45. Features of the instances in JAKOBS.....	79
Table 46. Features of the instances in JSL.....	79
Table 47. Features of the instances in KORF.	79
Table 48. Features of the instances in KR.	79
Table 49. Features of the instances in LC.	80
Table 50. Features of the instances in LCT.	80
Table 51. Features of the instances in LYT.....	80

Table 52. Features of the instances in MA	80
Table 53. Features of the instances in MAA	80
Table 54. Features of the instances in MB.....	80
Table 55. Features of the instances in MB2D.....	81
Table 56. Features of the instances in MG	81
Table 57. Features of the instances in MP.....	81
Table 58. Features of the instances in MWV.....	81
Table 59. Features of the instances in NGCUT.....	81
Table 60. Features of the instances in NGCUTAP	81
Table 61. Features of the instances in NGCUTCON	82
Table 62. Features of the instances in NGCUTFS.....	82
Table 63. Features of the instances in NHU.....	82
Table 64. Features of the instances in OF.....	82
Table 65. Features of the instances in OKP.	82
Table 66. Features of the instances in ONV.....	82
Table 67. Features of the instances in PGD.	82
Table 68. Features of the instances in PO.	83
Table 69. Features of the instances in RAND.....	83
Table 70. Features of the instances in RSS.	83
Table 71. Features of the instances in SCP.	83
Table 72. Features of the instances in SCPL.	83
Table 73. Features of the instances in SPIEKSMA.....	83
Table 74. Features of the instances in SS.	83
Table 75. Features of the instances in SSOOYKI.	84
Table 76. Features of the instances in STS.....	84
Table 77. Features of the instances in TEST.	84
Table 78. Features of the instances in VAG.....	84
Table 79. Features of the instances in VASSILIADIS.....	84
Table 80. Features of the instances in VENKATESWARLU.	84
Table 81. Features of the instances in WANG.	84
Table 82. Features of the instances in WV.	85
Table 83. Features of the instances in WVINT.....	85

Table 84. Features of the instances in WWD.....	85
Table 85. Features of the instances in ZDF.....	85
Table 86. Lodi <i>et al.</i> Instance Generator.....	85
Table 87. Wang and Valenzela Instance Generator.	86
Table 88. Hopper and Turton Instance Generator.	86
Table 89. SLOPPGEN Instance Generator.	86
Table 90. ep2 (and ep3) Instance Generator.....	86
Table 91. 2DCPackGen Instance Generator.....	86

Part III – Heuristics

Adaptive Sequence-based Heuristic for Two-Dimensional Non-Guillotine Packing Problems

Table 1. Summary of the results obtained.....	122
Table 2. Results for Berkey and Wang instances.	123
Table 3. Results for Martello and Vigo instances.	124
Table 4. Datasets.....	125
Table 5. Computational results - Problems from literature.	125
Table 6. Computational results - Zero-waste problems.	126

Adaptive Sequence-based Heuristic for Two-Dimensional Guillotine Cutting Problems

Table 1. Main features of the datasets.....	141
Table 2. Computational results on the instances of SET1 for exact problems.	142
Table 3. Computational results on the instances of ATP for exact problems.	143
Table 4. Computational results on the instances of SET1 for non-exact problems.	144
Table 5. Computational results on the instances of ATP for non-exact problems.....	145
Table 6. Computational results on the subsets of CLASS for exact problems.....	145
Table 7. Computational results on the subsets of CLASS for non-exact problems.	146
Table 8. Main features of the datasets.....	147
Table 9. Computational results for exact constrained two-staged SLOPP - Set HR.	147
Table 10. Computational results for non-exact constrained two-staged SLOPP - Set HR....	148
Table 11. Computational results for non-exact constrained two-staged SLOPP - Set ATP[30-49].	149

Table 12. Computational results for non-staged SLOPP - Set ATP.	149
Table 13. Computational results for unconstrained non-staged SLOPP - Set GCUT and CMWX.	150

List of Figures

Part I – Thesis

Figure 1. Articles timeline.	8
Figure 2. Solution approaches comparison map.	10
Figure 3. Randomized BubbleSearch.	13
Figure 4. Permutation probability variation.	13

Part II – Research Resources

A Review of Solution Approaches for Two-dimensional Cutting and Packing Problems

Figure 1. Cutting problem.	22
Figure 2. Cutting pattern.	23
Figure 3. One-dimensional problem.	24
Figure 4. Two-dimensional problem.	24
Figure 5. Three-dimensional problem.	24
Figure 6. 1.5-dimensional problem.	25
Figure 7. Regular and irregular items.	25
Figure 8. Non-orthogonal pattern.	25
Figure 9. Guillotine and non-guillotine patterns.	26
Figure 10. Two- and three-staged cuts.	26
Figure 11. Exact and non-exact problems.	27
Figure 12. Non-isotropic material.	27
Figure 13. Group patterns.	28
Figure 14. X-pattern and Y-pattern.	28
Figure 15. Discretization Points.	34
Figure 16. Reduced Raster Points.	34
Figure 17. Difference Process.	35
Figure 18. Corner Points.	35
Figure 19. Extreme Points.	36

Figure 20. Fekete and Schepers' graph-theoretical characterization – Feasible pattern	38
Figure 21. Fekete and Schepers' graph-theoretical characterization – Unfeasible pattern..	38
Figure 22. Non-orthogonal packing.	40
Figure 23. Jakobs' BL Heuristic.....	41
Figure 24. Liu and Teng improved BL algorithm.	41
Figure 25. Burke <i>et al.</i> Array of Occupied Positions.	42
Figure 26. Solutions generated by the placement procedures BL+LB and BL.....	43
Figure 27. Smooth Packing.....	43
Figure 28. Patterns: General (left), Normal (middle), Meet-in-the-Middle (right).	44
Figure 29. Potential improvement.....	48
Figure 30. Sequence Pair representation.	49

Resources for Two-dimensional (and Three-dimensional) Cutting and Packing Solution Methods Research

Figure 1. JSON structure.	96
Figure 2. Instance OF1 – Original format.....	96
Figure 3. Instance OF1 - Converted to JSON.....	97
Figure 4. Menu icon.	97
Figure 5. Instance OF1 – Rendered.....	98
Figure 6. 2D 2-staged SLOPP – 2D JSON.	98
Figure 7. 2D 2-staged SLOPP – 2D render.....	99
Figure 8. Disk icon.	99
Figure 9. 2D 2-staged SLOPP – 3D JSON.	100
Figure 10. 2D 2-staged SLOPP – 3D render.....	100
Figure 11. 3D SBSBPP – Wireframe render.....	101
Figure 12. 3D SBSBPP – Solid unoccupied space render.	102
Figure 13. 3D SBSBPP – Solid render.	102
Figure 14. Homepage.....	103
Figure 15. Literature review.....	105
Figure 16. Type Keywords – Non-guillotine keyword selected.	106
Figure 17. Type Datasets – CGCUT dataset selected.....	106
Figure 18. Type Comparison – Article Gonçalves and Resende (2011) selected.....	107

Figure 19. Quick view.....	107
Figure 20. Quick view - CGCUT dataset selected.....	108
Figure 21. Filter.....	108
Figure 22. Datasets list.....	109
Figure 23. Instance generators list.....	109

Part III – Heuristics

Adaptive Sequence-based Heuristic for Two-Dimensional Non-Guillotine Packing Problems

Figure 1. Difference Process.....	121
-----------------------------------	-----

Adaptive Sequence-based Heuristic for Two-Dimensional Guillotine Cutting Problems

Figure 1. Two- and three-staged cuts.....	132
Figure 2. Exact and non-exact problems.	132

Part I – Thesis

1. Introduction

But still there are many important and practical problems in the industries waiting to be solved efficiently...

Cheng *et al.* [1]

Combinatorial optimization problems can be represented by mathematical models that represent the objectives, resources, constraints, and decision variables of the problems.

Combinatorial optimization is the process of finding the optimal solution among a set of all possible solutions, determining the configuration for decision variables that allows achieving the best result considering the objectives, resources and constraints imposed by the model.

Due to the applicability and importance of combinatorial optimization problems in the most diverse areas, such as economy, industry, transport, medicine, the design of efficient methods that allow obtaining high-quality results in acceptable computational times has been the subject of increasing research activities at enterprise and academic levels.

1.1. Solution Methods

He must figure out how few of the boards he can purchase and still be able to cut all the required lengths from them.

Golden[2]

Solution methods to maximize or minimize the objective function value arise mainly in two flavours, exact and non-exact methods.

Although exact methods are guaranteed to find the optimal solution for any problem, they tend to be extremely demanding in computational resources, even for small and medium sized instances. The search for the optimal solution is performed through, explicit or implicit, enumeration of the entire solution space. Among the most commonly used exact methods, we refer to Branch-and-Bound (see Land and Doig [3], and Agin[4]), Dynamic Programming (see Bellman [5]) and Branch-and-Cut (see Mitchell [6]).

In some exact methods, the obtention of the optimal solution is guaranteed only if there is no time limit for the execution and if the necessary computing resources are provided, which for large-scale instances, can be totally impractical. Hence, the need for solution methods that can obtain solutions of good quality with less demanding computing resources and in an acceptable time.

Approximation algorithms (see Vazirani [7], and Williamson and Shmoys [8]) do not guarantee the optimal solution for a problem but ensure that the solution is within a quality threshold. Using as an example a minimization problem whose optimal solution is denoted as z^* , an algorithm is said to be an α -approximation algorithm, if it guarantees that the solution obtained is always at most $\alpha \times z^*$ (with $\alpha \geq 1$), i.e., for each solution z obtained by this algorithm, we are guaranteed to have $z^* \leq z \leq (\alpha \times z^*)$.

On the other side, heuristics are problem-specific approaches that opposite to approximation algorithms have no guarantee on the quality of the solution obtained but are, usually, much less demanding in computational resources. Commonly, heuristics used to solve optimization problems are divided into three types, namely, constructive, local search and metaheuristic-based heuristics.

Constructive heuristics start with an empty solution and create iteratively a new solution following a set of rules, e.g., add one element at a time to the current solution considering a given sequence of elements. A greedy heuristic refers to a heuristic that repeatedly adds the element that most positively influences the current partial solution considering the objective function. These heuristics are usually used to create an initial solution to be improved by other methods, such as local search heuristics.

Local Search (see Yagiura and Ibaraki [9]) heuristics iteratively explore the neighbourhood¹ of the current solution trying to find a solution that is better than the current one. The search ends when a given iteration fails to improve the current solution with the chosen neighbourhood structure, thus finding a local optimum. The main disadvantage of this approach is the inability to escape local optima.

¹ Assuming S as the space of admissible solutions of a problem P and $s \in S$, it is called neighborhood of s , $N(s)$, to the set of solutions $N(s) \subseteq S$ that is possible to reach by means of a move (specific for the neighborhood structure applied). Each solution of $N(s)$ is called a neighbor solution of s .

Metaheuristic-based heuristics are the application of a metaheuristic to a specific problem. Metaheuristics refer to general methodologies for solving combinatorial optimization problems that are easily adaptable to specific problems and can exploit more efficiently the solution space. Blum e Roli [10] characterize metaheuristics as follows:

- strategies that guide the search process;
- the goal is to efficiently explore the search space to find (near-)optimal solutions;
- can range from simple local search procedures to complex learning processes;
- can incorporate mechanisms to escape local optima;
- there are not specific to a problem;
- more advanced metaheuristics can incorporate memory to guide the search.

Metaheuristics try to find the correct balance between intensification, referring to a deeper exploration of neighbourhoods considered more promising, and diversification, referring to the exploration of less attractive neighbourhoods, to escape local optima.

Some of the most commonly used metaheuristics are the Greedy Randomized Adaptive Search Procedure (GRASP, Feo and Resende [11]), Tabu Search (Glover [12]), Path Relinking (Glover *et al.* [13]), Variable Neighbourhood Search (VNS, Mladenović and Hansen [14]), and Scatter Search (Glover [15][16]).

GRASP is a multi-start (see Martí *et al.* [18]) metaheuristic that applies some local search to solutions that are iteratively generated through a greedy heuristic. This metaheuristic is randomized, in the sense that randomness is added to the greedy heuristic, and adaptive as previous selections of elements can influence current ones.

Tabu Search extends the concept of local search in order to allow the exploration of zones in the solution space that are not considered promising. This metaheuristic makes use of memory structures to guide the search. In its simplest version, it uses short-term memory to store characteristics of the moves that will be considered forbidden, i.e., tabu, in following iterations. Short-term memory may not be enough to allow the search in certain areas of the solution space since it usually stores only the attributes of the most recently visited solutions. Long-term memory stores information that can allow to diversify the search to unexplored regions or to intensify your search for the most promising regions. The tabu state is not permanent being controlled by a parameter, generally referred to as tabu tenure, which may

be, for example, a given number of iterations. Moves considered tabu can only be performed in the following iterations if they fulfil the criteria defined by an aspiration criterion, e.g., the move execution leads to the best solution found so far.

Path Relinking incorporates into a solution attributes from another solution exploiting the trajectories connecting them.

VNS combines the strategy of local search heuristics with the dynamic change of neighbourhoods to escape the local optimums.

Scatter Search is an evolutionary method, in which a population of solutions evolves with the combination of its elements. This metaheuristic constructs new solutions from the combination of solutions belonging to a reference set. This reference set should contain high-quality and diversified solutions to maximize the information that can be derived from the combination of solutions.

Many more metaheuristics have been proposed and used to solve optimization problems. A comprehensive historical perspective on metaheuristic research is presented by Sorensen *et al.* [17].

We refer to Blum and Roli [10] and Sirenko [19] for diverse classifications of metaheuristics, as they can be classified according to various criteria, such as:

- Use of memory structures, e.g., Tabu Search uses of the search history, while Simulated Annealing (Kirkpatrick *et al.* [20]) is considered a memory-less solution method.
- Origin, e.g., Genetic Algorithm (Koza [21]) is considered a nature inspired approach, whereas the Tabu Search is considered a non-nature inspired method.
- Number of solutions considered at any time, e.g., Genetic Algorithms and Scatter Search are population-based approaches, while Tabu Search and VNS are considered trajectory methods.

For more information on metaheuristics, we refer the readers to the surveys of Boussaïd *et al.* [22] and Baghel *et al.* [23], and the books of El-Ghazali [24], Siarry [25] and Salhi [26].

1.2. Background and Motivation

The field of cutting and packing motivates many areas of operations research.

Burke *et al.* [27]

It is extremely common to find situations in which, aiming to maximize the material (or space) used, a set of items to fulfil some requirement must be either cut from a large object or packed into large bins. These problems belong to a class of problems referred to as cutting and packing problems. Cutting and packing problems are one of the most interesting subjects in optimization, mainly due to its complexity and wide applicability. The high impact of these problems in so many industrial areas fosters the need for better and faster solution methods.

Most of the solution methods proposed are specially tailored for specific (theoretical) problems, and to find the right solution method, if one exists, for a (real) problem is not an easy task. Even if it is found, the solution method can rely heavily on previous and/or deep knowledge of optimization concepts, so that only experienced researchers and practitioners can take full advantage of this method.

The goal of this thesis is to create heuristics to solve various cutting and packing problems that are simple to implement (dues straightforwardly adapted to promptly respond to problem domain changes, e.g., market trends changes) and that produce consistently good results.

2. Thesis Contribution

The problem that wouldn't go away.

Garey and Johnson [28]²

This chapter aims to fill the gap between the five papers contained in the second and third parts of this thesis. The papers were included in the order that makes it easier to follow the sequence taken to accomplish this (hard) endeavour.

² Denoting the ever-growing application and research made on this subject.

The first step of this journey was to clearly state the objectives to attain. The objective of this thesis was to present simple and effective solution methods for two two-dimensional problems, belonging to the class of cutting and packing problems, considering both guillotine and non-guillotine cut constraints. The first problem aims to return the most profitable layout of items to be cut from an individual object, e.g., raw material. The second problem aims to cut all items using as few objects as possible to minimize the material losses. If we follow the typology of Wäscher *et al.* [29], depending on the assortment of items, the first problem is classified as a Single Knapsack Problem (SKP) or as a Single Large Object Placement Problem (SLOPP), and the second problem as a Single Bin Size Bin Packing Problem (SBSBPP) or as a Single Stock Size Cutting Stock Problem (SSSCSP).

Obviously given the mass of scientific work that occurs it is impossible for any literature survey to be completely comprehensive...

Beasley [30]

Having in mind the objective to attain, the next step was to define and truly understand the problems under consideration. Cutting and packing problems are the subject of so many research in the last decades that to follow its evolution from the very beginning (here, considered as the seminal work of Gilmore and Gomory [31]) was an exhausting task.

We have gathered and analysed more than 400 documents directly related to the two problems that we were interested in. Figure 1 provides the timeline on the number of documents considered, grouped by decades.

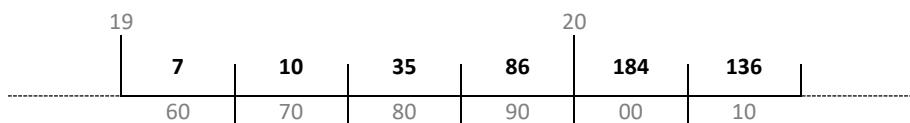


Figure 1. Articles timeline.

While analysing the articles, we were consistently concerned in collecting as much information as possible, such as, addressed problems, solution method proposed, with which methods does the current solution method compares and what data is used for evaluating the effectiveness of the proposed solution method.

The cutting and packing problems family has so many variants that specific jargon is commonly used making it difficult to trace back to their origin and recognize the details of each of them.

The first paper allows a deep understanding of cutting and packing as it gathers most of the specific terminology and concepts helping to classify and recognize each specific problem and its characteristics. This paper, also, reviews the most referred and relevant solution methods on the two-dimensional cutting and packing problem found in the literature. We consider that any newcomer to the cutting and packing world can withdraw great value reading this paper due to the comprehensive information included.

As already mentioned, while reading, anticipating the need for the datasets used by other researchers for our own computational experiments, we gathered the instances used in the articles. Some of these datasets were easily retrieved directly from the articles, while others were obtained from internet websites. We have spent a considerable amount of time tracking the instances used in the analysed articles.

The second paper gives an extensive review of the datasets and instance generators used in the articles reviewed. We consider that this paper is a powerful resource for the researchers in the field of rectangular two-dimensional cutting and packing problem as it provides for each dataset, the set characteristics, the article in which it was defined, and whenever exists an internet link for easy retrieval. This paper contains a description of 84 datasets, considering more than 6300 instances, and 6 instance generators.

Before we started researching for heuristics to solve the problems under consideration, we have created a set of resources to help our work. These resources, described in the third paper, are graphic user interfaces that allow visualising the instances and the generated cutting plans in two- and three-dimensions. The tools provide a powerful resource to researchers in this subject. Also, to ease the data input we converted all instances from the datasets considered in the second paper to a JSON format. This format, besides readability, allows the instance data to be less error-prone due to its structured nature. Finally, a website was created that apart from hosting all the resources developed, also make available a set of utilities that allows analysing the information gathered in our literature research. As an example, as we collected the connection between the solution method and the methods with which they were compared, we have created an iterative map (depicted in Figure 2 with all available connections) that allowed us to easily visualise those connections.

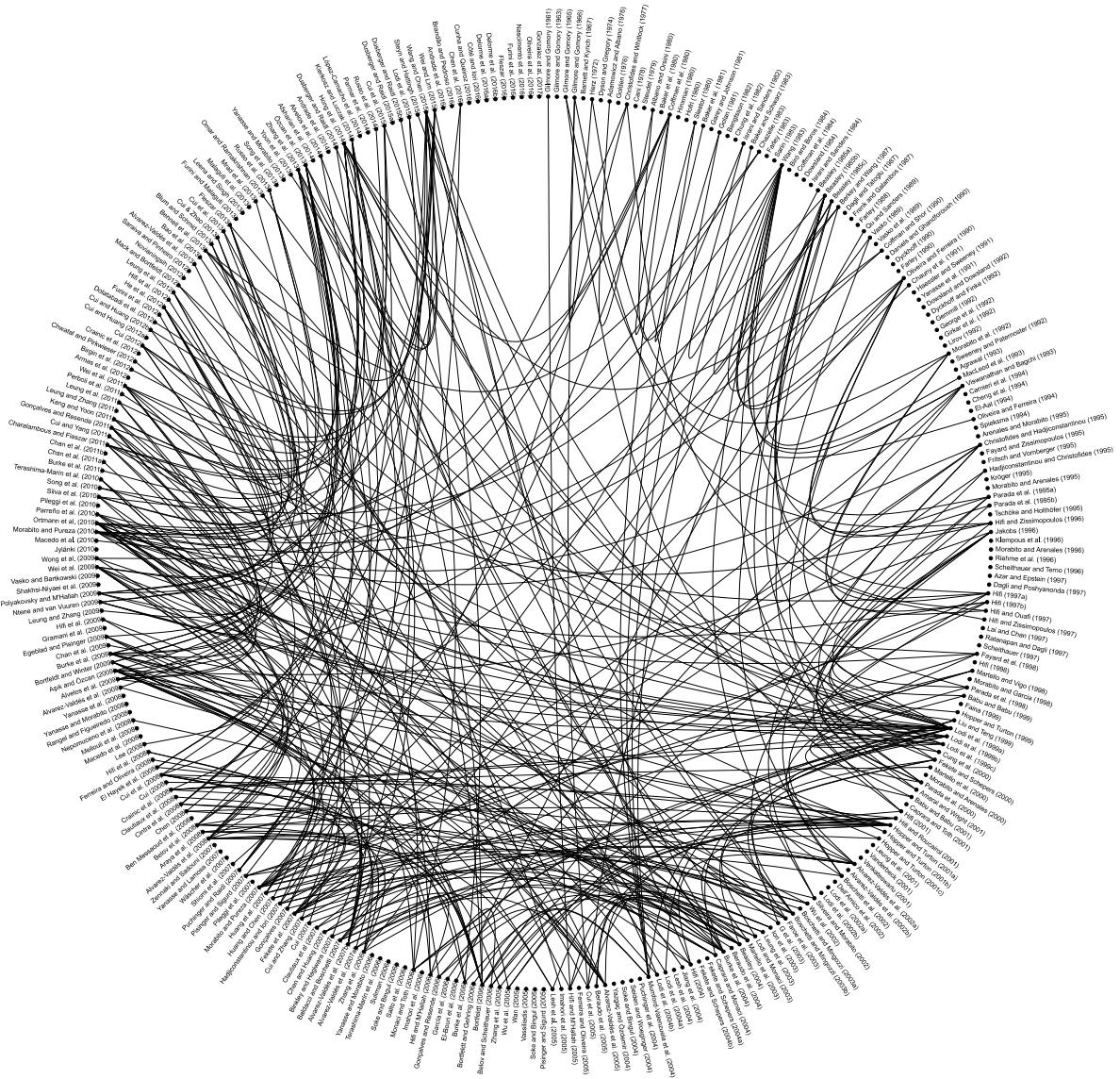


Figure 2. Solution approaches comparison map.

Noteworthy that the literature review could not be useful as it could be if only focused on the problems intended to solve. Many of the solution methods analysed were tailored for the Open Dimensional Problem as some of these can be, and have been, successively adapted for other problems.

Even apparently similar problems may require radically different heuristic techniques.
Hinxman [32]

Making a quick overview of the solution methods analysed in the literature review, it can be noticed that most of the exact methods where based on the Branch-and-Bound and Dynamic

Programming, while most heuristics are either constructive or greedy, or are based on more advanced approaches such as, GRASP, Path Relinking, Tabu Search, Genetic Algorithms, Simulated Annealing, Variable Neighbourhood Search and hybridizations of these approaches. Many other approaches were proposed, but not with the representations of the above-mentioned ones.

While constructive and greedy heuristic are quite simple to implement, usually the results obtained are not as good as the one obtained by more advanced approaches. Although most of these approaches provide good results, a set of issues were denoted. These advanced approaches are heavily based on previous metaheuristics and/or mathematical knowledge. Even comprehensive articles can be overwhelming due to the underlying complexity of the methods, not allowing a straightforward understanding and implementation. Others require so many parametrizations that great part of the method presentation is dedicated to the study of the parameters to use for specific problems (or datasets). Even methods based on metaheuristics have so many components that are problem-domain specific that the true nature of the metaheuristic is lost in the process.

While experienced researchers and practitioners can easily untangle these complex issues and (re)use state-of-the-art solution methods, this added complexity in search of better solutions can alienate others. In a perfect world, a good solution method would be the one that is as simple (to understand and implement) and fast as a constructive heuristic with the high-quality results of a more advanced approach.

Most of the analysed heuristics were two-phase heuristics. The first phase, that we refer to as the sequencing phase, generates a sequence of items that will define the order that the items are to be considered. The second phase, here referred to as the placement phase, places the items into the object(s) considering the sequence of items generated in the previous phase.

We started the development of our heuristics with the placement phase as it was counter-productive to start otherwise. We could use any sequence to evaluate the placement phase, the opposite was not possible.

For non-guillotine problems, to keep track of every possible empty rectangle space (ERS) to place the items, we have chosen the Difference Process (Lai and Chan [33]) which has already

proven its effectiveness in many works as can be observed in our review on solution approaches. Noteworthy, the description and exemplification of this method made by the authors provided a straightforward implementation. Considering a given item ordering, the placement method for non-guillotine problems, iteratively, places the current item in one existent ERS.

For problems with guillotine cut constraints, simple bottom-left procedures were implemented and evaluated. The results with these procedures were not near our expectations, and an alternative approach had to be devised. We observed during the literature review, that many placement methods considering guillotine cut constraints used the Knapsack Problem (KP, see Martello and Toth [34]) to maximize the space used. We knew, in advance, that the KP could be solved efficiently and most important rapidly by the algorithms proposed by Pisinger [35], and that these algorithms were freely made available³ for academic purpose. The developed placement method considering a given item ordering iteratively creates a strip then filled solving the associated KP.

We started to study the sequencing phase, adopting simple sequences of items ordered by some criteria, such as height, length, area, perimeter, value, randomized. Although extremely fast, the results were by no means close to our goals in terms of the quality of solutions. This is explained by the narrow solution space explored, so a sequencing that allowed a wider exploration of the solution space was needed.

While reading the articles in the literature review phase, we gain a particular interest in the work of Lesh *et al.* [36] that creates sequences based on the Kendall-tau distance between two permutations. This approach, denoted as BubbleSearch, was later generalized by Lesh and Mitzenmacher in [37] and evaluated on two distinct problems, namely ODP and Jobshop scheduling, comparing favourably against similar GRASP approaches. As GRASP is one of the simplest metaheuristics and very effective in so many optimization problems (see Festa *et al.* [38], [39], and Resende and Ribeiro[40]) we were impelled to work in this direction.

³ <http://hjemmesider.diku.dk/~pisinger/codes.html>

Some variants to the BubbleSearch were proposed by Lesh and Mitzenmacher in [37] to include randomization to the permutation probability (α) or to consider the replacement of the current ordering whenever the new ordering provides a better solution than the current.

The Randomized BubbleSearch can be implemented as a simple stochastic process, depicted in Figure 3 and next briefly described. While there are items of the base sequence (S_{base}) that are not in the new sequence ($S_{current}$), a random number is generated iteratively for each item of S_{base} that is not already in $S_{current}$. If the generated number is greater than the permutation probability (α), the item is copied to $S_{current}$ at the next empty position otherwise, the item will be copied in a following iteration.

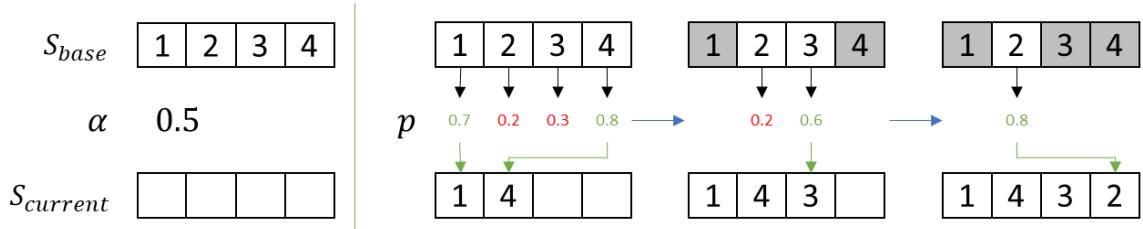


Figure 3. Randomized BubbleSearch.

These variants were implemented with a considerable improvement on the results obtained over the simple orderings. But, although improved, the results were not aligned with the ones obtained by more advanced approaches.

We have intensified the search near the solution space of the solution that obtained good results, i.e., when a new best solution is found, the current ordering will substitute the base ordering for the next iterations and α is reset to its minimum value to generate sequences with minor differences. To diversify the search space, whenever an ordering does not improve the best solution, α is incremented to generate sequences with incrementally more differences. As depicted in Figure 4, as α is incremented, greater is the difference between S_{base} and the new sequence generated.

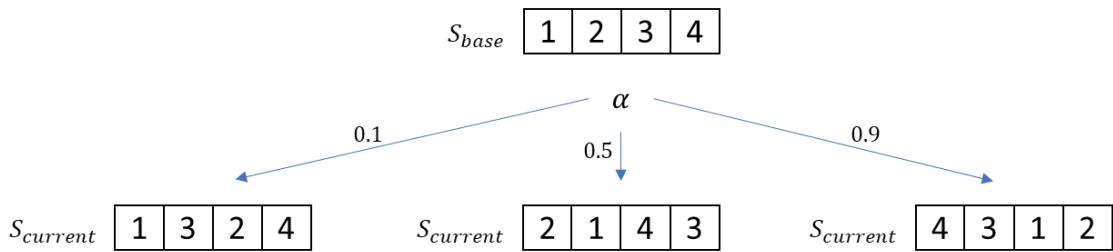


Figure 4. Permutation probability variation.

Now, with each of the major components exposed, i.e., sequencing and placement phases, the proposed heuristics can be more easily described.

The heuristics, denoted hereafter as Adaptive Sequence-based Heuristics (ASH), start with a base sequence of items and with α set to its minimum value.

A new solution is created at each iteration until the maximum number of iterations or the optimality criterium is reached. For multiple identical objects problems (SBSBPP and SSSCSP), as the aim is to place all the items in the minimum number of objects as possible, the optimality criterium is reached when a solution has an objective function value equal to the Continuous Lower Bound ($CLB = \lceil \sum_{i=1}^m l_i h_i / LH \rceil$). For problems with only one object (SKP and SLOPP), as the objective is to obtain the maximum profit pattern, no improvement exists when a pattern includes all items.

At each iteration, a new ordering is generated considering the current base sequence and the permutation probability to create a new solution. For multiple identical objects problems, a solution is created using the placement methods above described until all items are placed. For the SKP and SLOPP, a solution is generated with only one pattern.

If the solution generated is the best one found so far, the base sequence will be replaced by the current ordering and α is reset to its minimum value. Otherwise, the base sequence remains the same and α is incremented to generate orderings incrementally with more differences.

The first paper of the third part of this thesis presents the ASH for solving the non-guillotine SBSBPP and SKP, while the second paper presents the ASH for solving the two- and three-staged (non-)exact SSSCSP and two- and non-staged (non-)exact (un)constrained SLOPP.

The proposed heuristics attained good results comparing favourably against most of the state-of-the-art algorithms. Also, the computational times required by ASH, although always a controversial subject, seems to be extremely low and predictable when compared with more complex approaches.

The results obtained with ASH for the non-guillotine SBSBPP were published in [41] and the results obtained for the non-guillotine SKP were presented at the 20th Congress of the

Portuguese Operational Research Association and will be published in the "Springer Proceedings in Mathematics and Statistics" series.

3. Conclusions

The cutting stock problem is a large scale combinatorial problem for which several solution techniques exist in the literature each with its drawbacks and/or approximations.

Golden [2]

The complexity of cutting and packing problems encourages the investigation of methods, mainly heuristics, that allow high-quality solutions to be obtained in acceptable computational times, since solving these problems through exact methods, especially for large problems, may be impractical due to the computational resources required.

Besides the two surveys on solution methods and datasets, we have made available a set of research resources that can be an important contribution helping future research on the cutting and packing field. We have compiled and converted a great number of datasets into a common format. Also, graphical user interfaces for instance and solution visualisation, and a set of utilities for the literature analysis were created. All these resources can be now easily accessed and used through the website that we have created and made available online.

We propose heuristics to solve two related cutting and packing problems considering guillotine and non-guillotine cuts. The first problem aims the most valuable layout of items assigned to one object, while the second one aims to fully assign a given set of items to the minimum number of available objects possible.

The main concept of our heuristics is to generate orderings that are very similar to those that provided good solutions and to incrementally introduce more changes to the ordering when better solutions are not found to diversify the search.

The application of ASH to the considered problems produced very good results with very low computational times being capable to rival with more complex and tailored solution methods.

The proposed solution methods can be seen as effective heuristics to be applied when a fast solution method must be provided, a high number of variables exist or, to be used combined with more advanced solution methods to create an initial population of diverse solution or used on a bounding scheme.

The work of this thesis can be further developed, studying the influence of the parameters used by ASH to provide an implementation guide. These heuristics should be studied in a real-world situation to measure their effectiveness using as reference the current state (Filipič and Tušar [42]). Also, it would be interesting to study the application of ASH applied to other optimization problems.

References

- [1] C. H. Cheng, B. R. Feiring, and T. C. E. Cheng, “The cutting stock problem — A survey,” *Int. J. Prod. Econ.*, vol. 36, no. 3, pp. 291–305, 1994.
- [2] B. L. Golden, “Approaches to the Cutting Stock Problem,” *AIIE Trans.*, vol. 8, no. 2, pp. 265–274, 1976.
- [3] A. H. Land and A. G. Doig, “An Automatic Method of Solving Discrete Programming Problems,” *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.
- [4] N. Agin, “Optimum Seeking with Branch and Bound,” *Manage. Sci.*, vol. 13, no. 4, pp. B176–B185, 1966.
- [5] R. Bellman, “The Theory of Dynamic Programming,” *Bull. Am. Math. Soc.*, vol. 60, no. 6, pp. 503–515, 1954.
- [6] J. E. Mitchell, “Branch-and-Cut Algorithms for Combinatorial Optimization Problems,” *Handb. Appl. Optim.*, pp. 65–77, 2002.
- [7] V. V. Vazirani, *Approximation Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.
- [8] D. P. Williamson and D. B. Shmoys, *The design of approximation algorithms*, 1st ed. New York, NY, USA: Cambridge University Press, 2011.
- [9] M. Yagiura and T. Ibaraki, “Local Search,” in *Handbook of Applied Optimization*, P. M. Pardalos and M. G. C. Resende, Eds. Oxford University Press, 2002, pp. 104–123.
- [10] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.
- [11] T. Feo and M. G. C. Resende, “Greedy randomized adaptive search procedures,” *J. Glob. Optim.*, pp. 109–133, 1995.

- [12] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.
- [13] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control Cybern.*, vol. 39, no. 3, pp. 653–684, 2000.
- [14] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
- [15] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, 1977.
- [16] F. Glover, "A template for scatter search and path relinking," *Artif. Evol.*, vol. 1363, no. February 1998, pp. 3–51, 1998.
- [17] K. Sørensen, M. Sevaux, and F. Glover, "A History of Metaheuristics," *Handb. Heuristics*, no. January, pp. 1–16, 2016.
- [18] R. Martí, M. G. C. Resende, and C. C. Ribeiro, "Multi-start methods for combinatorial optimization," *Eur. J. Oper. Res.*, vol. 226, no. 1, pp. 1–8, 2013.
- [19] S. Sirenko, "Classification of heuristic methods in combinatorial optimization," *Inf. Theor. Appl.*, vol. 16, no. 4, pp. 303–322, 2009.
- [20] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [21] J. R. Koza, *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press, 1992.
- [22] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci. (Ny.)*, vol. 237, no. February, pp. 82–117, 2013.
- [23] M. Baghel, S. Agrawal, and S. Silakari, "Survey of Metaheuristic Algorithms for Combinatorial Optimization," *Int. J. Comput. Appl.*, vol. 58, no. 19, pp. 21–31, 2012.
- [24] T. El-Ghazali, "Metaheuristics: from design to implementation," *Jonh Wiley Sons Inc., Chichester*, 2009.
- [25] P. Siarry, *Metaheuristics*. Cham: Springer International Publishing, 2016.
- [26] S. Salhi, *Heuristic Search*. Cham: Springer International Publishing, 2017.
- [27] E. K. Burke, G. Kendall, and G. Whitwell, "A New Placement Heuristic for the Orthogonal Stock-Cutting Problem," *Oper. Res.*, vol. 52, no. 4, pp. 655–671, 2004.
- [28] M. R. Garey and D. S. Johnson, "Approximation Algorithms for Bin Packing Problems: A Survey," in *Analysis and Design of Algorithms in Combinatorial Optimization*, G. Ausiello and M. Lucertini, Eds. Vienna: Springer Vienna, 1981, pp. 147–172.
- [29] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [30] J. E. Beasley, "A population heuristic for constrained two-dimensional non-guillotine cutting," *Eur. J. Oper. Res.*, vol. 156, no. 3, pp. 601–627, 2004.

- [31] P. C. Gilmore and R. E. Gomory, “A linear programming approach to the cutting-stock problem,” *Oper. Res.*, vol. 9, no. 6, pp. 849–859, 1961.
- [32] A. I. Hinxman, “Trim-Loss and Assortment Problems - a Survey.,” *Eur. J. Oper. Res.*, vol. 5, no. 1, pp. 8–18, 1980.
- [33] K. K. Lai and J. W. M. Chan, “Developing a simulated annealing algorithm for the cutting stock problem,” *Comput. Ind. Eng.*, vol. 32, no. 1, pp. 115–127, 1997.
- [34] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. New York: Wiley, 1990.
- [35] D. Pisinger, “Algorithms for knapsack problems,” Copengagen, 1995.
- [36] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher, “New heuristic and interactive approaches to 2D rectangular strip packing,” *J. Exp. Algorithmics*, vol. 10, no. 1, p. 1.2, 2005.
- [37] N. Lesh and M. Mitzenmacher, “BubbleSearch: A simple heuristic for improving priority-based greedy algorithms,” *Inf. Process. Lett.*, vol. 97, no. 4, pp. 161–169, 2006.
- [38] P. Festa and M. G. C. Resende, “An annotated bibliography of GRASP,” *Oper. Res. Lett.*, vol. 8, no. 1, pp. 67–71, 2004.
- [39] P. Festa and M. G. C. Resende, “An annotated bibliography of GRASP – Part II: Applications,” *Int. Trans. Oper. Res.*, vol. 16, no. 1, pp. 131–172, 2009.
- [40] M. G. C. Resende and C. C. Ribeiro, *Optimization by GRASP*. New York, NY: Springer New York, 2016.
- [41] Ó. Oliveira and D. Gamboa, “Adaptive Sequence-Based Heuristic for the Two-Dimensional Non-guillotine Bin Packing Problem,” in *Hybrid Intelligent Systems*, A. M. Madureira, A. Abraham, N. Gandhi, and M. L. Varela, Eds. Cham: Springer International Publishing, 2020, pp. 370–375.
- [42] B. Filipič and T. Tušar, “Challenges of Applying Optimization Methodology in Industry,” in *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, 2013, pp. 1103–1104.

Part II – Research Resources

A Review of Solution Approaches for Two-dimensional Cutting and Packing Problems

Abstract Cutting and packing problems have been widely studied in the last decades mainly due to the variety of industrial applications where the problems emerge. This paper presents an overview of the solution approaches that have been proposed for solving two-dimensional rectangular cutting and packing problems. The main emphasis of this work is on two distinct problems belonging to the cutting and packing problem family. The first problem aims to place onto an object the maximum-profit subset of items, while the second one aims to place all the items using as few identical objects as possible. The objective of this review is not to be exhaustive but to provide a solid grasp on cutting and packing problems describing the most important and referred solution approaches proposed to solve the problems considered.

Keywords: Two-dimensional, Rectangular, Non-guillotine, Guillotine, Cutting Problems, Packing Problems

1. Introduction

Cutting and packing problems have been the focus of growing research due to its computational complexity, which is NP-hard (see Garey and Johnson [1]), and due to its wide applicability. Another important reason that motivates the research on the cutting and packing field is that these problems represent a critical logistic and production planning activity in many industries. Examples can be found in the automotive (e.g., El-Aal [2]), glass (e.g., Farley [3]), steel (e.g., Vasko *et al.* [4]), wood-based (e.g., Morábito and Garcia [5]), paper (e.g., Lai and Chan [6]) and TFT-LCD (e.g., Tsai *et al.* [7]) industries.

We refer to Sweeney and Paternoster [8] for a categorized application-orientated bibliography that counts with more than 400 published works related to cutting problems, to Singh and Jain [9] for a survey on the industrial scope of two-dimensional cutting and packing problems and to Macedo *et al.* [10] for a detailed survey on software packages.

The cutting and packing problems considered in this paper aim to assign, to either orthogonally cut or pack, a set of m rectangular items to one or more larger identical objects characterized by their length (L) and height (H). Each item type i , with $i = 1 \dots m$, have associated length (l_i), height (h_i), value (v_i), and demand (d_i). The emphasis of this work is on two specific problems belonging to the cutting and packing problem family. The first considers only one object and the objective to attain is to maximize the value of the items assigned to it, while the second one considers multiple identical objects and the objective is to minimize the total number of objects used to assign all items.

The objective of this paper is to give an overview of the work related to the two-dimensional (2D) rectangular cutting and packing problems, providing to the reader a solid grasp on this subject.

The rest of the paper is organized as follows. Section 2 provides a deep coverage on the cutting and packing topic describing the main problem characteristics and the typologies presented in the literature to classify the problems through their common characteristics. Section 3 is dedicated to the solution approaches available in the literature for solving cutting and packing problems. Finally, some conclusions are tissue in the last section.

In Appendix A, articles referred in this paper are classified by the cutting and packing problem for which a solution method was presented.

2. Cutting and Packing Problems

The cutting problem, as depicted in Figure 1, considers the existence of a set of available objects (a) and a set of items (b) that must be extracted from the objects to fulfil some demand (c).

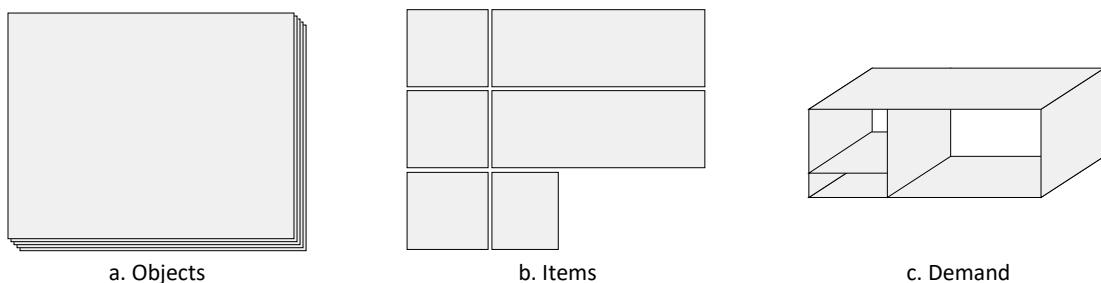


Figure 1. Cutting problem.

The cutting problem can be found in the literature with different names, deriving essentially from the industry or economic environment in which the problem arises. For example, it can appear as a packing problem where items, instead of being extracted from objects, must be arranged in a larger space aiming at the minimization of the unfilled space.

The expected result when solving this problem is called a cutting plan. A cutting plan is a set of cutting patterns (Figure 2), each of them with an associated cut frequency, and wherein the items are allocated to the objects. The residual parts, i.e., figures that occur in patterns that do not belong to the set of items, are considered losses (depicted with the darker area in Figure 2). These losses are commonly referred to as trim loss.

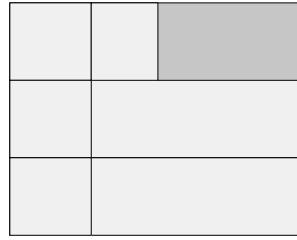


Figure 2. Cutting pattern.

The first formulation for cutting and packing problems was presented by the economist Kantorovich [11] as follows:

$$\text{minimize } z = \sum_{k \in K} y_k \quad (1)$$

$$\text{Subject to } \sum_{k \in K} x_{ik} \geq d_i \quad \forall i \in I \quad (2)$$

$$\sum_{i \in I} l_i x_{ik} \leq L y_k \quad \forall k \text{ in } K \quad (3)$$

$$x_{ik} \in \mathbb{N} \quad \forall i \text{ in } I, \forall k \in K \quad (4)$$

$$y_k \in \{0,1\} \quad \forall k \text{ in } K \quad (5)$$

Where K is the set of available objects, L is the length of the object, I is the set of items and d_i and l_i are, respectively, the demand and the size of item i . Considering the value of y_k as 1 if object k is used in the solution (0 otherwise) and x_{ik} as the number of times that item i is cut in object k , expression n (1) defines the objective function as the minimization of the number of objects used to cut all the items, (2) defines the demand constraints, (3) defines the pattern feasibility constraints and (4) and (5) represent the domain of the variables.

In most problems, objects and items represent geometric figures of fixed sizes, and although in most of the problems related to cutting and packing the space to be considered is three-dimensional, in some cases, only one or two dimensions are relevant to solve the problem. In terms of dimensionality, these problems can be considered as:

- One-dimensional – Only one dimension is relevant, e.g., cutting a roll of paper into smaller pieces, in which the only relevant dimension is the length (Figure 3).

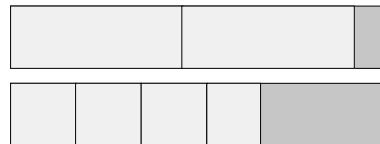


Figure 3. One-dimensional problem.

- Two-dimensional – Only two dimensions are relevant to solve the problem, e.g., cut items from a wooden plate that have the same thickness as the plate (Figure 4).

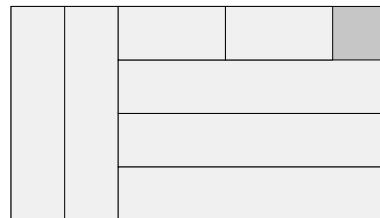


Figure 4. Two-dimensional problem.

- Three-dimensional – The three dimensions are relevant, e.g., arrangement of volumes in a larger space (Figure 5).

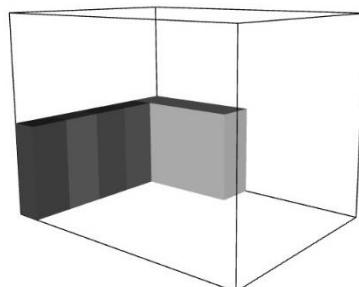


Figure 5. Three-dimensional problem.

- Multi-dimensional – More than three dimensions are considered, e.g., in addition to the physical dimensions, a temporal dimension is included.
- Open dimensional – One of the dimensions is unbounded in size. A 1.5-dimensional problem specifies a two-dimensional problem where one dimension is fixed and the other is variable, while in the 2.5-dimensional, two dimensions are fixed. An example

of the former is the cutting process of a roll of textile that has a fixed height but can be unrolled to accommodate more items when needed (Figure 6).

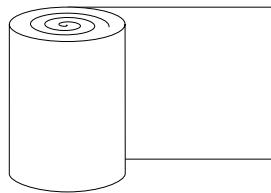
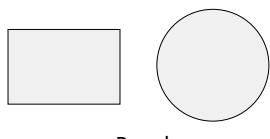


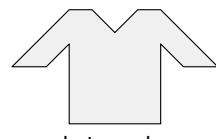
Figure 6. 1.5-dimensional problem.

We can find a wide variety of cutting and packing problems mainly due to specific requirements of industries, cutting machinery and, even, the raw material used.

The items to be considered when solving a problem can have regular or irregular shapes. Figure 7 depicts on the left two regular shapes, namely rectangular and circular, and to the right an irregular shape.



a. Regular



b. Irregular

Figure 7. Regular and irregular items.

Some problems may require an orthogonal placement of the items, i.e., parallel to the sides of the object, while other problems allow a non-orthogonal placement relative to the object edges as depicted in Figure 8.

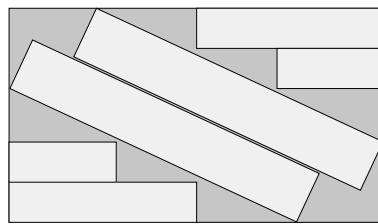
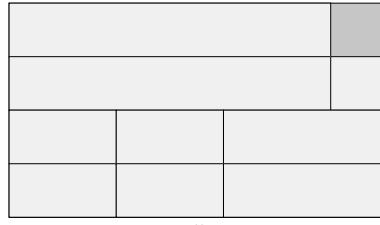
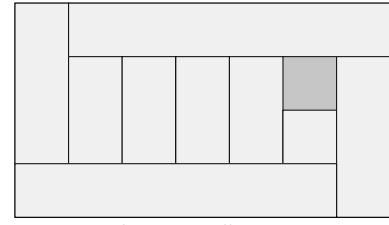


Figure 8. Non-orthogonal pattern.

The cutting machines may only perform straight cuts on objects from one side to the other, called guillotine cuts. Figure 9 depicts to the left a guillotine pattern, and to the right a non-guillotine pattern.



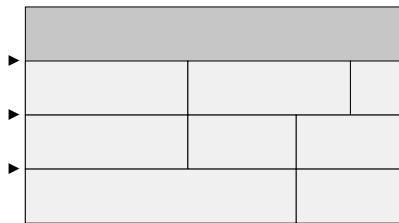
a. Guillotine



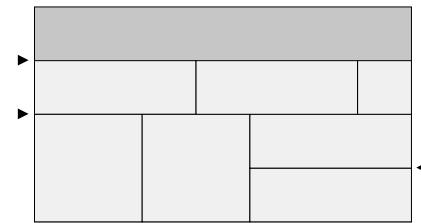
b. Non-guillotine

Figure 9. Guillotine and non-guillotine patterns.

Some machines may perform the object rotation (or blade rotation) and carry out the cuts in stages, i.e., number of rotations. If an upper bound (k) to the number of stages exists, the problem is considered as k -staged; otherwise called n -staged (or non-staged, e.g., Morabito *et al.* [12]). Figure 10 depicts two-staged (a) and three-staged (b) patterns. In this figure, the first stage is horizontal and is identified with the arrows pointing to the right. As staged patterns are considered, the second stage will perform vertical cuts on the stripes generated previously. The three-staged pattern (b) has a horizontal third stage, identified with the arrow pointing to the left.



a. Two-staged

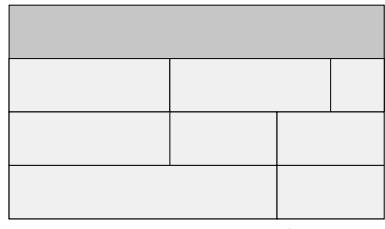


b. Three-staged

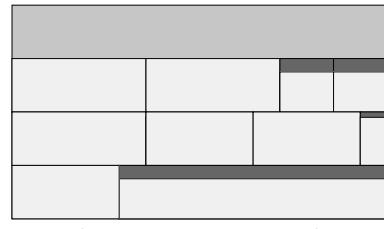
Figure 10. Two- and three-staged cuts.

A problem is classified as restricted (e.g., Silva *et al.* [13]) if it is required that all resulting strips have one of their dimensions defined by one of their contained items, e.g., a strip resulting from a horizontal cut have the height defined by the highest item included in the strip.

Staged problems can allow an extra cut to trim down the item to its exact dimensions. This extra cut is often called a trimming cut and the problem is classified as non-exact if trimming is allowed and as exact otherwise. Figure 11 depicts in the left an exact two-staged pattern and in the right a non-exact two-staged pattern (the darker grey identifies the extra trimming cut area).



a. Exact two-staged



b. Non-exact two-staged

Figure 11. Exact and non-exact problems.

The problems are considered constrained when an upper or lower bound on the quantity of the items to be cut is defined. Problems are considered double-constrained if both lower and upper bounds are present. When no bounds are considered to restrict the number of occurrences of each item, the problem is named as unconstrained.

A problem is considered unweighted if the value of all items is equal to its area or weighted if each item as another value associated, e.g., cost, priority.

If the objects are uniform in all orientations, i.e., isotropic material, the rotation of items can be allowed, classifying the problem as non-oriented or as oriented otherwise. Figure 12 illustrates a non-isotropic material in which the orientation of the cuts matters.

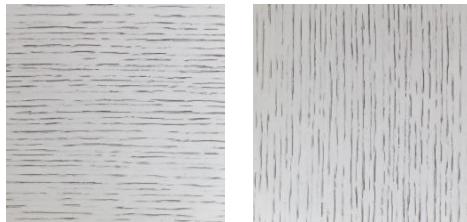
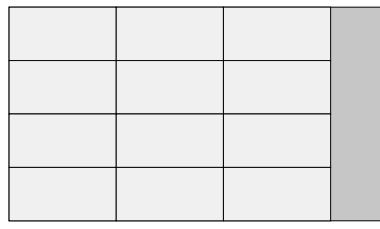


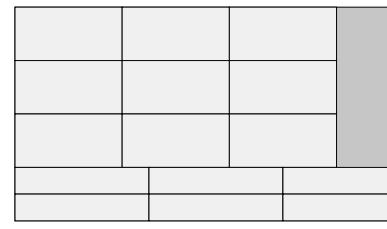
Figure 12. Non-isotropic material.

The objects to be considered in stock may vary in the quantity available and in the sizes. The stock can be considered unlimited when there is a large availability or when the objects can easily be obtained. All the objects may have the same size or have many different sizes. In the latter case, the sizes can be standardized or can arise from the use of retails (also found in the literature as usable leftovers, e.g., Andrade *et al.* [14]) that result from previous cutting processes.

A cutting pattern is classified as 1-group pattern if all second stage cuts must be made simultaneously on all the strips generated on the first stage. A p -group pattern is a guillotine pattern composed with p 1-group pattern forming blocks. Figure 13 illustrates to the left a 1-group pattern and to the right a 2-group pattern.



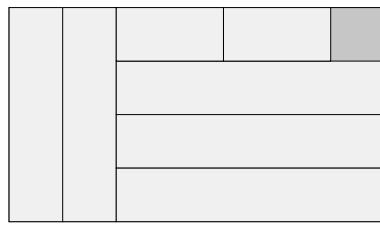
a. 1-group pattern



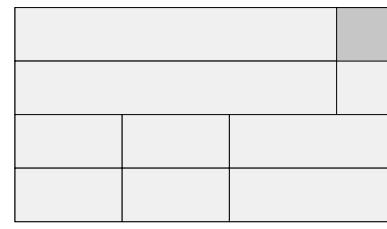
b. 2-group pattern

Figure 13. Group patterns.

Depending on the requirement for the first cut direction, a cutting pattern can be classified as an X-pattern and Y-pattern. An X-pattern corresponds to a pattern created with a vertical first cut direction, i.e., the first stage segments are placed side by side. Y patterns are created with horizontal first cut direction, i.e., the first stage segments are stacked on top of each other. Figure 14 illustrates to the left an X-pattern and to the right a Y-pattern.



a. X-pattern



b. Y-pattern

Figure 14. X-pattern and Y-pattern.

2.1. Typologies

This wide variety of possible cutting and packing problems led authors to create typologies to aggregate the problems with common characteristics.

In the typology of Dyckhoff [15], each type of problem is identified by four criteria (see Table 1) with the following structure:

Dimensionality / Kind of assignment / Assortment of large objects / Assortment of small items

The one-dimensional problem, where a set of items with a certain demand must be cut from available objects with the same size, is classified as 1/V/I/M. If there are many items with few different sizes it is classified as 1/V/I/R.

To overcome some limitations of the Dyckhoff's typology, such as the impossibility of differentiating similar problems that have different characteristics, Wäscher *et al.* [16] proposed an improved typology. The authors consider five criteria (see Table 2) and with the combination of these, three problem types are defined: Basic, Intermediate and Refined.

Table 1. Typology of Dyckhoff - Criteria.

Dimensionality	
(n)	Number of dimensions
Kind of Assignment	
(B)	All objects and a selection of items
(V)	A selection of objects and all items
Assortment of Large Objects	
(O)	One object
(I)	Identical figure
(V)	Different figures
Assortment of Small Items	
(F)	Few items (of different figures)
(M)	Many items of many different figures
(R)	Many items of relatively few different (non-congruent) figures
(C)	Congruent figures

Table 2. Typology of Wäscher *et al.* (2007) - Criteria.

Dimensionality	
One-dimensional	
Two-dimensional	
Three-dimensional	
Kind of Assignment	
Output (value) maximization (like B in Dyckhoff's typology in which all objects are used for assigning a selection of items)	
Input (value) minimization (like V in Dyckhoff's typology in which a selection of objects is used for assigning all items)	
Assortment of Small Items	
Identical	
Weakly heterogeneous	
Strongly heterogeneous	
Assortment of Large Objects	
One Large Object	
All dimensions fixed	
One or more variable dimensions	
Several Large Objects (only fixed dimensions are considered)	
Identical	
Weakly heterogeneous	
Strongly heterogeneous	
Shape of Small Items	
Regular	
Irregular	

The Basic types result from the combination of the criteria Kind of assignment and Assortment of small items. The six Basic types are combined with the Assortment of large objects criterion to obtain fourteen Intermediate problem types (see Table 3).

Table 3. Typology of Wäscher *et al.* (2007).

		Basic Problem Type	Assort. of large objects	Intermediate Problem Type	
Assort. of small items					
Kind of assignment	Identical	IIPP Identical Item Packing Problem	One object	IIPP	Identical Item Packing Problem
	Weakly heterogeneous	PP Placement Problem	One object	SLOPP	Single Large Object PP
			Identical	MILOPP	Multiple Identical Large Object PP
			Heterogeneous	MHLOPP	Multiple Heterogeneous Large Object PP
	Strongly heterogeneous	KP Knapsack Problem	One object	SKP	Single KP
			Identical	MIKP	Multiple Identical KP
			Heterogeneous	MHKP	Multiple Heterogeneous KP
	Arbitrary (Variable Dimensions)	ODP Open Dimension Problem	One object	ODP	Open Dimension Problem
	Weakly heterogeneous	CSP Cutting Stock Problem	Identical	SSSCSP	Single Stock Size CSP
			Weakly heterogeneous	MSSCSP	Multiple Stock Size CSP
			Strongly heterogeneous	RCSP	Residual CSP
	Strongly heterogeneous	BPP Bin Packing Problem	Identical	SBSBPP	Single Bin Size BPP
			Weakly heterogeneous	MBSBPP	Multiple Bin Size BPP
			Strongly heterogeneous	RBPP	Residual BPP

The Dimensionality and the Shape of small items (for two- and three-dimensional problems) combined with Intermediate problem types make up the Refined problems, according to the following structure:

$$\{ 1, 2, 3 \} D \{ \text{Rectangular, Circular, ..., Irregular} \} \{ \text{Intermediate Type of Problem} \}$$

The example mentioned above would be classified as 1D Single Bin Size Bin Packing Problem (or 1D Single Stock Size Cutting Stock Problem, if the items are weakly heterogeneous). A problem with rectangular items and in which the object has a variable dimension would be classified as a 2D rectangular Open Dimension Problem.

Wäscher *et al.* [16] focused their work classifying what they call *pure* cutting and packing problems, in which solutions consist in the information about the set of cutting patterns and the value of the corresponding objective function.

Problems with the same properties as the Refined problems but, with additional characteristics unrelated to the cutting or packing, are considered Extensions, e.g., minimize the number of different cutting patterns.

When the conditions of a problem are different from those presented in *pure* problems, it is considered a Variant, e.g., a problem with more than three dimensions.

The EURO Special Interest Group on Cutting and Packing (ESICUP) website¹ provides an extensive database of publications organized and classified by the Wäscher's typology.

3. Solution Approaches

The Column Generation procedure proposed by Gilmore and Gomory [17] for solving the CSP is commonly recognized as the seminal work on the cutting and packing problem research field. This method allows to find the optimal solution for the linear programming problem that is obtained relaxing the integrality constraints of the original problem. A last step is usually required to obtain a feasible solution for the integer problem.

Besides the Column Generation, many more solution approaches have been proposed for solving cutting and packing problems. These methods can be divided into exact methods, i.e., methods in which the obtention of the optimal solution is guaranteed, and in non-exact methods. Non-exact methods, although without the guarantee of finding the optimal solution, usually obtain good results with considerably less computational resources than the ones required by exact methods.

In this section, we review solution methods that have been proposed in the literature for solving cutting and packing problems, classifying them as exact, heuristic, i.e., non-exact problem-specific approaches, and metaheuristic, i.e., methods that are based in general methodologies for solving combinatorial optimization problems that are easily adaptable to specific problems.

The rest of the section is organized as follows. Due to its relevance for the cutting and packing research the Column Generation method is reviewed in subsection 3.1. Subsection 3.2. is

¹ <https://www.euro-online.org/websites/esicup/>

devoted to the location approaches, which are used in both exact and non-exact methods, to enumerate the possible locations to place the items or to keep track of the set of empty spaces that remain in the object after the placement of the items. Section 3.3. and section 3.4. present the exact and non-exact methods, respectively. While the focus is on the most recent solution methods, some of the earlier works are also discussed due to their importance in this research field. Finally, the last section, presents, in chronological order, the published surveys and reviews on solution methods for solving cutting and packing problems.

3.1. Column Generation

The Column Generation² (CG, see Ford and Fulkerson [18]) to solve the one-dimensional CSP was proposed by Gilmore and Gomory [17] [19]. Bearing in mind that even for small problems it can be impractical and time-consuming to enumerate all possible patterns, Gilmore and Gomory proposed the CG based on the Simplex method³ (see Dantzig [20]) modelling the problem as follows:

$$\min \left\{ \sum_{j=1}^{|J|} x_j \mid \sum_{j=1}^{|J|} a_{ij} x_j \geq d_i, i = 1, \dots, m; x_j \geq 0 \text{ and } x_j \in \mathbb{N}, j = 1, \dots, |J| \right\} \quad (6)$$

The set of valid patterns is represented by J , the number of times items of type i are present in pattern j by a_{ij} and the cut frequency of pattern j by x_j . The proposed approach solves the following associated linear programming model:

$$\min \left\{ \sum_{j=1}^{|J|} x_j \mid \sum_{j=1}^{|J|} a_{ij} x_j \geq d_i, i = 1, \dots, m; x_j \geq 0 \text{ and } x_j \in \mathbb{R}, j = 1, \dots, |J| \right\} \quad (7)$$

After the linear programming relaxation on the integrality constraint of the decision variables, an initial set of m patterns (columns) is selected. At each iteration of this method, a sub-problem ($\min\{1 - \sum_{i=1}^m \pi_i x_i\}$ with π_i as the value of the item type i obtained from the current dual linear programming solution) to find the most negative column is solved generating a new pattern. For the one-dimensional problem, the sub-problem (ignoring the constant 1) corresponds to a bounded knapsack problem formulated as:

$$\max \left\{ \sum_{i=1}^m \pi_i x_i \mid \sum_{i=1}^m l_i x_i \leq L; 0 \leq x_i \leq d_i, i = 1, \dots, m \right\} \quad (8)$$

² See Lübecke [182] for a recent overview of this method.

³ See Prabhu [183] for a recent overview of this method.

where x_i represents the frequency of item type i , i.e., number of times that the item appears in the pattern. This new pattern is added if its value is greater than 1. This process terminates when no column can be added, thus obtaining the optimal solution of the relaxed problem. Since this solution may be infeasible for the original problem, an additional method may have to be applied to obtain an integer solution.

An extension to their previous work was presented by Gilmore and Gomory [21] to deal with multistage multidimensional CSP and exemplified with the three-dimensional three-staged case the difficulties of dealing with higher dimensions. The authors also tackle the problem in which scheduling issues must be considered with an application of their method to the production of corrugated paper boxes.

Considering that the computational effort required by the CG approach is closely related to the resolution of the auxiliary problem, Oliveira and Ferreira [22] proposed the Faster Delayed Column Generation (FCG) that reduces the number of times that the auxiliary problem is solved providing an effective reduction of the computational effort required. The auxiliary problem is solved to optimality only if a heuristic cannot produce a column that improves the objective function. The authors present a greedy heuristic to create three-staged and non-staged patterns that place the items into the object by non-increasing reduced cost.

Solution methods for the SLOPP, SSSCSP, ODP, and MSSCSP were presented by Cintra *et al.* [23]. In this work, Dynamic Programming (Bellman [24]) algorithms for the k - and non-staged SLOPP are proposed. The algorithms for the SLOPP are used to solve the sub-problem in a CG approach for the mentioned problems.

A CG based algorithm for the SLOPP with guillotine cuts using mixed integer programming to obtain an integer solution was proposed by Novianingsih *et al.* [25].

The Repeated Constrained Column-Generation (RCCG) is a CG based algorithm for the two-staged SSSCSP proposed by Cui and Zhao [26] that solves the sub-problems as unconstrained ones, then, repeatedly, the residual problems are also solved through CG solving the (constrained) sub-problem until all demand is satisfied.

3.2. Location

In this section, we present discretization techniques that allows to enumerate the possible positions for item placement aiming the reduction of the search space. In addition, we present methods that allow to keep track of the empty spaces in the object which result from the placement of items.

Herz [27] presented a recursive algorithm for the unconstrained SLOPP with guillotine cuts that considers for the placement of items only discretization points obtained from the linear combinations of the items sizes (see Figure 15). Patterns wherein the items are placed at discretization points at the bottom-left-most possible position without overlapping are usually referred to as Normal Patterns (Christofides and Whitlock [28]).

Discretization Points

Considering $L_{min} = L - \min_{1 \leq i \leq m} \{l_i\}$ and $H_{min} = H - \min_{1 \leq i \leq m} \{h_i\}$, the discretization points are calculated as follows:

$$X_d = \{p \in \mathbb{Z} | p = \sum_{i=1}^m l_i z_i, 0 \leq p \leq L_{min}, 0 \leq z_i \leq d_i, z_i \in \mathbb{Z}, i = 1, \dots, m\} \quad (1)$$

$$Y_d = \{q \in \mathbb{Z} | q = \sum_{i=1}^m h_i z_i, 0 \leq q \leq H_{min}, 0 \leq z_i \leq d_i, z_i \in \mathbb{Z}, i = 1, \dots, m\} \quad (2)$$

Figure 15. Discretization Points.

A method (and supporting data structure) was presented by Chazelle [29] to report all the possible locations in which an item can be placed considering the presence of empty spaces that occur from the placement of other items.

An algorithm to calculate the set of discretization points named as Useful Numbers was proposed by Carnieri *et al.* [30].

The Reduced Raster Points (see Figure 16), a subset of the Discretization Points presented by Herz [27], were described by Scheithauer and Terno [31].

Reduced Raster Points

The Reduced Raster Points are calculated as follows:

$$X_r = \{(L - p)_x | \forall p \in X_d\} \cup \{0\} \text{ with } (L - p)_x = \max\{s \in X_d | s \leq L - p\} \quad (3)$$

$$Y_r = \{(H - q)_y | \forall q \in Y_d\} \cup \{0\} \text{ with } (H - q)_y = \max\{t \in Y_d | t \leq H - q\} \quad (4)$$

Figure 16. Reduced Raster Points.

To keep track of the empty rectangular spaces (ERS) that are created after the placement of an item, Lai and Chan [6] presented the Difference Process that creates interval lists to identify current ERSs. This process, first, places the item inside a given ERS, then generates the new ERSs that result from the intersection of the item with the existing ERSs and removes intersected ERSs. The last step removes the ERSs that are infinitely thin or are totally inscribed by other ERSs. This method is illustrated in Figure 17, in which the darker rectangles depict the available ERS at the beginning of the process (a) and at the end of each item placement (b and c).

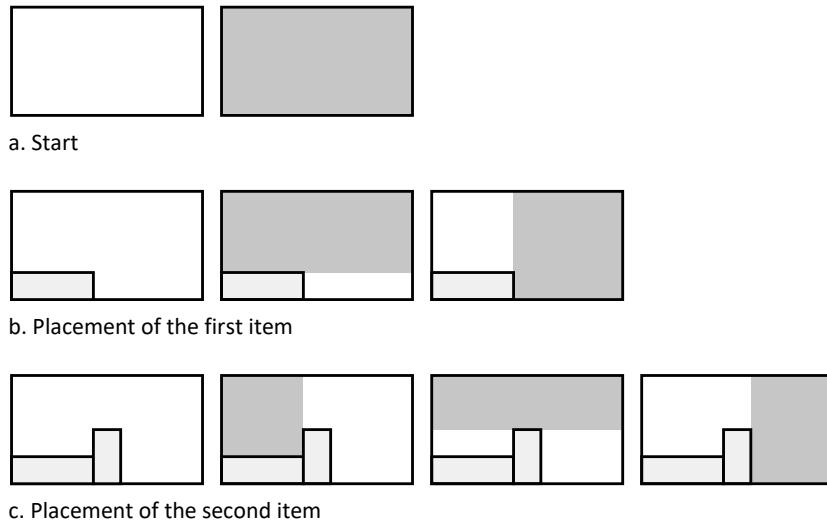


Figure 17. Difference Process.

Considering the already packed items, Martello *et al.* [32] presented an algorithm that identify the locations, called Corner Points, where new items can be placed. In Figure 18, the bullets identify the Corner Points, i.e., points where the slope of the black line changes from vertical to horizontal.

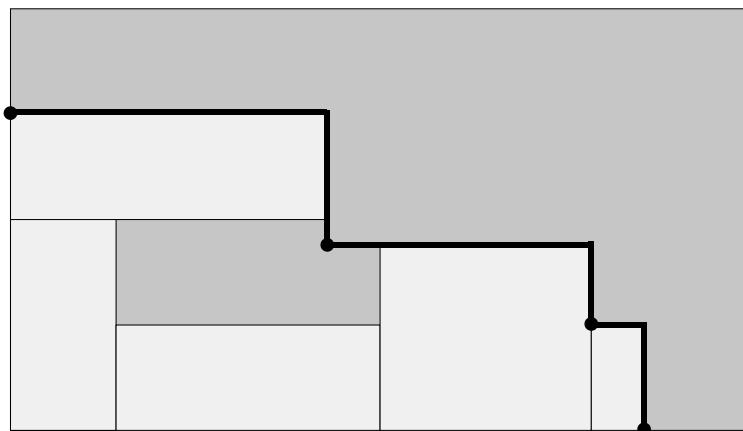


Figure 18. Corner Points.

An extension to the Corner Points, called Extreme Points, was presented by Crainic *et al.* [33] that considers corners that are inside the boundaries defined by the Corner Points. Figure 19 depicts the Extreme Points wherein the darker grey area the region that was not considered by the Corner Points.

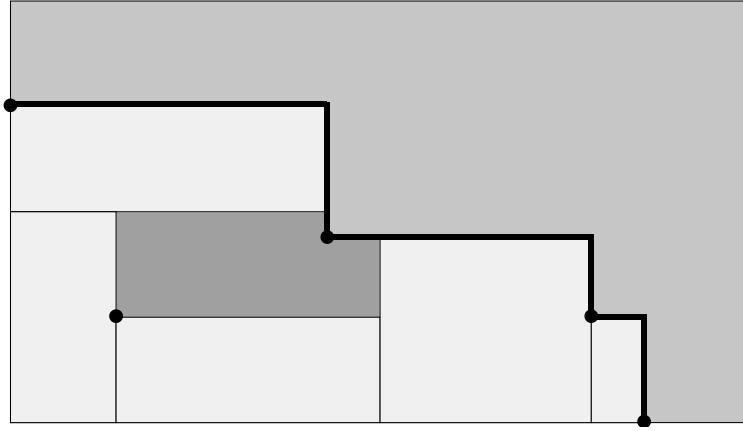


Figure 19. Extreme Points.

Wei *et al.* [34] presented a greedy heuristic to solve the SLOPP using the concept of skyline representation of a pattern as a sequence of line segments (s) expressing the rectilinear contour of the current pattern where the y coordinate of s_i is different of the y coordinate of s_{i+1} and the x coordinate of the right endpoint of s_i is the same as the x coordinate of the left endpoint of s_{i+1} .

To keep track of the gaps resulting from the placement of items, Bennell *et al.* [35] used a list of points (x, y) that define the current profile of packed items. To find the lowest gap, the minimum y value is identified (y_{min}), and the gap width is calculated considering the next x point in the list (if last, considers L) and the gap height as $H - y_{min}$.

A review on discretization points types, namely Discretization Points, Useful Numbers, Reduced Raster Points and Corners Points was presented by Cunha and Queiroz [36]. The authors denoted an equivalence on the number of points obtained by Discretization Points, Useful Numbers and Corners Points. The authors concluded that the Reduced Raster Points provide a higher reduction in the number of points, notwithstanding the lack of investigation on whether there is or not loss of generality applying this discretization type. We refer to Nascimento *et al.* [37] and Cunha and Queiroz [38] for further comparative studies on discretization points.

3.3. Exact Methods

An exact algorithm for the unconstrained two-staged SKP was proposed by Gilmore and Gomory [21] solving $m + 1$ one-dimensional SKP, one SKP for each of the items types, creating m strips and +1 to fill the object with those strips.

A characterization of knapsack functions and a Dynamic Programming approach for the unconstrained SKP with guillotine cuts was proposed by Gilmore and Gomory [39], later improved by Russo *et al.* [40] [41].

A recursive algorithm for the unconstrained SLOPP with guillotine cuts was presented by Herz [27] that reduces the computational effort through the use of bounds, memorization, and considering for the placement of items the linear combinations of the items dimensions, i.e., discretization points (see Figure 15). Based on this work, Hifi and Zissimopoulos [42] presented an exact algorithm providing improved bounds and optimality criteria to reduce the branching on the generated tree.

A tree-search algorithm for the SLOPP with guillotine cuts was presented by Christofides and Whitlock [28] that reduces the search space not generating equivalent patterns, i.e., patterns that contain the same items but with different layouts, and using upper bounds calculated by means of two methods, the Dynamic Programming procedure proposed by Gilmore and Gomory [39] to solve the unconstrained related problem and a method proposed by Desler and Hakimi [43] to solve the Transportation Problem (see Hitchcock [44]). Improvements to this approach were introduced by Christofides and Hadjiconstantinou [45] and by Hifi and Zissimopoulos [46].

Making use of an upper bound obtained from the Lagrangean Relaxation (see Fisher [47]) of the problem and optimized through Subgradient Optimization (see Shor [48]), Beasley [49] presented a tree-search procedure for the SLOPP.

Martello and Vigo [50] showed that the Continuous Lower Bound ($CLB = \lceil \sum_{i=1}^m l_i h_i / LH \rceil$) for the SSBPP has a worst-case performance ratio of $\frac{1}{4}$ and presented new lower bounds that are used in a Branch-and-Bound (see Land and Doig [51], and Agin[52]) to solve to optimality this problem. Clautiaux *et al.* [53] proposed an improvement to this Branch-and-Bound to avoid equivalent patterns, and presented a second exact algorithm based on a new problem

relaxation. A Branch-and-Bound using the Corner Points for finding possible positions for placing an item was presented by Martello *et al.* [32] for the three-dimensional SBSBPP.

A tree-search algorithm for the d -dimensional knapsack problem using a graph-theoretical characterization of feasible packings was presented by Fekete and Schepers [54][55][56]. In this characterization, if no overlapping occurs in both graphs (x and y axis projection) the pattern is feasible (see Figure 20), otherwise unfeasible (see Figure 21, wherein the overlapping edges are illustrated bolder and darker than the others). The authors refer that this characterization can be easily extended to higher dimensional problems. These technical reports were later revised and published in [57], [58] and [59].

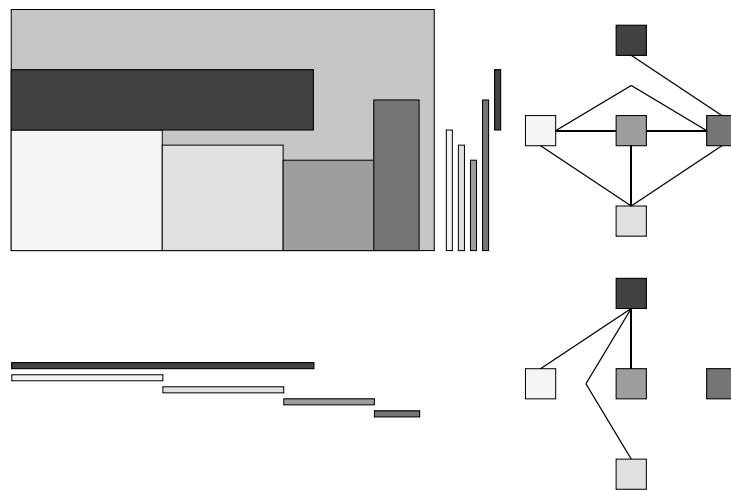


Figure 20. Fekete and Schepers' graph-theoretical characterization – Feasible pattern.

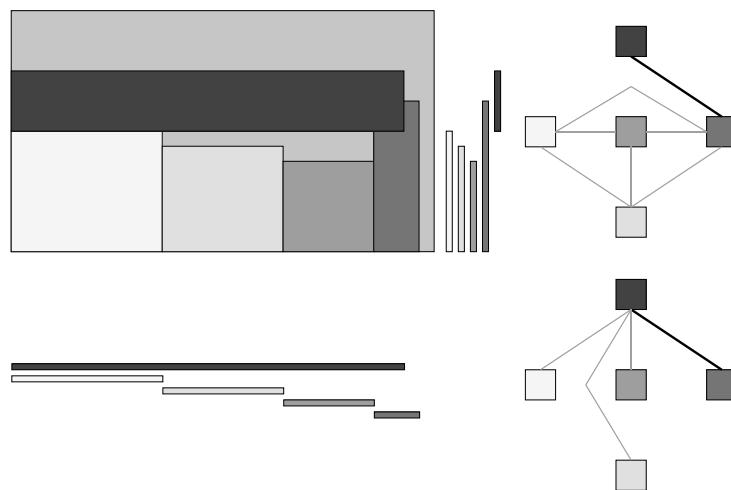


Figure 21. Fekete and Schepers' graph-theoretical characterization – Unfeasible pattern.

Ferreira and Oliveira [60] presented some degenerated cases that can occur when applying Fekete and Schepers procedure. Ferreira and Oliveira [61] presented an additional property to the Fekete *et al.* [59] graph-based algorithm to avoid degenerative packing situations, verifying that the packing is actually constrained inside the object boundaries.

Lower bounds that dominate the ones presented by Martello and Vigo [50] and Fekete and Schepers [55] were presented by Boschetti and Mingozzi [62].

Silveira and Morabito [63] proposed a Dynamic Programming and Subgradient Optimization based algorithm to solve the SLOPP with guillotine cuts.

A two-stage exact algorithm to solve the SKP making use of problem reductions and new upper bounds to reduce the search space was presented by Baldacci and Boschetti [64]. The upper bounds are embedded in an enumeration tree, and at each integer solution found a feasibility check is performed to verify if the subset of items can generate a feasible layout. The feasibility tests are performed solving to optimality a called Feasibility Problem that the authors mathematically formulate and present.

Pisinger and Sigurd [65] presented a Branch-and-Price (see Barnhart *et al.* [66]) algorithm for the SBSBPP with guillotine cuts. Branch-and-Price based algorithms were also proposed by Puchinger and Raidl [67] for the three-staged SBSBPP and by Mrad *et al.* [68] for the two-staged SSSCSP.

Clautiaux *et al.* [69] presented a Constraint-based Scheduling model (see Baptiste *et al.* [70]) for the SKP and a Branch-and-Bound algorithm considering this model.

An integer programming formulation and upper bounds (obtained through the relaxation of the mathematical formulation) for the SLOPP were proposed by Boschetti *et al.* [71]. Lodi and Monaci [72] presented two integer linear programming models for the two-staged SLOPP and extensions to deal with non-oriented, unconstrained and double-constrained problems. The authors presented some linear inequalities that remove symmetric solutions from the solution space highly reducing the computational effort required to solve the models. Silva *et al.* [13] proposed an integer programming model for the two- and three-staged SSSCSP. Due to the flexibility of the model, other issues were addressed such as the rotation of the items, the lengths of the cuts, and the value of the remaining plates. The model proposed by Silva *et*

al. [13] was extended in Furini *et al.* [73] for the SKP with guillotine cuts as for the SSSCSP, ODP, and MSSCSP.

Based on the work of Carvalho [74], Macedo *et al.* [75] presented an Arc-Flow (see Wolsey [76]) model for the two-staged SSSCSP. Brandão and Pedroso [77] presented an Arc-Flow formulation for the Vector Packing Problem (VPP; see Coffman *et al.* [78]) that can be applied to model other cutting and packing problems through the reduction to a VPP.

3.4. Non-exact Methods

The non-exact methods for solving cutting and packing problems will be presented next. This subsection starts with a review of some of the most used and referred placement strategies, i.e., policy to apply in order to place an item into a given layout, then, the heuristics and the solution methods based on metaheuristics are also presented.

3.4.1. Placement Strategies

Although most research made on two-dimensional cutting and packing problems deals only with orthogonal patterns, Cani [79] demonstrated that in practice non-orthogonal cuts or packing must be considered when it is required the absolute minimal area as they can lead to better arrangements of items or even make possible the inclusion of items that the orthogonal case cannot handle. Figure 22 depicts to the left a white square object and three grey items and it can be noticed that the longest item cannot be orthogonally placed into the square. To the right, the figure depicts a feasible non-orthogonal pattern considering the same object and items.

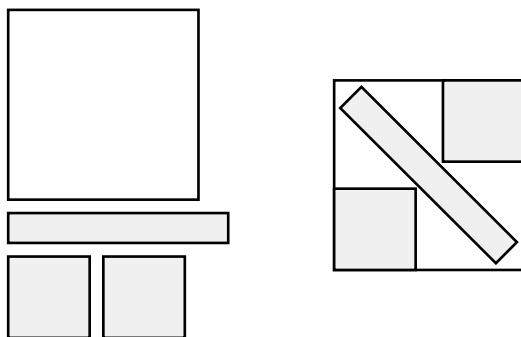


Figure 22. Non-orthogonal packing.

Baker *et al.* [80] presented a study on heuristics for the ODP in which each item is packed, in turn, at the lowest possible position, and then left-justified without overlapping, i.e., Bottom-

Left (or BL for short) heuristics. The authors evaluated the influence of the order in which items are packed, concluding that these heuristics achieve reasonable results when the items are ordered by non-increasing lengths.

The Hybrid First-Fit (HFF) heuristic for the SSBPP with guillotine cuts, proposed by Chung *et al.* [81], starts ordering the items by non-increasing heights, then, in turn, pack the items through a BL policy at the lowest existent strip that has room to fit the current item. If none exists, a new strip is created with a length equal to the bin length and the height equal to the item height. The generated strips are packed into bins by the First-Fit Decreasing (FFD) heuristic. The First-Fit (FF) is an algorithm for the one-dimensional case that places each item in turn in the lowest indexed bin in which it fits, while the First-Fit Decreasing (FFD), places the items as in FF, but assumes that the items are ordered by non-increasing height (see Johnson [82] and Johnson *et al.* [83] for further details).

A Bottom-Left approach was proposed by Jakobs [84] that slides the items from the top-right corner, while possible, downwards until it finds an item or object edge, then to the left-most position available (see Figure 23).

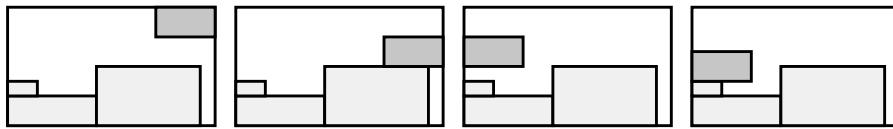


Figure 23. Jakobs' BL Heuristic.

The Bottom-Left Fill (BLF) was presented by Hopper and Turton [85] as an improvement to the Bottom-Left approach used by Jakobs [84], as the later can create large empty regions due to blocking items. To overcome this weakness, BLF places directly the items in the left-most and lowest sufficiently large empty region.

An improvement to the Bottom-Left heuristic of Jakobs [84] was proposed by Liu and Teng [86] in which when moving to the left, whenever possible, the item is placed as downward as possible first, as depicted in Figure 24.

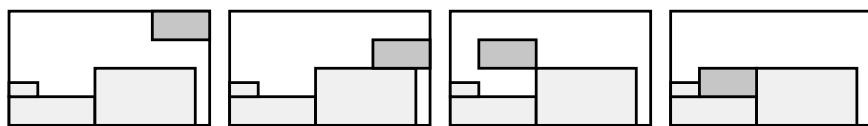


Figure 24. Liu and Teng improved BL algorithm.

Lodi *et al.* [87] presented two heuristics for the SSBPP, both starting with *CLB* opened bins and opening new ones when none of the remaining item fits on the opened bins. The Alternate Directions heuristic starts sorting the items with non-increasing heights, then packs a subset of the items into the bins following a Best-Fit Decreasing policy, i.e., packing the current item onto the shelf that minimizes the residual horizontal space. The remaining items are packed, alternatively, from left-to-right and from right-to-left at the lowest position possible. The Touching Perimeter heuristic starts by sorting the items by non-increasing area, then the position/bin to pack an item is done calculating the percentage of the perimeter that touches edges, either of the bin or other items.

The Efficient Management of Holes, presented by Beraudo *et al.* [88], iteratively packs the items as follows: 1) tries to place the item in the left margin stacked over already placed items, 2) if the item is not placed in the previous step, tries to place the current item in the leftmost free empty space, i.e., hole, and 3) if the item is not already placed, tries to place it in the object bottom edge at the right of already placed items.

The Best-Fit heuristic, presented by Burke *et al.* [89], unlike most approaches in which the items are pre-sorted and then placed in the object one at a time, examines the lowest available space and then evaluate the best item to place at this position. This heuristic does not need to search for each of the free locations due to the use of an array that identifies the height occupied defining the packing skyline as depicted in Figure 25.

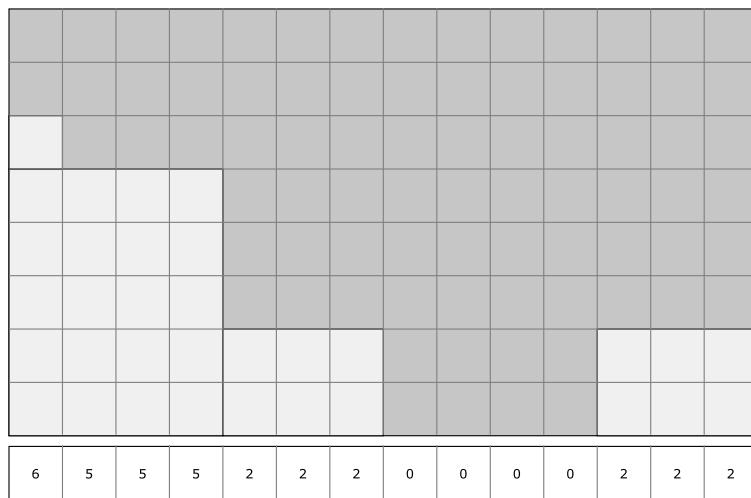


Figure 25. Burke *et al.* Array of Occupied Positions.

Gonçalves and Resende [90] presented an Genetic Algorithm that considers two placement strategies, namely Bottom-Left (BL) and Left-Bottom (LB). Using the Difference Process to

keep track of the ERS, the BL places the item in the closest ERS relatively to the bottom-left corner of the object, while LB places the item in the closest ERS to left-bottom corner of the object. The LB was introduced to overcome the inability of the BL to attain, in some situation, the optimal solution. Figure 26 a. illustrates the packing for the following sequence, considering BL and LB, 3 BL, 2 BL, 1 LB, 4 BL. Figure 26 b. illustrates the same sequence packed only with BL, i.e., 3 BL, 2 BL, 1 BL, 4 BL, and as it can be observed the last item cannot be placed.

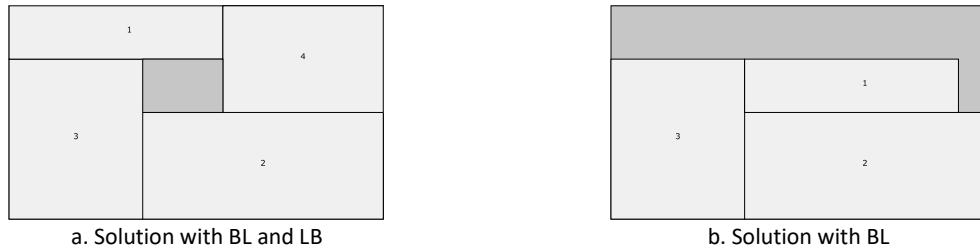


Figure 26. Solutions generated by the placement procedures BL+LB and BL.

The Four Corners heuristic, proposed by Binkley and Hagiwara [91], divides a genome (that represents a sequence of items) into four sets, each of them to be packed in a different object corner. Alternating through the four sets, the current item is packed as close as possible to its assigned corner.

The Least Wasted First Heuristic, presented by Wei *et al.* [92], makes use of the Corner Points and to reduce the search space, points are discarded when they cannot accommodate unpacked items. The items are packed in the location that leaves the empty space as smooth as possible, e.g., packing the darker grey item on the location depicted in Figure 27 provides a smooth packing since its dimensions are the same as the lines that define the Corner Point.

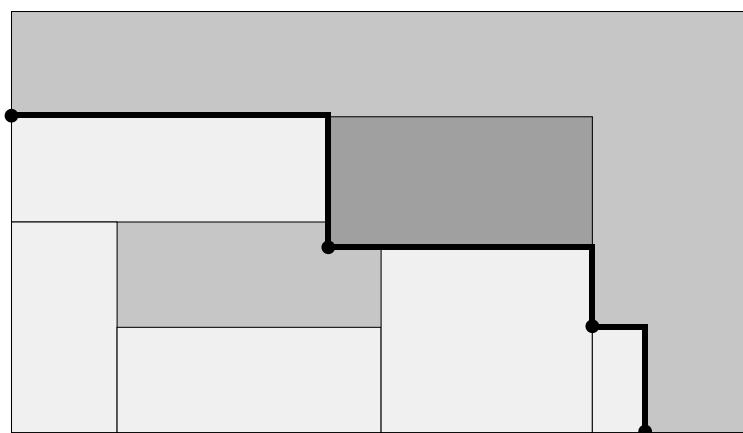


Figure 27. Smooth Packing.

The Meet-in-the-Middle (MIM) principle is presented by Côté and Lori [93] as an alternative to the Normal Patterns. MIM packing divides the first dimension, e.g., length, by a given threshold value, then places the items whose left border is at the left of the threshold as left as possible and the remaining to the right. This process is repeated for the successive dimensions, e.g., for the height dimension, places the items at the top or at the bottom. Figure 28 depicts a general pattern, corresponding Normal Pattern and the MIM pattern with a threshold value of $L/2$ and $H/2$, for length and height dimensions respectively.

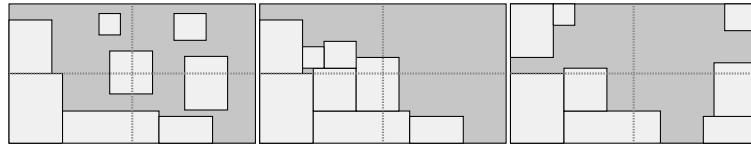


Figure 28. Patterns: General (left), Normal (middle), Meet-in-the-Middle (right).

3.4.2. Heuristics

Heuristics to solve the ODP and the SSBSP were presented by Bengtsson [94] which aim to be efficient solution methods for large-scale instances due to the small memory requirements. The heuristic for the ODP is a procedure that recursively packs the items. The heuristic for the SSBSP starts with a rough distribution of items into bins and repeatedly discards the bin that presents more unused space. The items of the discarded bin are used to improve the remaining bins.

Wang [95] proposed two heuristics to solve the guillotine SLOPP. The underlying algorithm of these heuristics is the same, differing in the aspiration criterion used as the condition for the acceptance of the percentage of waste generated. This implicit enumeration approach builds successively a larger guillotine rectangle from smaller ones. Parada *et al.* [96] presented a computational comparison on approaches based on Wang's algorithm [95], more specifically, Oliveira and Ferreira [97] and Parada *et al.* [98] [99] [100].

Beasley [101] presented an exact algorithm and a heuristic based on Dynamic Programming for the (non-)staged unconstrained SLOPP. The author demonstrated that considering only Normal Patterns and discretization points can lead to improved recursion since it permits to reduce the search space. Denoting that the exact method can become computationally too demanding for a large set of discretization points, a heuristic is proposed reducing the set size. Morabito and Arenales [102] compared the heuristic of Beasley [101] and the algorithm

of Gilmore and Gomory [21] to solve large-scale unconstrained non-staged SLOPP. Although the latter considers two-staged problems, it provides feasible solutions for non-staged problems⁴. The authors remarked that since Beasley's heuristic removes some discretization points it can, in some cases, exclude points essential to achieve optimality. Faced with increasing object size and number of items, Beasley's heuristic was not able to attain better result than the one proposed by Gilmore and Gomory.

Berkey and Wang [103] presented several heuristics for the SBSBPP adapted from heuristics found in the literature for the ODP. Adaptations to these heuristics for the non-oriented SBSBPP were proposed by Lodi *et al.* [104].

Using an And/Or-Graph to represent the solutions, Morabito *et al.* [12] proposed a hybrid search heuristic that combines depth-first and hill-climbing search strategies for the unconstrained SLOPP with guillotine cuts.

The HBP heuristic, proposed by Boschetti and Mingozzi [105] to solve the SBSBPP, generates solutions using two placement methods and different pricing rules. At the end of each iteration, the value of the items is updated and a new solution is generated with these values.

Hifi and M'Hallah [106] presented algorithms for the two-staged SLOPP, namely, Strip Generation Algorithm (SGA), Extended SGA (ESGA), and Hill-climbing ESGA (HESGA). The SGA solves the problem as a two-stage algorithm, first generating a set of strips then searching for good combinations of those strips both solving bounded one-dimensional SKP. The ESGA fills the object dividing it into two sections, one filled with the SGA while the other is filled using an alternative procedure that makes use of horizontal discretization points. Finally, the HESGA combines the ESGA with hill-climbing strategies.

The Set-Covering Heuristic (SCH), presented by Monaci and Toth [107] for the SBSBPP, formulates the problem as a Set-Covering Problem (see Caprara *et al.* [108]) and solves it through a two-phase heuristic. The Column-Generation phase generates, thought greedy heuristics to achieve diversity, a large set of columns that define the Set-Covering instance to be solved in the Column-Optimization phase.

⁴ Farley [184] analysed the trade-off between non-exact two-staged patterns and non-staged ones and concluded that the waste reduction obtained with non-staged may not justify, in some cases, the extra computational effort required by this cutting style.

Five greedy heuristics for the SLOPP with guillotine cuts were presented by Hadjiconstantinou and Iori [109], namely, the HC_{KP} , HC_{HV} , HC_{GAP} , HC_{ORD} , and HC_{ORD2} . The HC_{KP} , considers some item ordering, iteratively pack, using a BL policy, one item into a strip, then it is completed solving a one-dimensional SKP. The HC_{HV} pack the items alternatively on top of each other or side by side until no more items fit in this direction. The HC_{GAP} solves a one-dimensional SKP to pack vertically items then for each of them a horizontal strip is created and completed solving a Generalized Assignment Problem (GAP; see Cattrysse and Van Wassenhove [110]). The HC_{ord} creates for each item a strip using HC_{KP} then the best strips are packed considering the values of the items in the strip until their exit strips that fit. The HC_{ord2} is similar to the previous one but recreate the strips for each unpacked item after the packing of the best strip created.

Huang and Chen [111] proposed two heuristics to solve the SKP which placement strategy is to place the current item in a corner defined either by other items or by the edges of the object. When evaluating the possible corners to place the items, a caving degree is calculated to achieve high area usage. The second heuristic is a backtracking process over the solution generated by the first heuristic.

The Extreme Point First Fit Decreasing (EP-FFD) heuristic and Extreme Point Best Fit Decreasing (EP-BFD) heuristic were presented by Crainic *et al.* [33]. Both heuristics consider a pre-ordering of the items and then, in turn, the items are placed into the bins. The EP-FFD places the items at the lower and the left-most Extreme Point in which it fits. If the current bin cannot accommodate a new one is opened, and the item is placed in the left-bottom corner. The EP-BFD places the item into the bins that present the best merit function. The authors present and compare several options for the calculation of the merit function.

The Least Wasted First Heuristic, proposed by Wei *et al.* [92], makes use of the Smooth Packing to place the items and since the ordering influence the quality of the solution generated by this placement method, the authors apply a random local search which alters the order of the items at each call, trying to obtain a better solution.

Based on Christofides and Hadjiconstantinou's algorithm [45], Morabito and Pureza [112] presented a heuristic using And/Or-Graph search approach for the (un)weighted SLOPP with guillotine cuts.

The Sequential Grouping Heuristic (SGH) was presented by Cui *et al.* [113] to solve the SSSCSP with guillotine cuts where the main objective is the input minimization and the secondary objective is the minimization of the number of patterns generated. This heuristic uses the sequential value correction proposed by Belov and Scheithauer [114] and the heuristic proposed by Cui [115] for the SLOPP.

Three heuristics for the SBSBPP with guillotine cuts were proposed by Fleszar [116], namely the First-Fit Insertion Heuristic (FFIH), Best-Fit Insertion Heuristic (BFIH), and Critical-Fit Insertion Heuristic (CFIH). A tree representation of patterns is used, in which, leaf nodes represent items and the other nodes represent the vertical or horizontal cuts. Considering the items sorted by their area, the FFIH(/BFIH) packs the items in the first(/best) bin using a fitness function to evaluate the best location. The CFIH, Iteratively, evaluates all the unpacked items and packs the critical item on the best insertion point considering the fitness function adopted. An improvement procedure is presented in which, at each iteration, the last pattern is removed from the current solution and the unpacked items are packed into a new solution using FFIH. The patterns from the current solution and the new solution constitute a new feasible solution, being the new best solution if this (combined) solution uses fewer bins.

Alvelos *et al.* [117] presented heuristics for the two- and three-staged SBSBPP based on the SearchCol framework (see Alvelos *et al.* [118] and Alvelos *et al.* [119]).

Cui *et al.* [120] presented a heuristic for the SBSBPP that generates a given number of cutting plans, adjusting the items' value after each pattern generation through a correction formula. The correction formula prioritizes items that do not combine well with the current pattern and larger items more difficult to pack. The patterns are generated in a similar manner as in the work presented by Wei *et al.* [92].

The Residual-Space-Maximized Packing heuristic, presented by Wang and Chen [121], for the ODP and SBSBPP considers that at each packing step the residual space must be maximized. The Difference Process is used to keep track of the ERS, and when packing an item all the ERS corners are evaluated, choosing the one that generates the largest free region after packing.

3.4.3. Metaheuristics

The Genetic-And/Or-Graph (GAO) is a Genetic Algorithm (see Goldberg [122]) proposed by Parada *et al.* [99] to solve the SLOPP with guillotine cuts which represents solutions by mean

of a string. The string represents a binary tree associated with the cutting pattern where the items are represented by lowercase letters and the operators V and H represent vertical and horizontal cuts, respectively. The algorithm applies crossover operations using the above operators to create from two patterns, four new ones.

A Genetic Algorithm for the Irregular two-dimensional ODP was proposed by Jakobs [84]. Although capable to deal with polygons, the author remarks that the algorithm can, alternatively, be used on rectangular shapes that embed the polygons to decrease the computational effort. The author determines the embedding rectangle with the minimum area on all polygons and then applies the Genetic Algorithm. A last step is performed when dealing with this embedding approach, called the Shrinking-step, that shifts the polygons closer to each other.

A Genetic Algorithm for the SKP was proposed by Gonçalves and Resende [90] wherein the chromosome, besides the representation of the items packing order, contains also the corresponding packing strategy. This algorithm uses two placement strategies, namely Bottom-Left (BL) and Left-Bottom (LB). Different layouts with the same assignments will have the same trim loss, but one can present larger ERSs and so will have a higher improvement potential, as depicted in Figure 29. Taking this into account, Gonçalves [123] present a Genetic Algorithm that use a modified trim loss evaluation to measure the quality of a pattern. Gonçalves and Resende [124] presented a parallel multi-population Genetic Algorithm for solving the SLOPP.

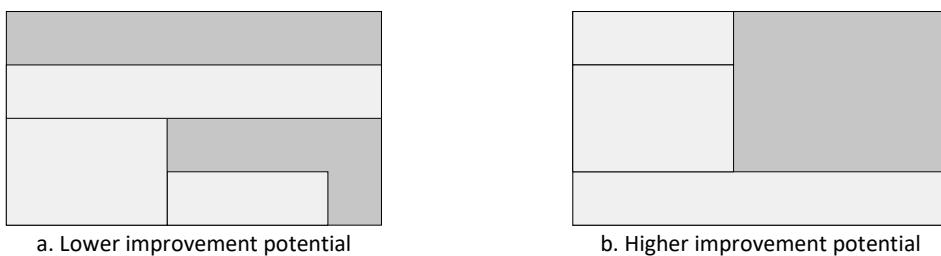


Figure 29. Potential improvement.

Bortfeldt and Winter [125] presented a Genetic Algorithm approach to solve the (un)constrained (non-)guillotine SKP/SLOPP.

A Multi-Crossover Genetic Algorithm (see Lee [126]) for the SBSBPP with due dates was proposed by Bennell *et al.* [35] where the objective is to minimize the maximum lateness of

the items and to minimize the number of bins used. This heuristic searches for the lowest available gap in the current bin and packs the item that best fills this gap.

Simulated Annealing (see Kirkpatrick *et al.* [127]) approaches were proposed by Lai and Chan [6] for solving the SKP, by Parada *et al.* [100] for the SLOPP with guillotine cuts and by Faina [128] for the SSSCSP.

Egeblad and Pisinger [129] presented integer programming formulations for the two- and three-dimensional SKP and a Simulated Annealing that make use of the Sequence Pair representation proposed by Murata *et al.* [130] for the VLSI problem. Figure 30 depicts a packing pattern represented by the sequence of $A = < 3, 2, 5, 6, 1, 4 >$ (graph to the left) and $B = < 1, 2, 4, 3, 5, 6 >$ (graph to the right). Considering the two sets A and B , if item i precedes j in A and B , then i is placed to the left of j . If i succeeds j in A but precedes in B , item i is placed below j .

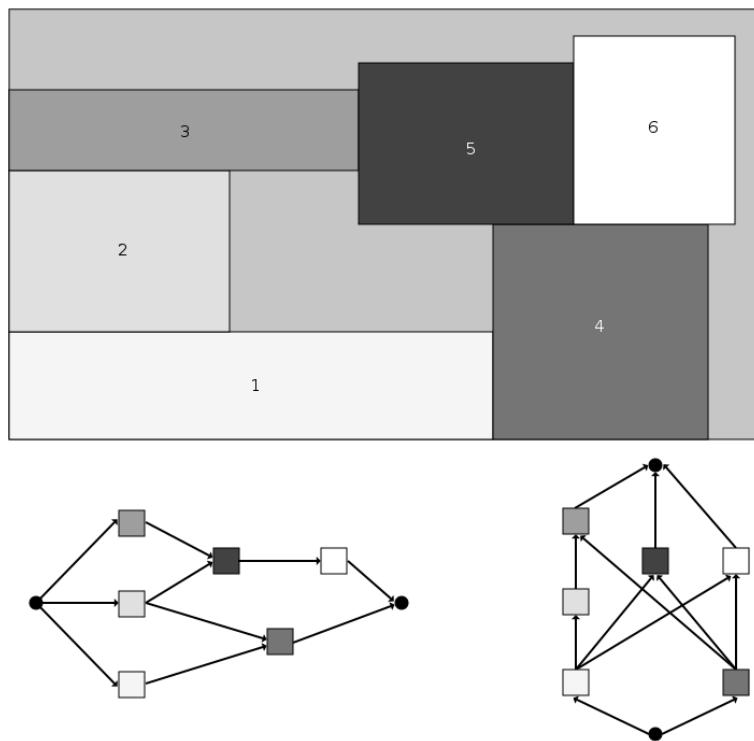


Figure 30. Sequence Pair representation.

Binkley and Hagiwara [91] presented a Simulated Annealing and a Genetic Algorithm for the SKP using the Four Corners heuristic for item placement.

Hopper and Turton [131], and Leung *et al.* [132] presented comparative studies on the performance of Genetic Algorithm and Simulated Annealing. Leung *et al.* [133] compared a

Genetic Algorithm and a hybrid meta-heuristic (Genetic Algorithm with Simulated Annealing) called MSAGA for the SKP. The objective of this work was to verify if the hybrid approach could prevent the early convergence observed in the Genetic Algorithm. The results point to the superiority of the hybrid approach. The decoder, i.e., placement method, uses the Difference Process proposed by Lai and Chan [6].

A new non-linear mathematical formulation for the SLOPP was presented by Beasley [134] giving indications on how to extend the formulation to deal with defective areas, multiple size stock objects and item rotation. Based on this formulation, the author proposed a Population Heuristic (see Beasley [135]) and reported the results obtained considering large-scale instances ($m = 1000$). Based in this work, Beraudo *et al.* [136] presented an evolutionary approach using the Efficient Management of Holes. An Evolutionary Algorithm for the SKP/SLOPP using a tree-search placement algorithm using a bottom-left placement policy was presented by Kierkosz and Luczak [137].

Lodi *et al.* [104], [138] presented a Tabu Search (see Glover [139]) approaches for the SBSBPP with guillotine cuts and in [87], presented a Unified Tabu Search Framework meant to be adaptable for each specific problem changing uniquely the inner heuristics to explore the neighbourhoods. A Tabu Search approach was also presented by Alvarez-Valdés *et al.* [140] for solving the SLOPP using an alternative objective function, that considers symmetry, number of empty rectangles, waste concentration and feasibility (for doubly constrained problems), to prevent equal objective function values on solutions that although with a different layout have the same assignments.

Besides a Tabu Search, Alvarez-Valdés *et al.* [141] presented a Greedy Randomized Adaptive Search Procedure (GRASP, see Feo and Resende [142]) and a Path Relinking (see Glover [143]) to solve large-scale (un)constrained (un)weighted SLOPP with guillotine cuts. Alvarez-Valdés *et al.* [144] proposed a GRASP to solve the double-constrained SLOPP and Alvarez-Valdés *et al.* [145] presented two GRASP and a Path Relinking approach for the two-staged SLOPP. In the later, the Path Relinking makes use of solutions obtained by both GRASP since one provides high-quality solutions while the other provides a more diverse set of solutions. Parreño *et al.* [146] presented a GRASP with Variable Neighbourhood Descent (VND, see Hansen and Mladenović [147]) based heuristic for the two- and three-dimensional SBSBPP.

Alvelos *et al.* [148] presented a VND algorithm for the two- and three-staged SBSBPP, while Chan *et al.* [149] presented a VND to solve the two-staged SBSBPP. Chan *et al.* [150] presented a heuristic called Stochastic Neighbourhood Structures (SNS) for the two- and three-staged SBSBPP. The main difference between SNS and VND is that all neighbourhood structures explored are stochastics.

A Variable Neighbourhood Search (VNS, see Mladenović and Hansen [151]) heuristic for the three-staged SSSCSP using the Ruin and Recreate Principle was proposed by Dusberger and Raidl [152]. The solution is represented by the root node of a cutting tree, in which children's root represent individual bins, each consecutive level represents guillotine cuts, and the leaves represent individual items. Three greedy heuristics are presented, namely, Three-staged First Fit Decreasing Height with Rotations (3SFFDHR), 3SFFDHR preceded by a matching step (MATCH) and Fill Strip (FS). The 3SFFDHR first sorts the items by non-increasing height then, iteratively, tries to accommodate the current item in the cutting tree at the first possible position using a post-order traversal. The MATCH includes a pre-processing stage, based on the work proposed by Fritsch and Vornberger [153], pairing items into meta-rectangles that will be packed alongside with the remaining items using the 3SFFDHR. The FS is an adaptation of the FFFWS proposed by Puchinger *et al.* [154]. The solutions are created using the above greedy heuristics, being the best one selected as the base solution. Following the Ruin and Recreate Principle, parts of the base solution are destroyed and recreated using the previously described heuristics. The authors present an integer linear programming model for the two-staged case that can be used as an alternative method to recreate the solution.

A Guided Local Search (GLS, see Voudouris and Tsang [155]) based heuristic was presented by Faroe *et al.* [156] for the two- and three-dimensional SBSBPP. The heuristic starts with an upper bound on the number of available bins and iteratively decreases this number. This process is repeated until the time limit is reached or the current solution is the same as a calculated lower bound.

Jiang *et al.* [157] presented a hybrid algorithm based on Particle Swarm Optimization (see Kennedy and Eberhart [158]) and Simulated Annealing to solve the SLOPP. Bao *et al.* [159] presented an Artificial Fish Swarm Algorithm (see Li *et al.* [160]) for the SKP comparing the results with the ones obtained with the Particle Swarm Optimization. Omar and

Ramakrishnan [161] presented an Evolutionary Particle Swarm Optimization (EPSO) for the SSBPP combining PSO with concepts of Evolutionary Algorithms to diversify the search.

A Beam Search (see Ow and Morton [162]) algorithm was proposed by Hifi *et al.* [163] for the two-staged SLOPP and based on this work, Hifi *et al.* [164] presented a parallel Beam Search.

3.5. Surveys and Reviews

The following list presents, in chronological order, the surveys and reviews found in the literature focusing solution approaches for cutting and packing problems.

- Golden [165] surveyed the solution approaches found in the literature for the CSP.
- Hinxman [166] presented in this work a taxonomy of assortment and trim loss problems, and surveyed the solution methods found in the literature for these problems.
- Coffman *et al.* [78] presented an extensive survey on approximation algorithms for the BPP and for most of the variants associated with this problem.
- Dowsland and Dowsland [167] surveyed two- and three-dimensional packing and related problems focusing on models and solution approaches.
- Dyckhoff and Finke [168] presented the Dyckhoff's typology [15] and an extensive survey of the works published on cutting and packing problems.
- Lirov [169] surveyed the existent literature, discussing the domain of applicability, model, solution approaches and the related problem of these works.
- Cheng *et al.* [170] surveyed the literature on the one- and two-dimensional CSP and some of their related problem, such as bin packing problem, pallet loading problem and VLSI placement problem. The authors focused on the solution methods proposed and on the practical and industrial aspects of these problems.
- Hopper and Turton [171] presented an extensive review of metaheuristics approaches applied to the two-dimensional regular and irregular strip packing problems, such as Genetic Algorithm, Simulated Annealing, Tabu Search and Artificial Neural Networks.
- Lodi *et al.* [172] review exact, heuristic and metaheuristic approaches for the two-dimensional BPP.
- Lodi *et al.* [173] review mathematical models, lower bounds, classical approximation algorithms, and solution methods for packing problems.

- Jylänki [174] present an extended review of packing algorithms⁵. The author presents most of the packing heuristics found in the literature and performs computational tests on more than two thousand distinct algorithms.
- Chan *et al.* [175] reviews heuristics for the BPP grouping them as one-phase (items are packed directly into bins), two-phase (first, strips of items are created and then the strips are packed into bins), and local search heuristics.
- Crainic *et al.* [176] reviewed and compared models and approaches to solving the two- and three-dimensional knapsack and bin packing problems.
- Delorme *et al.* [177] reviewed mathematical models and exact algorithms for the SSBPP and SSSCSP.
- Oliveira *et al.* [178] presented an extensive review of heuristics proposed, mainly, in the last decade for the two-dimensional ODP.
- Silva *et al.* [179] presented a review of solution approaches for the two-dimensional IIPP.
- Gonzalez *et al.* [180] presented a wide overview of CSP focusing on its classification and solution approaches.

4. Conclusion

As this paper has shown, a great amount of research has been undertaken on the cutting and packing problem family. In this paper, solution methods for the two-dimensional rectangular cutting and packings problems have been reviewed with the main emphasis in two distinct problems. The first aims to generate the layout that maximizes the value of the items to cut or pack considering only one object, while the second aims to cut or pack all items using as few identical objects as possible. We review the solution methods that have been proposed in the literature for solving these problems. Although any other classification criteria could be adopted, we have chosen a simple classification scheme to group the solution approaches, classifying them as exact, heuristic or metaheuristic. Furthermore, we present methods to enumerate the possible positions for item placement and a list of surveys focusing solution approaches for cutting and packing problems.

⁵ The review and source code used can be obtained at <https://github.com/juj/RectangleBinPack>.

Appendix A

The following table (Table 4) relates the articles referred in this paper with the types of problem for which a solution method was presented. This table presents, by order of appearance, the article reference, the publication year, the problems types following the typology proposed by Wäscher *et al.* [16] and, finally, the cut types considered. The cuts are identified by G for guillotine, NG for non-guillotine, and (N)G when both cut types are considered by the corresponding article.

Table 4. Solution approaches for cutting and packing problems.

Article	Year	Problem						Cut
		IIPP	PP	KP	ODP	CSP	BPP	
El-Aal [2]	1994					•		G
Farley [3]	1983					•		G
Vasko <i>et al.</i> [4]	1989					•		G
Morábito and Garcia [5]	1998					•		G
Lai and Chan [6]	1997			•				NG
Tsai <i>et al.</i> [7]	2009					•		NG
Morabito <i>et al.</i> [12]	1992		•					G
Silva <i>et al.</i> [13]	2010					•		G
Andrade <i>et al.</i> [14]	2016					•		G
Gilmore and Gomory [17]	1961			•		•		G
Gilmore and Gomory [19]	1963			•		•		G
Gilmore and Gomory [21]	1965			•	•	•		G
Oliveira and Ferreira [22]	1994		•			•		G
Cintra <i>et al.</i> [23]	2008		•		•	•		G
Novianingsih <i>et al.</i> [25]	2012					•		G
Cui and Zhao [26]	2013					•		G
Herz [27]	1972		•					G
Christofides and Whitlock [28]	1977		•					G
Chazelle [29]	1983				•			NG
Carnieri <i>et al.</i> [30]	1994					•		G
Scheithauer and Terno [31]	1996	•						NG
Martello <i>et al.</i> [32]	2000						•	NG
Crainic <i>et al.</i> [33]	2008						•	NG
Wei <i>et al.</i> [34]	2011		•		•			NG
Bennell <i>et al.</i> [35]	2013						•	NG
Gilmore and Gomory [39]	1966			•				G
Russo <i>et al.</i> [40]	2013			•				G
Russo <i>et al.</i> [41]	2014		•					G
Hifi and Zissimopoulos [42]	1996			•				G
Christofides and Hadjiconstantinou [45]	1995		•					G
Hifi and Zissimopoulos [46]	1997		•					G
Beasley [49]	1985		•			•		NG

Article	Year	Problem						Cut
		IIPP	PP	KP	ODP	CSP	BPP	
Martello and Vigo [50]	1998						•	NG
Clautiaux <i>et al.</i> [53]	2007			•			•	NG
Fekete and Schepers [54][55][56]	2000			•				NG
Fekete and Schepers [57]	2004			•				NG
Fekete and Schepers [58]	2004			•				NG
Fekete <i>et al.</i> [59]	2007			•				NG
Ferreira and Oliveira [60]	2005			•				NG
Ferreira and Oliveira [61]	2008			•				NG
Boschetti and Mingozi [62]	2003						•	NG
Silveira and Morabito [63]	2002		•					G
Baldacci and Boschetti [64]	2007			•				NG
Pisinger and Sigurd [65]	2007						•	G
Puchinger and Raidl [67]	2007						•	G
Mrad <i>et al.</i> [68]	2013					•		G
Clautiaux <i>et al.</i> [69]	2008			•				NG
Boschetti <i>et al.</i> [71]	2002		•					NG
Lodi and Monaci [72]	2003			•				G
Furini <i>et al.</i> [73]	2016			•	•	•		G
Macedo <i>et al.</i> [75]	2010					•		G
Brandão and Pedroso [77]	2016					•	•	NG
Baker <i>et al.</i> [80]	1980				•			NG
Chung <i>et al.</i> [81]	1982						•	NG
Jakobs [84]	1996				•			NG
Hopper and Turton [85]	1999				•			NG
Liu and Teng [86]	1999				•			NG
Lodi <i>et al.</i> [87]	1999						•	(N)G
Beraudo <i>et al.</i> [88]	2004		•					NG
Burke <i>et al.</i> [89]	2004				•			NG
Gonçalves and Resende [90]	2006			•				NG
Binkley and Hagiwara [91]	2007			•			•	NG
Wei <i>et al.</i> [92]	2009			•				NG
Côté and Iori [181]	2016					•	•	(N)G
Bengtsson [94]	1982				•		•	NG
Wang [95]	1983		•			•		G
Parada <i>et al.</i> [96]	2000		•					G
Oliveira and Ferreira [97]	1990		•					G
Parada <i>et al.</i> [98]	1995		•					G
Parada <i>et al.</i> [99]	1995		•					G
Parada <i>et al.</i> [100]	1998		•					G
Beasley [101]	1985		•					G
Morabito and Arenales [102]	1995		•					G
Berkey and Wang [103]	1987						•	(N)G
Lodi <i>et al.</i> [104]	1999						•	G
Boschetti and Mingozi [105]	2003						•	NG
Hifi and M'Hallah [106]	2006		•					G
Monaci and Toth [107]	2006						•	NG

Article	Year	Problem						Cut
		IIPP	PP	KP	ODP	CSP	BPP	
Hadjiconstantinou and Iori [109]	2007		•					(N)G
Huang and Chen [111]	2007			•				NG
Morabito and Pureza [112]	2010		•					G
Cui <i>et al.</i> [113]	2013					•		G
Cui [115]	2007		•					G
Fleszar [116]	2013						•	G
Alvelos <i>et al.</i> [117]	2014						•	G
Cui <i>et al.</i> [120]	2015						•	(N)G
Wang and Chen [121]	2015				•		•	NG
Gonçalves [123]	2007			•				NG
Gonçalves and Resende [124]	2011		•	•				NG
Bortfeldt and Winter [125]	2009		•	•				(N)G
Faina [128]	1999					•		(N)G
Egleblad and Pisinger [129]	2009			•				NG
Hopper and Turton [131]	2001				•			NG
Leung <i>et al.</i> [132]	2001			•				NG
Leung <i>et al.</i> [133]	2003			•				NG
Beasley [134]	2004		•					NG
Beraudo <i>et al.</i> [136]	2005		•					NG
Kierkosz and Luczak [137]	2014		•	•				NG
Lodi <i>et al.</i> [138]	1999						•	G
Alvarez-Valdés <i>et al.</i> [140]	2007		•					NG
Alvarez-Valdés <i>et al.</i> [141]	2002		•					G
Alvarez-Valdés <i>et al.</i> [144]	2005		•					NG
Alvarez-Valdés <i>et al.</i> [145]	2007		•					G
Parreño <i>et al.</i> [146]	2010						•	NG
Alvelos <i>et al.</i> [148]	2009						•	G
Chan <i>et al.</i> [149]	2009						•	G
Chan <i>et al.</i> [150]	2011					•	•	G
Dusberger and Raidl [152]	2014					•		G
Puchinger <i>et al.</i> [154]	2004					•		G
Faroe <i>et al.</i> [156]	2003						•	NG
Jiang <i>et al.</i> [157]	2004		•					NG
Bao <i>et al.</i> [159]	2013		•					NG
Omar and Ramakrishnan [161]	2013						•	NG
Hifi <i>et al.</i> [163]	2008		•					G
Hifi <i>et al.</i> [164]	2012		•					G

References

- [1] M. R. Garey and D. S. Johnson, “A Guide to the Theory of NP-Completeness,” *Mathematical Sciences*. W.H. Freeman and Company, San Francisco, 1979.
- [2] R. M. S. A. El-Aal, “An interactive technique for the cutting stock problem with multiple objectives,” *Eur. J. Oper. Res.*, vol. 78, no. 3, pp. 304–317, 1994.

- [3] A. Farley, "Trim-loss pattern rearrangement and its relevance to the flat-glass industry," *Eur. J. Oper. Res.*, vol. 14, no. 4, pp. 386–392, 1983.
- [4] F. J. Vasko, F. E. Wolf, and K. L. Stott, "A practical solution to a fuzzy two-dimensional cutting stock problem," *Fuzzy Sets Syst.*, vol. 29, no. 3, pp. 259–275, 1989.
- [5] R. Morabito and V. Garcia, "The cutting stock problem in a hardboard industry: A case study," *Comput. Oper. Res.*, vol. 25, no. 6, pp. 469–485, 1998.
- [6] K. K. Lai and J. W. M. Chan, "Developing a simulated annealing algorithm for the cutting stock problem," *Comput. Ind. Eng.*, vol. 32, no. 1, pp. 115–127, 1997.
- [7] J. F. Tsai, P. L. Hsieh, and Y. H. Huang, "An optimization algorithm for cutting stock problems in the TFT-LCD industry," *Comput. Ind. Eng.*, vol. 57, no. 3, pp. 913–919, 2009.
- [8] P. E. Sweeney and E. Paternoster, "Cutting and packing problems: a categorized, application-orientated research bibliography," *J. Oper. Res. Soc.*, vol. 43, no. 7, pp. 691–706, 1992.
- [9] K. Singh and L. Jain, "Industrial Scope of 2D packing problems," *Natl. J. Syst. Inf. Technol.*, vol. 2, no. 2, pp. 224–237, 2009.
- [10] R. Macedo, E. M. da C. Silva, and C. Alves, "2D Cutting Stock Optimization Software Survey," 2008.
- [11] L. Kantorovich, "Mathematical methods of organizing and planning production," *Manage. Sci.*, pp. 366–423, 1960.
- [12] R. Morabito, M. N. Arenales, and V. F. Arcaro, "An and-or-graph approach for two-dimensional cutting problems," *Eur. J. Oper. Res.*, vol. 58, no. 2, pp. 263–271, 1992.
- [13] E. M. da C. Silva, F. Alvelos, and J. M. V. de Carvalho, "An integer programming model for two- and three-stage two-dimensional cutting stock problems," *Eur. J. Oper. Res.*, vol. 205, no. 3, pp. 699–708, 2010.
- [14] R. Andrade, E. G. Birgin, and R. Morabito, "Two-stage two-dimensional guillotine cutting stock problems with usable leftover," *Int. Trans. Oper. Res.*, vol. 23, no. 1–2, pp. 121–145, 2016.
- [15] H. Dyckhoff, "A typology of cutting and packing problems," *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 145–159, 1990.
- [16] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [17] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Oper. Res.*, vol. 9, no. 6, pp. 849–859, 1961.
- [18] L. R. Ford and D. R. Fulkerson, "A Suggested Computation for Maximal Multi-Commodity Network Flows," *Manage. Sci.*, vol. 50, no. 12_supplement, pp. 1778–1780, 2004.
- [19] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem-Part II," *Oper. Res.*, vol. 11, no. 6, pp. 863–888, 1963.

- [20] G. B. Dantzig, *Linear Programming and Extensions*. Princeton University Press, 1963.
- [21] P. C. Gilmore and R. E. Gomory, “Multistage cutting stock problems of two and more dimensions,” *Oper. Res.*, vol. 13, no. 1, pp. 94–120, 1965.
- [22] J. F. Oliveira and J. S. Ferreira, “A faster variant of the Gilmore and Gomory technique for cutting stock problems,” *Belgian J. of Oper. Res. Stat. Comput. Sci.*, vol. 34, no. 1, pp. 23–38, 1994.
- [23] G. F. Cintra, F. K. Miyazawa, Y. Wakabayashi, and E. C. Xavier, “Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation,” *Eur. J. Oper. Res.*, vol. 191, no. 1, pp. 61–85, 2008.
- [24] R. Bellman, “The Theory of Dynamic Programming,” *Bull. Am. Math. Soc.*, vol. 60, no. 6, pp. 503–515, 1954.
- [25] K. Novianingsih, R. Hadianti, and S. Uttunggadewa, “Column generation technique for solving two-dimensional cutting stock problems: method of stripe approach,” *J. Indones. Math. Soc.*, vol. 13, no. 2, pp. 161–172, 2012.
- [26] Y. Cui and Z. Zhao, “Heuristic for the rectangular two-dimensional single stock size cutting stock problem with two-staged patterns,” *Eur. J. Oper. Res.*, vol. 231, no. 2, pp. 288–298, 2013.
- [27] J. Herz, “Recursive Computational Procedure for Two-dimensional Stock Cutting,” *IBM J. Res. Dev.*, vol. 16, no. 5, pp. 462–469, 1972.
- [28] N. Christofides and C. Whitlock, “An Algorithm for Two-Dimensional Cutting Problems,” *Oper. Res.*, vol. 25, no. 1, pp. 30–44, 1977.
- [29] B. Chazelle, “The Bottomn-Left Bin-Packing Heuristic: An Efficient Implementation,” *IEEE Trans. Comput. Syst.*, vol. 32, no. 8, pp. 697–707, 1983.
- [30] C. Carnieri, G. A. Mendoza, and L. G. Gavinho, “Solution procedures for cutting lumber into furniture parts,” *Eur. J. Oper. Res.*, vol. 73, no. 3, pp. 495–501, 1994.
- [31] G. Scheithauer and J. Terno, “The G4-Heuristic for the Pallet Loading Problem,” *J. Oper. Res. Soc.*, vol. 47, no. 4, pp. 511–522, 1996.
- [32] S. Martello, D. Pisinger, D. Vigo, R. V. A. N. Slyke, and Y. I. Young, “The Three-Dimensional Bin Packing Problem,” *Oper. Res.*, vol. 48, no. 2, pp. 256–267, 2000.
- [33] T. G. Crainic, G. Perboli, and R. Tadei, “Extreme point-based heuristics for three-dimensional bin packing,” *INFORMS J. Comput.*, vol. 20, no. 3, pp. 368–384, 2008.
- [34] L. Wei, W. C. Oon, W. Zhu, and A. Lim, “A skyline heuristic for the 2D rectangular packing and strip packing problems,” *Eur. J. Oper. Res.*, vol. 215, no. 2, pp. 337–346, 2011.
- [35] J. A. Bennell, L. Soon Lee, and C. N. Potts, “A genetic algorithm for two-dimensional bin packing with due dates,” *Int. J. Prod. Econ.*, vol. 145, no. 2, pp. 547–560, 2013.
- [36] J. Cunha and T. Queiroz, “Malha de Pontos no Problema da Mochila Bidimensional Limitada,” in *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, 2016.

- [37] O. Nascimento, J. Cunha, and T. Queiroz, “Resolução Do Problema De Empacotamento Ortogonal com Diferentes Malhas e Restrições Reais,” *Rev. eletrônica Pesqui. Operacional para o Desenvolv.*, vol. 8, no. 3, pp. 236–264, 2016.
- [38] J. G. de A. Cunha and T. A. de Queiroz, “Estudo de Malhas utilizando Atualização de Itens para Problemas de Empacotamento Bidimensional,” in *XLIX Simpósio Brasileiro de Pesquisa Operacional, Blumenau-SC, 27 a 30 de Agosto de 2017.*, 2017.
- [39] P. C. Gilmore and R. E. Gomory, “The theory and computation of knapsack functions,” *Oper. Res.*, vol. 14, no. August 2015, pp. 1045–1074, 1966.
- [40] M. Russo, A. Sforza, and C. Sterle, “An improvement of the knapsack function based algorithm of Gilmore and Gomory for the unconstrained two-dimensional guillotine cutting problem,” *Int. J. Prod. Econ.*, vol. 145, no. 2, pp. 451–462, 2013.
- [41] M. Russo, A. Sforza, and C. Sterle, “An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems,” *Comput. Oper. Res.*, vol. 50, pp. 97–114, 2014.
- [42] M. Hifi and V. Zissimopoulos, “A recursive exact algorithm for weighted two-dimensional cutting,” *Eur. J. Oper. Res.*, vol. 91, no. 3, pp. 553–564, 1996.
- [43] J. Desler and S. Hakimi, “A graph-theoretic approach to a class of integer-programming problems,” *Oper. Res.*, vol. 17, no. 6, pp. 1017–1033, 1969.
- [44] F. Hitchcock, “The Distribution of a Product from Several Sources to Numerous Localities,” *J. Math. Phys.*, vol. 20, no. 1–4, pp. 224–230, 1941.
- [45] N. Christofides and E. Hadjiconstantinou, “An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts,” *Eur. J. Oper. Res.*, vol. 83, no. 1, pp. 21–38, 1995.
- [46] M. Hifi and V. Zissimopoulos, “Constrained two-Dimensional cutting: an improvement of Christofides and Whitlock’s exact algorithm,” *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 324–331, 1997.
- [47] M. L. Fisher, “The Lagrangian Relaxation Method for Solving Integer Programming Problems,” *Manage. Sci.*, vol. 50, no. 12 Supplement, pp. 1861–1871, 2004.
- [48] N. Z. Shor, “The Subgradient Method,” in *Newton Methods for Nonlinear Problems*, vol. 18, no. 7, Springer, 1985, pp. 22–47.
- [49] J. E. Beasley, “An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure,” *Oper. Res.*, vol. 36, no. 1, pp. 49–64, 1985.
- [50] S. Martello and D. Vigo, “Exact Solution of the Two-Dimensional Finite Bin Packing Problem,” *Manage. Sci.*, vol. 44, no. April 2015, pp. 388–399, 1998.
- [51] A. H. Land and A. G. Doig, “An Automatic Method of Solving Discrete Programming Problems,” *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.
- [52] N. Agin, “Optimum Seeking with Branch and Bound,” *Manage. Sci.*, vol. 13, no. 4, pp. B176–B185, 1966.

- [53] F. Clautiaux, J. Carlier, and A. Moukrim, “A new exact method for the two-dimensional orthogonal packing problem,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1196–1211, 2007.
- [54] S. P. Fekete and J. Schepers, “On more-dimensional packing I: Modeling,” ZPR Technical Report 97.288, 2000.
- [55] S. P. Fekete and J. Schepers, “On more-dimensional packing II: Bounds,” ZPR Technical Report 97.289, 2000.
- [56] S. P. Fekete and J. Schepers, “On more-dimensional packing III: Exact Algorithms,” ZPR Technical Report 97.290, 2000.
- [57] S. P. Fekete and J. Schepers, “A Combinatorial Characterization of Higher-Dimensional Orthogonal Packing,” *Math. Oper. Res.*, vol. 29, no. 2, pp. 353–368, 2004.
- [58] S. P. Fekete and J. Schepers, “A general framework for bounds for higher-dimensional orthogonal packing problems,” *Math. Methods Oper. Res.*, vol. 60, no. 2, pp. 311–329, 2004.
- [59] S. P. Fekete, J. Schepers, and J. C. van der Veen, “An Exact Algorithm for Higher-Dimensional Orthogonal Packing,” *Oper. Res.*, vol. 55, no. 3, pp. 569–587, 2007.
- [60] E. Ferreira and J. F. Oliveira, “A note on Fekete and Schepers’ algorithm for the non-guillotinable two-dimensional packing problem.” Technical report, FEUP, pp. 2–5, 2005.
- [61] E. P. Ferreira and J. F. Oliveira, “Fekete and Schepers’ graph-based algorithm for the two-dimensional orthogonal packing problem revisited,” in *Intelligent Decision Support - Current Challenges and Approaches*, 2008, pp. 15–31.
- [62] M. A. Boschetti and A. Mingozzi, “The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case,” *Q. J. Belgian, French Ital. Oper. Res. Soc.*, vol. 1, no. 1, pp. 27–42, 2003.
- [63] R. J. Silveira and R. Morabito, “Um método heurístico baseado em programação dinâmica para o problema de corte bidimensional guilhotinado restrito,” *Gestão & Produção*, vol. 9, no. 1, pp. 78–92, 2002.
- [64] R. Baldacci and M. A. Boschetti, “A cutting-plane approach for the two-dimensional orthogonal non-guillotine cutting problem,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1136–1149, 2007.
- [65] D. Pisinger and M. Sigurd, “Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem,” *INFORMS J. Comput.*, vol. 19, no. 1, pp. 36–51, 2007.
- [66] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance, “Branch-and-price: Column generation for solving huge integer programs,” *Oper. Res.*, vol. 46, no. 3, pp. 316–329, 1998.
- [67] J. Puchinger and G. R. Raidl, “Models and algorithms for three-stage two-dimensional bin packing,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1304–1327, 2007.
- [68] M. Mrad, I. Meftahi, and M. Haouari, “A branch-and-price algorithm for the two-stage

- guillotine cutting stock problem," *J. Oper. Res. Soc.*, vol. 64, no. 5, pp. 629–637, 2013.
- [69] F. Clautiaux, A. Jouglet, J. Carlier, and A. Moukrim, "A new constraint programming approach for the orthogonal packing problem," *Comput. Oper. Res.*, vol. 35, no. 3, pp. 944–959, 2008.
- [70] P. Baptiste, C. Le Pape, and W. Nuijten, *Constraint-based scheduling : applying constraint programming to scheduling problems*. 2001.
- [71] M. A. Boschetti, A. Mingozzi, and E. Hadjiconstantinou, "New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem," *IMA J. Manag. Math.*, vol. 13, no. 2, pp. 95–119, 2002.
- [72] A. Lodi and M. Monaci, "Integer linear programming models for 2-staged two-dimensional Knapsack problems," *Math. Program.*, vol. 94, no. 2–3, pp. 257–278, 2003.
- [73] F. Furini, E. Malaguti, and D. Thomopoulos, "Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming," *INFORMS J. Comput.*, vol. 28, no. 4, pp. 736–751, 2016.
- [74] J. M. V. de Carvalho, "Exact solution of bin-packing problems using column generation and branch-and-bound," *Ann. Oper. Res.*, vol. 86, pp. 629–659, 1999.
- [75] R. Macedo, C. Alves, and J. M. V. de Carvalho, "Arc-flow model for the two-dimensional guillotine cutting stock problem," *Comput. Oper. Res.*, vol. 37, no. 6, pp. 991–1001, 2010.
- [76] L. A. Wolsey, "Valid Inequalities, Covering Problems and Discrete Dynamic Programs," *Ann. Discret. Math.*, vol. 1, pp. 527–538, 1977.
- [77] F. Brandão and J. P. Pedroso, "Bin packing and related problems: General arc-flow formulation with graph compression," *Comput. Oper. Res.*, vol. 69, pp. 56–67, 2016.
- [78] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation Algorithms for Bin-Packing — An Updated Survey," in *Journal of Mechanical Design*, vol. 130, no. 3, G. Ausiello, M. Lucertini, and P. Serafini, Eds. Vienna: Springer, 1984, pp. 49–106.
- [79] P. De Cani, "A Note on the Two-Dimensional Rectangular Cutting-Stock Problem," *J. Oper. Res. Soc.*, vol. 29, no. 7, p. 703, 1978.
- [80] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest, "Orthogonal Packings in Two Dimensions," *SIAM J. Comput.*, vol. 9, no. 4, pp. 846–855, 1980.
- [81] F. R. K. Chung, M. R. Garey, and D. S. Johnson, "On Packing Two-Dimensional Bins," *SIAM J. Algebr. Discret. Methods*, vol. 3, no. 1, pp. 66–76, 1982.
- [82] D. S. Johnson, "Near-optimal bin packing algorithms," Massachusetts Institute of Technology, 1973.
- [83] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms," *SIAM J. Comput.*, vol. 3, no. 4, pp. 299–325, 1974.
- [84] S. Jakobs, "On genetic algorithms for the packing of polygons," *Eur. J. Oper. Res.*, vol.

88, no. 1, pp. 165–181, 1996.

- [85] E. Hopper and B. C. H. Turton, “Genetic algorithm for a 2D industrial packing problem,” *Comput. Ind. Eng.*, vol. 37, no. 1, pp. 375–378, 1999.
- [86] D. Liu and H. Teng, “An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles,” *Eur. J. Oper. Res.*, vol. 112, no. 2, pp. 413–420, 1999.
- [87] A. Lodi, S. Martello, and D. Vigo, “Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems,” *INFORMS J. Comput.*, vol. 11, no. 4, pp. 345–357, 1999.
- [88] V. Beraudo, H. Alfonso, G. Minetti, and C. Salto, “Constrained Two-Dimensional Non-Guillotine Cutting Problem: an Evolutionary Approach,” in *XXIV International Conference of the Chilean Computer Science Society*, 2004, pp. 84–89.
- [89] E. K. Burke, G. Kendall, and G. Whitwell, “A New Placement Heuristic for the Orthogonal Stock-Cutting Problem,” *Oper. Res.*, vol. 52, no. 4, pp. 655–671, 2004.
- [90] J. F. Gonçalves and M. G. C. Resende, “A hybrid heuristic for the constrained two-dimensional non-guillotine orthogonal cutting problem,” 2006.
- [91] K. J. Binkley and M. Hagiwara, “Applying self-adaptive evolutionary algorithms to two-dimensional packing problems using a four corners’ heuristic,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1230–1248, 2007.
- [92] L. Wei, D. Zhang, and Q. Chen, “A least wasted first heuristic algorithm for the rectangular packing problem,” *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1608–1614, 2009.
- [93] J. F. Côté and M. Iori, “The meet-in-the-middle principle for cutting and packing problems,” *INFORMS J. Comput.*, vol. 30, no. 4, pp. 646–661, 2018.
- [94] B. E. E. Bengtsson, “Packing rectangular pieces—a heuristic approach,” *Comput. J.*, vol. 25, no. 3, pp. 253–257, 1982.
- [95] P. Y. Wang, “Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems,” *Oper. Res.*, vol. 31, no. 3, pp. 573–586, 1983.
- [96] V. Parada, R. Palma, and D. Sales, “A comparative numerical analysis for the guillotine two-dimensional cutting problem,” *Ann. Oper. Res.*, vol. 96, pp. 245–254, 2000.
- [97] J. F. Oliveira and J. S. Ferreira, “An improved version of Wang’s algorithm for two-dimensional cutting problems,” *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 256–266, 1990.
- [98] V. Parada, A. Gómez de Alvarenga, and J. de Diego, “Exact solutions for constrained two-dimensional cutting problems,” *Eur. J. Oper. Res.*, vol. 84, no. 3, pp. 633–644, 1995.
- [99] V. Parada, R. Muñoz, and A. G. de Alvarenga, “A Hybrid Genetic Algorithm for the Two-Dimensional Guillotine Cutting Problem,” in *Evolutionary Algorithms in Management Applications*, J. Biethahn and V. Nissen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 183–196.
- [100] V. Parada, M. Sepúlveda, M. Solar, and A. Gómez, “Solution for the constrained guillotine cutting problem by simulated annealing,” *Comput. Oper. Res.*, vol. 25, no. 1,

pp. 37–47, 1998.

- [101] J. E. Beasley, “Algorithms for Unconstrained Two-Dimensional Guillotine Cutting,” *J. Oper. Res. Soc.*, vol. 36, no. 4, pp. 297–306, 1985.
- [102] R. Morabito and M. N. Arenales, “Performance Of Two Heuristics For Solving Large Scale Two-Dimensional Guillotine Cutting Problems,” *INFOR Inf. Syst. Oper. Res.*, vol. 33, no. 2, pp. 145–155, 1995.
- [103] J. O. Berkey and P. Y. Wang, “Two-Dimensional Finite Bin-Packing Algorithms,” *J. Oper. Res. Soc.*, vol. 38, no. 5, p. 423, 1987.
- [104] A. Lodi, S. Martello, and D. Vigo, “Neighborhood Search Algorithm for the Guillotine Non-Oriented Two-Dimensional Bin Packing Problem,” in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Boston, MA: Springer US, 1999, pp. 125–139.
- [105] M. A. Boschetti and A. Mingozzi, “The Two-Dimensional Finite Bin Packing Problem. Part II: New lower and upper bounds,” *Q. J. Belgian, French Ital. Oper. Res. Soc.*, vol. 1, no. 2, pp. 135–147, 2003.
- [106] M. Hifi and R. M’Hallah, “Strip generation algorithms for constrained two-dimensional two-staged cutting problems,” *Eur. J. Oper. Res.*, vol. 172, no. 2, pp. 515–527, 2006.
- [107] M. Monaci and P. Toth, “A Set-Covering-Based Heuristic Approach for Bin-Packaging Problems,” *Informs J. Comput.*, vol. 18, no. 1, pp. 71–85, 2006.
- [108] A. Caprara, P. Toth, and M. Fischetti, “Algorithms for the Set Covering Problem,” *Ann. Oper. Res.*, vol. 98, no. 1, pp. 353–371, 2000.
- [109] E. Hadjiconstantinou and M. Iori, “A hybrid genetic algorithm for the two-dimensional single large object placement problem,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1150–1166, 2007.
- [110] D. G. Cattrysse and L. N. Van Wassenhove, “A survey of algorithms for the generalized assignment problem,” *Eur. J. Oper. Res.*, vol. 60, no. 3, pp. 260–272, 1992.
- [111] W. Huang and D. Chen, “An efficient heuristic algorithm for rectangle-packing problem,” *Simul. Model. Pract. Theory*, vol. 15, no. 10, pp. 1356–1365, 2007.
- [112] R. Morabito and V. Pureza, “A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem,” *Ann. Oper. Res.*, vol. 179, no. 1990, pp. 1–25, 2010.
- [113] Y. Cui, L. Yang, Z. Zhao, T. Tang, and M. Yin, “Sequential grouping heuristic for the two-dimensional cutting stock problem with pattern reduction,” *Int. J. Prod. Econ.*, vol. 144, no. 2, pp. 432–439, 2013.
- [114] G. Belov and G. Scheithauer, “Setup and Open-Stacks Minimization in One-Dimensional Stock Cutting,” *INFORMS J. Comput.*, vol. 19, no. 1, pp. 27–35, 2007.
- [115] Y. Cui, “Simple block patterns for the two-dimensional cutting problem,” *Math. Comput. Model.*, vol. 45, pp. 943–953, 2007.

- [116] K. Fleszar, "Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 463–474, 2013.
- [117] F. Alvelos, E. M. da C. Silva, and J. M. V. De Carvalho, "A hybrid heuristic based on column generation for two- and three- stage bin packing problems," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8580 LNCS, no. PART 2, pp. 211–226, 2014.
- [118] F. Alvelos, A. de Sousa, and D. Santos, "SearchCol: Metaheuristic Search by Column Generation," in *Hybrid Metaheuristics - Volume 6373 of the series Lecture Notes in Computer Science*, 2010, pp. 190–205.
- [119] F. Alvelos, A. de Sousa, and D. Santos, "Combining Column Generation and Metaheuristics," in *Hybrid Metaheuristics*, E.-G. Talbi, Ed. Springer Berlin Heidelberg, 2013, pp. 285–334.
- [120] Y.-P. Cui, Y. Cui, and T. Tang, "Sequential heuristic for the two-dimensional bin-packing problem," *Eur. J. Oper. Res.*, vol. 240, no. 1, pp. 43–53, 2015.
- [121] Y. Wang and L. Chen, "Two-dimensional residual-space-maximized packing," *Expert Syst. Appl.*, vol. 42, no. 7, pp. 3297–3305, 2015.
- [122] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA: Addison-Wesley Longman Publishing Co., Inc., 1989.
- [123] J. F. Gonçalves, "A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1212–1229, 2007.
- [124] J. F. Gonçalves and M. G. C. Resende, "A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem," *J. Comb. Optim.*, vol. 22, no. 2, pp. 180–201, 2011.
- [125] A. Bortfeldt and T. Winter, "A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces," *Int. Trans. Oper. Res.*, vol. 16, no. 6, pp. 685–713, 2009.
- [126] L. S. Lee, "Multicrossover Genetic Algorithms for Combinatorial Optimisation," University of Southampton, UK, 2006.
- [127] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598. pp. 671–680, 1983.
- [128] L. Faina, "An application of simulated annealing to the cutting stock problem," *Eur. J. Oper. Res.*, vol. 114, no. 3, pp. 542–556, 1999.
- [129] J. Egeblad and D. Pisinger, "Heuristic approaches for the two- and three-dimensional knapsack packing problem," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1026–1049, 2009.
- [130] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 15, no. 12, pp. 1518–1524, 1996.
- [131] E. Hopper and B. C. H. Turton, "An empirical investigation of meta-heuristic and

heuristic algorithms for a 2D packing problem," *Eur. J. Oper. Res.*, vol. 128, no. 1, pp. 34–57, 2001.

- [132] T. W. Leung, C. H. Yung, and M. D. Troutt, "Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem," *Comput. Ind. Eng.*, vol. 40, no. 3, pp. 201–214, 2001.
- [133] T. W. Leung, C. K. Chan, and M. D. Troutt, "Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem," *Eur. J. Oper. Res.*, vol. 145, no. 3, pp. 530–542, 2003.
- [134] J. E. Beasley, "A population heuristic for constrained two-dimensional non-guillotine cutting," *Eur. J. Oper. Res.*, vol. 156, no. 3, pp. 601–627, 2004.
- [135] J. E. Beasley, "Population heuristics," in *Handbook of applied optimization*, O. U. Press, Ed. Oxford: Oxford University Press, 2002, pp. 138–157.
- [136] V. Beraudo, A. Orellana, H. Alfonso, G. Minetti, and C. Salto, "Solving the Two Dimensional Cutting Problem using Evolutionary Algorithms with Penalty Functions," *Workshop de Agentes y Sistemas Inteligentes (WASI)*. 2005.
- [137] I. Kierkosz and M. Luczak, "A hybrid evolutionary algorithm for the two-dimensional packing problem," *Cent. Eur. J. Oper. Res.*, vol. 22, no. 4, pp. 729–753, 2014.
- [138] A. Lodi, S. Martello, and D. Vigo, "Approximation algorithms for the oriented two-dimensional bin packing problem," *Eur. J. Oper. Res.*, vol. 112, no. 1, pp. 158–166, 1999.
- [139] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.
- [140] R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit, "A tabu search algorithm for a two-dimensional non-guillotine cutting problem," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1167–1182, 2007.
- [141] R. Alvarez-Valdés, A. Parajón, and J. M. Tamarit, "A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems," *Comput. Oper. Res.*, vol. 29, no. 7, pp. 925–947, 2002.
- [142] T. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *J. Glob. Optim.*, pp. 109–133, 1995.
- [143] F. Glover, "A template for scatter search and path relinking," *Artif. Evol.*, vol. 1363, no. February 1998, pp. 3–51, 1998.
- [144] R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit, "A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems," *J. Oper. Res. Soc.*, vol. 56, no. 4, pp. 414–425, 2005.
- [145] R. Alvarez-Valdés, R. Martí, J. M. Tamarit, and A. Parajón, "GRASP and path relinking for the two-dimensional two-stage cutting-stock problem," *INFORMS J. Comput.*, vol. 19, no. 2, pp. 261–272, 2007.
- [146] F. Parreño, R. Alvarez-Valdés, J. F. Oliveira, and J. M. Tamarit, "A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing," *Ann. Oper. Res.*, vol. 179, no.

1, pp. 203–220, 2010.

- [147] P. Hansen and N. Mladenović, “Variable neighborhood search,” *Search Methodol. Introd. Tutorials Optim. Decis. Support Tech.*, pp. 211–238, 2005.
- [148] F. Alvelos, T. M. Chan, P. Vilaca, T. Gomes, E. M. da C. Silva, and J. M. V. de Carvalho, “Sequence based heuristics for two-dimensional bin packing problems,” *Eng. Optim.*, vol. 41, no. 8, pp. 773–791, 2009.
- [149] T. M. Chan, F. Alvelos, E. M. da C. Silva, and J. M. V. de Carvalho, “A combined local search approach for the two-dimensional bin packing problem,” in *Proceedings of the EU/MEEeting 2009 European Chapter on Metaheuristics Workshop*, 2009, pp. 153–158.
- [150] T. M. Chan, F. Alvelos, E. M. da C. Silva, and J. M. V. De Carvalho, “Heuristics with stochastic neighborhood structures for two-dimensional bin packing and cutting stock problems,” *Asia-Pacific J. Oper. Res.*, vol. 28, no. 2, pp. 255–278, 2011.
- [151] N. Mladenović and P. Hansen, “Variable neighborhood search,” *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
- [152] F. Dusberger and G. R. Raidl, “A Variable Neighborhood Search Using Very Large Neighborhood Structures for the 3-Staged 2-Dimensional Cutting Stock Problem,” in *International Workshop on Hybrid Metaheuristics*, 2014, pp. 85–99.
- [153] A. Fritsch and O. Vornberger, “Cutting Stock by Iterated Matching,” in *Operations Research Proceedings 1994*, Springer Berlin Heidelberg, 1995, pp. 92–97.
- [154] J. Puchinger, G. R. Raidl, and G. Koller, “Solving a Real-World Glass Cutting Problem,” in *Evolutionary Computation in Combinatorial Optimization: 4th European Conference, EvoCOP 2004, Coimbra, Portugal, April 5-7, 2004. Proceedings*, vol. 3004, 2004, pp. 165–176.
- [155] C. Voudouris and E. Tsang, “Guided local search and its application to the traveling salesman problem,” *Oper. Res.*, vol. 113, pp. 469–499, 1999.
- [156] O. Faroe, D. Pisinger, and M. Zachariasen, “Guided Local Search for the Three-Dimensional Bin-Packing Problem,” *INFORMS J. Comput.*, vol. 15, no. 3, pp. 267–283, 2003.
- [157] J. Q. Jiang, Y. C. Liang, X. H. Shi, and H. P. Lee, “A Hybrid Algorithm Based on PSO and SA and Its Application for Two-Dimensional Non-guillotine Cutting Stock Problem,” in *Lecture Notes in Computer Science*, vol. 3037, no. ICCS 2004, IEEE, 2004, pp. 666–669.
- [158] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948, 1995.
- [159] L. Bao, J. Jiang, C. Song, L. Zhao, and J. Gao, “Artificial Fish Swarm Algorithm for Two-Dimensional Non-Guillotine Cutting Stock Problem,” in *Advances in Neural Networks – ISNN 2013: 10th International Symposium on Neural Networks, Dalian, China, July 4-6, 2013, Proceedings, Part II 2013*, 2013, pp. 552–559.
- [160] X. Li, Z. Shao, and J. Qian, “An optimizing method based on autonomous animats: fish-swarm algorithm,” *Syst. Eng. Theory Pract.*, vol. 22, no. 11, pp. 32–38, 2002.

- [161] M. K. Omar and K. Ramakrishnan, "Solving non-oriented two dimensional bin packing problem using evolutionary particle swarm optimisation," *Int. J. Prod. Res.*, vol. 51, no. 20, pp. 6002–6016, 2013.
- [162] P. S. Ow and T. E. Morton, "Filtered beam search in scheduling," *Int. J. Prod. Res.*, vol. 26, no. 1, pp. 35–62, 1988.
- [163] M. Hifi, R. M'Hallah, and T. Saadi, "Algorithms for the constrained two-staged two-dimensional cutting problem," *INFORMS J. Comput.*, vol. 20, no. 2, pp. 212–221, 2008.
- [164] M. Hifi, S. Negre, R. Ouafi, and T. Saadi, "A parallel algorithm for constrained two-staged two-dimensional cutting problems," *Comput. Ind. Eng.*, vol. 62, no. 1, pp. 177–189, 2012.
- [165] B. L. Golden, "Approaches to the Cutting Stock Problem," *A I I E Trans.*, vol. 8, no. 2, pp. 265–274, 1976.
- [166] A. I. Hinxman, "Trim-Loss and Assortment Problems - a Survey.," *Eur. J. Oper. Res.*, vol. 5, no. 1, pp. 8–18, 1980.
- [167] K. A. Dowsland and W. B. Dowsland, "Packing problems," *Eur. J. Oper. Res.*, vol. 56, no. 1, pp. 2–14, 1992.
- [168] H. Dyckhoff and U. Finke, *Cutting and Packing in Production and Distribution: a typology and bibliography*. 1992.
- [169] Y. Lirov, "Knowledge based approach to the cutting stock problem," *Math. Comput. Model.*, vol. 16, no. 1, pp. 107–125, 1992.
- [170] C. Cheng, B. Feiring, and T. Cheng, "The cutting stock problem — a survey," *Int. J. Prod. Econ.*, vol. 36, no. 3, pp. 291–305, 1994.
- [171] E. Hopper and B. C. H. Turton, "A Review of the Application of Meta-Heuristic Algorithms to 2D Strip Packing Problems," *Artif. Intell. Rev.*, vol. 16, no. 4, pp. 257–300, 2001.
- [172] A. Lodi, S. Martello, and D. Vigo, "Recent advances on two-dimensional bin packing problems," *Discret. Appl. Math.*, vol. 123, no. 1–3, pp. 379–396, 2002.
- [173] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *Eur. J. Oper. Res.*, vol. 141, no. 2, pp. 241–252, 2002.
- [174] J. Jylänki, "A thousand ways to pack the bin-a practical approach to two-dimensional rectangle bin packing," pp. 1–50, 2010.
- [175] T. Chan, F. Alvelos, E. M. da C. Silva, and J. Valériode Carvalho, "Heuristics for Two-Dimensional Bin-Packing Problems," in *The Industrial Electronics Handbook*, 2nd ed., J. D. I. Bogdan M. Wilamowski, Ed. CRC Press, 2011, pp. 1–18.
- [176] T. G. Crainic, G. Perboli, and R. Tadei, "Recent Advances in Multi-Dimensional Packing Problems," *New Technol. - Trends, Innov. Res.*, pp. 91–110, 2012.
- [177] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *Eur. J. Oper. Res.*, vol. 255, no. 1, pp. 1–

20, 2016.

- [178] J. F. Oliveira, A. Neuenfeldt Júnior, E. M. da C. Silva, and M. A. Carraville, "A Survey on Heuristics for the Two-Dimensional Rectangular Strip Packing Problem," *Pesqui. Operacional*, vol. 36, no. 2, pp. 197–226, 2016.
- [179] E. Silva, J. F. Oliveira, and G. Wäscher, "The pallet loading problem: A review of solution methods and computational experiments," *Int. Trans. Oper. Res.*, vol. 23, no. 1–2, pp. 147–172, 2016.
- [180] C. A. Gil Gonzalez, J. P. Orejuela Cabrera, and D. Peña, "El Problema de patrones de corte, clasificación y enfoques/Cutting stock problem, classification and approaches," *Prospectiva*, vol. 15, no. 1, p. 112, 2017.
- [181] J. Côté and M. Iori, "The Meet-in-the-Middle Principle for Cutting and Packing Problems," CIRRELT-2016-28, 2016.
- [182] M. E. Lübbecke, "Column Generation," in *Wiley Encyclopedia of Operations Research and Management Science*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011.
- [183] N. Prabhu, "The Simplex Method and Its Complexity," *Wiley Encycl. Oper. Res. Manag. Sci.*, pp. 1–9, 2010.
- [184] A. A. Farley, "Selection of stockplate characteristics and cutting style for two dimensional cutting stock situations," *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 239–246, 1990.

Datasets and Generators for Two-dimensional Cutting and Packing Problems

Abstract We present an extensive survey of datasets and instance generators that are usually used by the researchers when dealing with two-dimensional rectangular cutting and packing problems. This paper seeks to help researchers to refer, find, and use these datasets and instances generators.

Keywords: Datasets, Generators, Two-dimensional, Cutting and Packing Problem

1. Introduction

We present the datasets and the instance generators that have been consistently used in cutting and packing research of new optimization methods. Some of these datasets have been used by researchers, maintaining the main characteristics of the instances (e.g., object and item sizes) or adapting (neglecting, changing or adding) some of their characteristics to correspond to their needs (e.g., neglecting the items' value [1], generating randomly demand [2], adding objects [2], among others).

The motivation for this work is to ease future references to the datasets and provide to researchers a means to find those datasets as easily and fast as possible. Since some internet links can become inactive, we have a copy of all these datasets and instances generators that we will provide upon request¹.

The rest of the paper is organized as follows. In Section 2, the datasets are characterized considering their most relevant features. In Section 3, the instances generators for two-dimensional rectangular cutting and packing problems are presented.

¹ They can also be obtained through our website at <https://oscar-oliveira.github.io/2D-Cutting-and-Packing/>.

2. Literature Benchmarks

In this section, 84 literature datasets are characterized considering their most relevant features. We tried to maintain as much as possible the datasets names for which they are best-known and usually referred to in the literature. For the others, we have decided to name the dataset with an acronym formed by the first letter of the authors names, e.g., the set defined in Adamowicz and Albano [3] was named as AA.

In Section 2.1, for each dataset alphabetically sorted by name, Table 1 gives the dataset name, the article(s) in which it was presented, the publication year, and the problems types considered in the article(s) following the typology proposed by Wäscher *et al.* [3].

In Section 2.2, Table 2 to Table 85 present, for each dataset, the name, the number of instances in the set (#), a link (if one exists) to an internet source for download, and the objects and items characteristics. The objects are characterized by the number of object types (k), the range of objects length and height (LxH), the range of available stock (e) and the range of object cost (Cost). The items are characterized by the number of item types (m), the range of items length and height (lxh), the range of items demand (d), and the range of items value (value).

2.1. Datasets

Table 1. Datasets for cutting and packing problems.

Dataset	Article	Year	Problem					
			IIPP	PP	KP	ODP	CSP	BPP
AA	Adamowicz and Albano [4]	1976					•	
AB	Cui and Huang [5]	2012		•				
ABM	Andrade <i>et al.</i> [6]	2016					•	
ABMR	Andrade <i>et al.</i> [7]	2016					•	
AH	Bortfeldt and Gehring [8]	2006				•		
ASSORT	Beasley [9]	1985					•	
ATP	Alvarez-Valdés <i>et al.</i> [10]	2002		•				
B	Cui <i>et al.</i> [11]	2005			•			
BABU	Babu and Babu [12]	1999			•			
BABU2	Babu and Babu [13]	2001			•			
BENG	Bengtsson [14]	1982				•		•
BKW	Burke <i>et al.</i> [15]	2004				•		
BRPB	El-Bouri <i>et al.</i> [16]	2006					•	
CGCUT	Christofides and Whitlock [17]	1977		•				

Dataset	Article	Year	Problem					
			IIPP	PP	KP	ODP	CSP	BPP
CH	Cui and Huang [18]	2012					•	
CHL	Cung et al. [19]	2000		•				
CJCM	Clautiaux et al. [20]	2008			•			
CLASS	[01-06] Berkey and Wang [21]	1987						•
	[07-10] Martello and Vigo [22]	1998						•
CMWX	Cintra et al. [2]	2008		•		•	•	
CUI	Cui [23]	2008		•				
CWL	Cui et al. [11]	2005			•			
CY	Cui and Yang [24]	2011		•				
CZ	Cui and Zhao [25]	2013					•	
D	[1, 2] Ratanapan and Dagli [26]	1997					•	
	3 Ratanapan and Dagli [27]	1998					•	
	4 Dagli and Poshyanonda [28]	1997					•	
DOWSLAND	Dowsland [29]	1984	•					
EL-AAL	El-Aal [30]	1994						•
EP2	Egeblad and Pisinger [31]	2009			•			
FHZ	Fayard et al. [32]	1998		•				
FO	Ferreira and Oliveira [33]	2005			•			
GARD	Gardner [34]	1966			•			
GCUT	Beasley [35]	1985		•				
HADCHR	Hadjiconstantinou and Christofides [36]	1995			•			
HERZ	Herz [37]	1972		•				
HIFI1997a	Hifi [38]	1997		•				
HIFI1997b	Hifi [39]	1997		•				
HIFI2001	Hifi [40]	2001		•				
HOPPER	Hopper [41]	2000					•	•
HT2001a	Hopper and Turton [42]	2001						•
HT2001b	Hopper and Turton [43]	2001					•	
HZ1	Hifi and Zissimopoulos [44]	1996		•				
HZ2	Hifi and Zissimopoulos [45]	1996			•			
IS	Israni and Sanders [46]	1982						•
IYUAI	Imahori et al. [47]	2005						•
JAKOBS	Jakobs [48]	1996					•	
JLSL	Jiang et al. [49]	2004		•				
KORF	Korf et al. [50]	2010					•	
KR	Kröger [51]	1995					•	
LC	Lai and Chan [52]	1997			•			
LCT	Leung et al. [53]	2003			•			
LYT	Leung et al. [54]	2001			•			
MA	Morabito and Arenales [55]	2000						•
MAA	Morabito et al. [56]	1992		•				
MB	Pisinger and Sigurd [57]	2005						•
MB2D	Mack and Bortfeldt [58]	2012					•	•
MG	Morabito and Garcia [59]	1998					•	
MP	Morabito and Pureza [60]	2010		•				
MWV	Mumford-Valenzuela et al. [61]	2004					•	

Dataset	Article	Year	Problem					
			IIPP	PP	KP	ODP	CSP	BPP
NGCUT	Beasley [62]	1985		•			•	
NGCUTAP	Beasley [63]	2004		•				
NGCUTCON	Beasley [63]	2004		•				
NGCUTFS	Beasley [63]	2004		•				
NHU	Novianingsih et al. [64]	2012					•	
OF	Oliveira and Ferreira [65]	1990		•				
OKP	Fekete and Schepers [66]	2000			•			
ONV	Ortmann et al. [67]	2010				•		•
PGD	Parada et al. [68]	1995		•				
PO	Pinto and Oliveira [69]	2005			•			
RAND	Martello and Monaci [70]	2015				•		
RSS	Russo et al. [71]	2014		•				
SCP	Hifi [72]	1998				•		
SCPL	Hifi [73]	1999				•		
SPEKSMA	Spieksma [74]	1994						•
	Caprara and Toth [75]	2001						•
SS	Skalbeck and Schultz [76]	1976					•	
SSOYKI	Shiomii et al. [77]	2007				•		
STS	Tschöke and Holthöfer [78]	1995		•				
TEST	Yanasse et al. [79]	1991					•	
VAG	Vianna et al. [80]	2003		•				
VASSILIADIS	Vassiliadis [81]	2005			•			
VENKATESWARLU	Venkateswarlu [82]	2001		•				
WANG	Wang [83]	1983		•				•
WV	Wang and Valenzuela [84]	2001			•			
WVINT	Wei et al. [85]	2011		•		•		
WWD	Wan et al. [86]	2005		•				
ZDF	ZDF1-9	Zhang et al. [87]	2013				•	
	ZDF10-16	Leung and Zhang [88]	2011				•	

2.2. Characterization

Table 2. Features of the instances in AA.

Name	AA								
#	Objects				Items				
2	k	1	e	-	m	15	d	[8-96]	
	LxH	[12030-36090]×2550			l×h	[270-2500]×[74-785]			
	Cost	-		Value	-				

Table 3. Features of the instances in AB.

Name	AB								
#	Objects				Items				
60	k	1	e	-	m	[50-150]	d	[1-50]	
	LxH	[2003-2997]×[1014-1474]			l×h	[50-499]×[50-499]			
	Cost	-		Value	[914-171432]				

Table 4. Features of the instances in ABM.

Name	ABM							
#	Objects				Items			
20	k	[1-4]	e	[1-8]	m	[1-37]	d	[1-17]
	LxH	[10-290]×[10-183]			Ixh	[1-114]×[1-59]		
	Cost	-		Value	-			

Table 5. Features of the instances in ABMR.

Name	ABMR							
#	Objects				Items			
10	k	[2-3]	e	[1-2]	m	[1-3]	d	[1-7]
	LxH	[13-29]×[10-30]			Ixh	[1-11]×[1-11]		
	Cost	-		Value	-			

Table 6. Features of the instances in AH.

Name	AH									
#	Objects				Items					
360	k	1	e	-	m	1000	d	-		
	LxH	1000]×[1257-56390]			Ixh	[6-375]×[6-375]				
	Cost	-		Value	-					
URL	http://www.computational-logistics.org/orlib/topic/2D%20Strip%20Packing/									

Table 7. Features of the instances in ASSORT.

Name	ASSORT									
#	Objects				Items					
12	k	10	e	-	m	[10-30]	d	[20-[20-39]]		
	LxH	[50-248]×[51-248]			Ixh	[25-124]×[25-124]				
	Cost	-		Value	-					
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/assortinfo.html									

Table 8. Features of the instances in ATP.

Name	ATP									
#	Objects				Items					
10	k	1	e	-	m	[31-59]	d	-		
	LxH	[1674-2899]×[1612-2994]			Ixh	[101-1117]×[81-1192]				
	Cost	-		Value	-					
10	k	1	e	-	m	[36-59]	d	-		
	LxH	[1793-2885]×[1656-2858]			Ixh	[96-1142]×[82-1143]				
	Cost	-		Value	[7963-576621]					
10	k	1	e	-	m	[27-56]	d	[1-9]		
	LxH	[241-960]×[124-983]			Ixh	[15-363]×[6-390]				
	Cost	-		Value	-					
10	k	1	e	-	m	[25-59]	d	[1-9]		
	LxH	[167-931]×[138-917]			Ixh	[8-355]×[6-362]				
	Cost	-		Value	[68-27446]					
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/									

Table 9. Features of the instances in B.

Name	B									
#	Objects				Items					
7	k	1	e	-	m	[30-180]	d	-		
	LxH	4000×2000			Ixh	[200-695]×[200-698]				
	Cost	-		Value	-					
URL	http://lagrange.ime.usp.br/~lobato/utdc/instances.php									

Table 10. Features of the instances in BABU.

Name	BABU							
#	Objects				Items			
1	k	1	e	-	m	14	d	[1-14]
	LxH	1000×375			Ixh	[15-175]×[50-150]		
	Cost	-			Value	-		
1	k	[1-5]	e	[2-10]	m	14	d	[1-14]
	LxH	[100-600]×[400-650]			Ixh	[15-175]×[50-150]		
	Cost	-			Value	-		

Table 11. Features of the instances in BABU2.

Name	BABU2							
#	Objects				Items			
1	k	1	e	-	m	20	d	-
	LxH	80×200			Ixh	[5-35]×[5-35]		
	Cost	-			Value	-		

Table 12. Features of the instances in BENG.

Name	BENG							
#	Objects				Items			
10	k	1	e	-	m	[20-200]	d	-
	LxH	[25-40]×[10-25]			Ixh	[1-12]×[1-8]		
	Cost	-			Value	-		
URL	http://or.dei.unibo.it/library/orthogonal-stock-cutting-problems							

Table 13. Features of the instances in BKW.

Name	BKW							
#	Objects				Items			
13	k	1	e	-	m	[10-3152]	d	-
	LxH	[30-640]×[40-960]			Ixh	[1-74]×[1-125]		
	Cost	-			Value	-		
URL	http://or.dei.unibo.it/library/orthogonal-stock-cutting-problems							

Table 14. Features of the instances in BRPB.

Name	BRPB							
#	Objects				Items			
1	k	3	e	-	m	5	d	[1-9]
	LxH	[130-149]×[44-51]			Ixh	[19-45]×[10-29]		
	Cost	-			Value	-		
URL	https://paginas.fe.up.pt/~esicup/datasets							

Table 15. Features of the instances in CGCUT.

Name	CGCUT							
#	Objects				Items			
3	k	1	e	-	m	[7-20]	d	[1-5]
	LxH	[15-40]×[10-70]			Ixh	[2-33]×[1-43]		
	Cost	-			Value	[2-582]		
URL	https://paginas.fe.up.pt/~esicup/datasets							

Table 16. Features of the instances in CH.

Name	CH							
#	Objects				Items			
80	k	1	e	-	m	[5-40]	d	[102-9993]
	LxH	[2000-2986]×[1000-1498]			Ixh	[5-999]×[50-999]		
	Cost	-			Value	-		

Table 17. Features of the instances in CHL.

Name	CHL							
#	Objects				Items			
7	k	1	e	-	m	[10-35]	d	[1-5]
	LxH	[62-207]×[55-231]			l×h	[7-69]×[7-63]		
	Cost	-			Value	[100-1523]		
9	k	1	e	-	m	[10-40]	d	[1-8]
	LxH	[20-263]×[20-244]			l×h	[1-109]×[2-135]		
	Cost	-			Value	-		
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 18. Features of the instances in CJCM.

Name	CJCM							
#	Objects				Items			
42	k	1	e	-	m	[10-19]	d	[1-4]
	LxH	20×20			l×h	[1-20]×[2-20]		
	Cost	-			Value	[2-168]		
URL	https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/SoftwareAlgoKP							

Table 19. Features of the instances in CLASS.

Name	CLASS							
#	Objects				Items			
500	k	1	e	-	m	[20-100]	d	-
	LxH	[10-300]×[10-300]			l×h	[1-100]×[1-100]		
	Cost	-			Value	-		
URL	http://or.dei.unibo.it/library/two-dimensional-bin-packing-problem							

Table 20. Features of the instances in CMWX.

Name	CMWX							
#	Objects				Items			
4	k	1	e	-	m	[42-82]	d	-
	LxH	3500×3500			l×h	[254-970]×[116-1890]		
	Cost	-			Value	[50924-1081080]		

Table 21. Features of the instances in CUI.

Name	CUI							
#	Objects				Items			
21	k	1	e	-	m	[49-96]	d	[3-15]
	LxH	[1017-4518]×[1005-4323]			l×h	[60-1342]×[40-1282]		
	Cost	-			Value	[653-1006880]		

Table 22. Features of the instances in CWL.

Name	CWL							
#	Objects				Items			
50	k	1	e	-	m	30	d	-
	LxH	3000×1500			l×h	[200-699]×[200-699]		
	Cost	-			Value	-		

Table 23. Features of the instances in CY.

Name	CY							
#	Objects				Items			
300	k	1	e	-	m	[25-200]	d	[1-30]
	LxH	[2000-2950]×[1000-1450]			l×h	[50-700]×[50-700]		
	Cost	-			Value	[1491-462830]		

Table 24. Features of the instances in CZ.

Name	CZ							
#	Objects				Items			
5	k	1	e	-	m	58	d	[2-328]
	LxH	[2000-4200]x[1830-2900]			Ixh	[368-1749]x[474-1589]		
	Cost	-		Value	-			

Table 25. Features of the instances in D.

Name	D							
#	Objects				Items			
4	k	1	e	-	m	[4-8]	d	[2-9]
	LxH	[20-60]x-			Ixh	[4-15]x[4-13]		
	Cost	-		Value	-			
URL	https://paginas.fe.up.pt/~esicup/datasets							

Table 26. Features of the instances in DOWSLAND.

Name	DOWSLAND							
#	Objects				Items			
8	k	1	e	-	m	1	d	-
	LxH	[22-86]x[16-82]			Ixh	[5-15]x[3-11]		
	Cost	-		Value	-			

Table 27. Features of the instances in EL-AAL.

Name	EL-AAL							
#	Objects				Items			
2	k	1	e	-	m	[4-10]	d	[1-300]
	LxH	[20-2000]x[10-1000]			Ixh	[4-1000]x[2-645]		
	Cost	-		Value	-			

Table 28. Features of the instances in EP2.

Name	EP2							
#	Objects				Items			
80	k	1	e	-	m	[30-200]	d	1
	LxH	[33-739]x[65-1479]			Ixh	[1-100]x[1-100]		
	Cost	-		Value	[1-30000]			
URL	http://www.diku.dk/~pisinger/codes.html							

Table 29. Features of the instances in FHZ.

Name	FHZ							
#	Objects				Items			
11	k	1	e	-	m	[25-50]	d	[1-9]
	LxH	[100-977]x[125-985]			Ixh	[20-527]x[25-576]		
	Cost	-		Value	-			
11	k	1	e	-	m	[25-60]	d	[1-12]
	LxH	[125-992]x[105-970]			Ixh	[25-636]x[21-623]		
	Cost	-		Value	[140-999]			
11	k	1	e	-	m	[25-60]	d	-
	LxH	[500-3500]x[500-3765]			Ixh	[37-2254]x[101-2226]		
	Cost	-		Value	-			
11	k	1	e	-	m	[25-60]	d	-
	LxH	[500-3500]x[500-3650]			Ixh	[96-2245]x[98-2354]		
	Cost	-		Value	[110-748]			
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 30. Features of the instances in FO.

Name	FO							
#	Objects				Items			
1	k	1	e	-	m	5	d	-
	LxH	9x9		Ixh	[2-5]x[4-5]			
	Cost	-		Value	-			

Table 31. Features of the instances in GARD.

Name	GARD							
#	Objects				Items			
40	k	-	e	-	m	[1-40]	d	-
	LxH	-		Ixh	[1-40]x[1-40]			
	Cost	-		Value	-			
URL	http://or.dei.unibo.it/library/orthogonal-stock-cutting-problems							

Table 32. Features of the instances in GCUT.

Name	GCUT							
#	Objects				Items			
13	k	1	e	-	m	[10-50]	d	-
	LxH	[250-3000]x[250-3000]		Ixh	[62-970]x[63-1890]			
	Cost	-		Value	[4554-1081080]			
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gcutinfo.html							

Table 33. Features of the instances in HADCHR.

Name	HADCHR							
#	Objects				Items			
2	k	1	e	-	m	[7-15]	d	1
	LxH	30x30		Ixh	[1-22]x[4-21]			
	Cost	-		Value	[17-828]			

Table 34. Features of the instances in HERZ.

Name	HERZ							
#	Objects				Items			
1	k	1	e	-	m	5	d	-
	LxH	127x98		Ixh	[18-54]x[13-65]			
	Cost	-		Value	[273-1170]			
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 35. Features of the instances in HIFI1997a.

Name	HIFI1997a							
#	Objects				Items			
3	k	1	e	-	m	[10-20]	d	-
	LxH	[4500-7350]x[4070-6579]		Ixh	[232-2828]x[347-2647]			
	Cost	-		Value	-			
3	k	1	e	-	m	[20-40]	d	-
	LxH	[3427-7500]x[2769-7381]		Ixh	[437-5751]x[316-3787]			
	Cost	-		Value	[398-4351]			
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 36. Features of the instances in HIFI1997b.

Name	HIFI1997b							
#	Objects				Items			
3	k	1	e	-	m	[5-20]	d	[1-6]
	LxH	[50-127]x[60-98]			lxh	[9-54]x[11-65]		
	Cost	-			Value	[140-1170]		
3	k	1	e	-	m	20	d	[1-5]
	LxH	[70-132]x[70-100]			lxh	[9-69]x[11-63]		
	Cost	-			Value	-		
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 37. Features of the instances in HIFI2001.

Name	HIFI2001							
#	Objects				Items			
5	k	1	e	-	m	[40-200]	d	-
	LxH	[7350-45237]x[6579-35983]			lxh	[28-9098]x[80-8726]		
	Cost	-			Value	-		
11	k	1	e	-	m	[10-200]	d	-
	LxH	[100-45237]x[156-35983]			lxh	[16-9098]x[32-8726]		
	Cost	-			Value	[152-15877830]		
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 38. Features of the instances in HOPPER.

Name	HOPPER							
#	Objects				Items			
70	k	1	e	-	m	[17-199]	d	-
	LxH	200x200			lxh	[1-178]x[1-180]		
	Cost	-			Value	-		
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/stripinfo.html							

Table 39. Features of the instances in HT2001a.

Name	HT2001a							
#	Objects				Items			
21	k	1	e	-	m	[16-197]	d	-
	LxH	[20-160]x[15-240]			lxh	[1-72]x[1-113]		
	Cost	-			Value	-		
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpacktwoinfo.html							

Table 40. Features of the instances in HT2001b.

Name	HT2001b							
#	Objects				Items			
15	k	6	e	[2-4]	m	[100-150]	d	-
	LxH	[10-60]x[10-120]			lxh	[1-30]x[1-30]		
	Cost	-			Value	-		

Table 41. Features of the instances in Hz1.

Name	Hz1							
#	Objects				Items			
1	k	1	e	-	m	6	d	-
	LxH	78x67			lxh	[6-32]x[5-54]		
	Cost	-			Value	-		
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 42. Features of the instances in HZ2.

Name	HZ2							
#	Objects				Items			
1	k	1	e	-	m	5	d	-
	LxH	99×80		Ixh	[18-54]×[13-35]			
	Cost	-		Value	[273-1083]			

Table 43. Features of the instances in IS.

Name	IS							
#	Objects				Items			
2	k	1	e	-	m	20	d	[1-25]
	LxH	70×40		Ixh	[1-14]×[1-6]			
	Cost	-		Value	-			

Table 44. Features of the instances in IYUAI.

Name	IYUAI							
#	Objects				Items			
60	k	1	e	-	m	[20-50]	d	[1-200]
	LxH	[1400-4000]×[700-2000]		Ixh	[141-996]×[70-500]			
	Cost	-		Value	-			

Table 45. Features of the instances in JAKOBS.

Name	JAKOBS							
#	Objects				Items			
7	k	1	e	-	m	[20-50]	d	-
	LxH	[40-120]×[15-80]		Ixh	[2-40]×[2-36]			
	Cost	-		Value	-			

Table 46. Features of the instances in JSL.

Name	JSL							
#	Objects				Items			
5	k	1	e	-	m	[5-18]	d	[1-5]
	LxH	[40-100]×[20-80]		Ixh	[3-30]×[3-50]			
	Cost	-		Value	-			

Table 47. Features of the instances in KORF.

Name	KORF							
#	Objects				Items			
40	k	-	e	-	m	[1-40]	d	-
	LxH	-		Ixh	[1-40]×[2-41]			
	Cost	-		Value	-			
URL	http://or.dei.unibo.it/library/orthogonal-stock-cutting-problems							

Table 48. Features of the instances in KR.

Name	KR							
#	Objects				Items			
12	k	1	e	-	m	[25-60]	d	-
	LxH	100×[102-280]		Ixh	[1-40]×[1-40]			
	Cost	-		Value	-			
URL	http://www.computational-logistics.org/orlib/topic/2D%20Guillotine%20Strip%20Packing%20Problem/							

Table 49. Features of the instances in LC.

Name	LC							
#	Objects				Items			
3	k	1	e	-	m	[10-20]	d	-
	LxH	400×[200-400]		Ixh	[30-200]×[30-150]			-
	Cost	-		Value	-			-
5	k	1	e	-	m	[4-9]	d	[1-5]
	LxH	2325×1825		Ixh	[200-1350]×[200-1100]			-
	Cost	-		Value	-			-

Table 50. Features of the instances in LCT.

Name	LCT							
#	Objects				Items			
2	k	1	e	-	m	[40-50]	d	-
	LxH	[150-160]×[110-120]		Ixh	[6-48]×[4-40]			-
	Cost	-		Value	-			-

Table 51. Features of the instances in LYT.

Name	LYT							
#	Objects				Items			
1	k	1	e	-	m	5	d	-
	LxH	300×200		Ixh	[100-200]×[50-120]			-
	Cost	-		Value	-			-

Table 52. Features of the instances in MA.

Name	MA							
#	Objects				Items			
1	k	5	e	[391-3452]	m	25	d	[368-7308]
	LxH	[1220-2130]×[2100-3050]		Ixh	[205-680]×[431-2130]			-
	Cost	[33550-57747]		Value	-			-
1	k	1	e	-	m	15	d	[90-4410]
	LxH	1850×3670		Ixh	[250-361]×[348-1956]			-
	Cost	-		Value	-			-

Table 53. Features of the instances in MAA.

Name	MAA							
#	Objects				Items			
5	k	1	e	-	m	10	d	-
	LxH	[100-750]×[156-806]		Ixh	[16-445]×[32-449]			-
	Cost	-		Value	-			-
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							-

Table 54. Features of the instances in MB.

Name	MB							
#	Objects				Items			
500	k	5	e	-	m	[20-100]	d	-
	LxH	[5-300]×[5-300]		Ixh	[1-100]×[1-100]			-
	Cost	[10-90000]		Value	-			-
URL	https://paginas.fe.up.pt/~esicup/datasets							-

Table 55. Features of the instances in MB2D.

Name	MB2D							
#	Objects				Items			
900	k	1	e	-	m	[3-736]	d	[1-1487]
	LxH	[404-587]x[50-233]		Ixh	[12-60]x[10-50]			
	Cost	-		Value	-			
URL	http://www.fernuni-hagen.de/evis/service/downloads.shtml							

Table 56. Features of the instances in MG.

Name	MG							
#	Objects				Items			
1	k	1	e	-	m	10	d	-
	LxH	50x50		Ixh	[5-22]x[13-23]			
	Cost	-		Value	-			
1	k	1	e	-	m	29	d	[200-20000]
	LxH	4880x2130		Ixh	[870-2451]x[615-1232]			
	Cost	-		Value	-			

Table 57. Features of the instances in MP.

Name	MP							
#	Objects				Items			
450	k	1	e	-	m	[10-50]	d	[1-85]
	LxH	100x100		Ixh	[10-74]x[10-74]			
	Cost	-		Value	[100-5476]			
URL	https://paginas.fe.up.pt/~esicup/datasets							

Table 58. Features of the instances in MWV.

Name	MWV							
#	Objects				Items			
480	k	1	e	-	m	[24-5000]	d	-
	LxH	100x100		Ixh	[0,09-100]x[0,09-97,62]			
	Cost	-		Value	-			
URL	http://www.vuuren.co.za/benchmarks.html							

Table 59. Features of the instances in NGCUT.

Name	NGCUT							
#	Objects				Items			
12	k	1	e	-	m	[5-10]	d	[1-3]
	LxH	[10-30]x[10-30]		Ixh	[1-30]x[1-30]			
	Cost	-		Value	[4-507]			
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ngcutinfo.html							

Table 60. Features of the instances in NGCUTAP.

Name	NGCUTAP							
#	Objects				Items			
21	k	1	e	-	m	[5-33]	d	[0-[1-5]]
	LxH	[10-100]x[10-100]		Ixh	[1-100]x[1-99]			
	Cost	-		Value	[4-6668]			
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ngcutinfo.html							

Table 61. Features of the instances in NGCUTCON.

Name	NGCUTCON							
#	Objects				Items			
21	k	1	e	1	m	[5-33]	d	[[0-1]-[1-5]]
	LxH	[10-100]x[10-100]			Ixh	[1-100]x[1-99]		
	Cost	1		Value	[4-6668]			
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ngcutinfo.html							

Table 62. Features of the instances in NGCUTFS.

Name	NGCUTFS							
#	Objects				Items			
630	k	1	e	-	m	[40-1000]	d	[0-[1-4]]
	LxH	100x100			Ixh	[2-100]x[2-100]		
	Cost	-		Value	[4-30000]			
URL	http://people.brunel.ac.uk/~mastjjb/jeb/orlib/ngcutinfo.html							

Table 63. Features of the instances in NHU.

Name	NHU							
#	Objects				Items			
1	k	1	e	-	m	10	d	[5-125]
	LxH	35x25			Ixh	[2-18]x[2-35]		
	Cost	-		Value	-			

Table 64. Features of the instances in OF.

Name	OF							
#	Objects				Items			
2	k	1	e	-	m	10	d	[1-4]
	LxH	70x40			Ixh	[9-55]x[4-39]		
	Cost	-		Value	-			
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/							

Table 65. Features of the instances in OKP.

Name	OKP							
#	Objects				Items			
5	k	1	e	-	m	[15-33]	d	[1-5]
	LxH	100x100			Ixh	[1-100]x[1-99]		
	Cost	-		Value	[12-6668]			

Table 66. Features of the instances in ONV.

Name	ONV							
#	Objects				Items			
340	k	[2-6]	e	[1-6]	m	[25-500]	d	-
	LxH	[165-1000]x[153-1000]			Ixh	[1-500]x[1-500]		
	Cost	-		Value	-			
URL	http://www.vuuren.co.za/benchmarks.html							

Table 67. Features of the instances in PGD.

Name	PGD							
#	Objects				Items			
8	k	1	e	-	m	[4-20]	d	[1-5]
	LxH	[8-70]x[4-50]			Ixh	[1-55]x[1-39]		
	Cost	-		Value	-			

Table 68. Features of the instances in PO.

Name	PO							
#	Objects				Items			
7	k	1	e	-	m	[50-15000]	d	-
	LxH	400×600		Ixh	[1-298]×[1-415]			-
	Cost	-		Value	-			-
URL	http://www.computational-logistics.org/orlib/topic/2D%20Strip%20Packing/							

Table 69. Features of the instances in RAND.

Name	RAND							
#	Objects				Items			
400	k	-	e	-	m	[5-20]	d	-
	LxH	-		Ixh	[10-190]×[10-190]			-
	Cost	-		Value	-			-
URL	http://or.dei.unibo.it/library/orthogonal-stock-cutting-problems							

Table 70. Features of the instances in RSS.

Name	RSS							
#	Objects				Items			
7	k	1	e	-	m	[100-550]	d	-
	LxH	[20789-45237]×[23681-35983]		Ixh	[28-9100]×[80-8726]			-
	Cost	-		Value	[448-79389148]			-
URL	https://paginas.fe.up.pt/~esicup/datasets							

Table 71. Features of the instances in SCP.

Name	SCP							
#	Objects				Items			
25	k	1	e	-	m	[4-15]	d	[1-5]
	LxH	[13-145]×[4-100]		Ixh	[1-55]×[1-51]			-
	Cost	-		Value	-			-
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/Strip-cutting/							

Table 72. Features of the instances in SCPL.

Name	SCPL							
#	Objects				Items			
9	k	1	e	-	m	[20-43]	d	[1-9]
	LxH	-×[127-657]		Ixh	[2-121]×[2-48]			-
	Cost	-		Value	-			-
URL	ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/Strip-cutting/							

Table 73. Features of the instances in SPIEKSMA.

Name	SPIEKSMA							
#	Objects				Items			
400	k	1	e	-	m	[24-201]	d	-
	LxH	[100-1000]×[100-1000]		Ixh	[2-999]×[2-999]			-
	Cost	-		Value	-			-
URL	http://or.dei.unibo.it/library/two-constraint-bin-packing-problem							

Table 74. Features of the instances in SS.

Name	SS							
#	Objects				Items			
1	k	2	e	-	m	5	d	[100-180]
	LxH	[48-60]×[96-108]		Ixh	[12-28]×[18-30]			-
	Cost	-		Value	-			-

Table 75. Features of the instances in SSOOYKI.

Name	SSOOYKI							
#	Objects				Items			
7	k	1	e	-	m	[7-20]	d	[100-9000]
	LxH	- ×[200-250]			Ixh	[15-100]×[15-100]		
	Cost	-			Value	-		

Table 76. Features of the instances in STS.

Name	STS							
#	Objects				Items			
4	k	1	e	-	m	[10-30]	d	[1-5]
	LxH	[40-99]×[70-99]			Ixh	[9-44]×[7-49]		
	Cost	-			Value	[90-1877]		

Table 77. Features of the instances in TEST.

Name	TEST							
#	Objects				Items			
2	k	[2-5]	e	[10-20]	m	[30-32]	d	[1-6]
	LxH	[73-2440]×[49-1220]			Ixh	[21,25-2040]×[2-1589]		
	Cost	-			Value	-		

Table 78. Features of the instances in VAG.

Name	VAG							
#	Objects				Items			
1	k	1	e	-	m	5	d	[3-8]
	LxH	100×100			Ixh	[15-40]×[14-44]		
	Cost	-			Value	-		

Table 79. Features of the instances in VASSILIADIS.

Name	VASSILIADIS							
#	Objects				Items			
2	k	1	e	-	m	7	d	[1-18]
	LxH	- ×200			Ixh	[5-103]×[5-50]		
	Cost	-			Value	-		

Table 80. Features of the instances in VENKATESWARLU.

Name	VENKATESWARLU							
#	Objects				Items			
1	k	31	e	[32-232]	m	228	d	-
	LxH	[250-290]×[29-52]			Ixh	[3-21]×[37-76]		
	Cost	-			Value	-		

Table 81. Features of the instances in WANG.

Name	WANG							
#	Objects				Items			
3	k	1	e	-	m	[20-20]	d	[1-4]
	LxH	[33-40]×[69-70]			Ixh	[9-33]×[11-43]		
	Cost	-			Value	-		
2	k	[2-3]	e	[163-3000]	m	[5-10]	d	[150-700]
	LxH	[48-66000]×[96-96000]			Ixh	[12-34000]×[18-41125]		
	Cost	-			Value	-		
URL	https://paginas.fe.up.pt/~esicup/datasets							

Table 82. Features of the instances in WV.

Name	WV									
#	Objects				Items					
20	k	1	e	-	m	[10-5000]	d	-		
	LxH	100×100			l×h	[0.00-100]×[0.00-198.44]				
	Cost	-		Value	-					
URL	https://users.cs.cf.ac.uk/C.L.Mumford/Research%20Topics/layout/Outline.html									

Table 83. Features of the instances in WVINT.

Name	WVINT									
#	Objects				Items					
72	k	1	e	-	m	[25-5000]	d	-		
	LxH	1000×[995-1004]			l×h	[1-1000]×[1-808]				
	Cost	-		Value	-					
URL	http://www.computational-logistics.org/orlib/topic/2D%20Strip%20Packing/									

Table 84. Features of the instances in WWD.

Name	WWD							
#	Objects				Items			
1	k	1	e	-	m	4	d	[1-2]
	LxH	20×15			l×h	[7-15]×[4-7]		
	Cost	-		Value	-			

Table 85. Features of the instances in ZDF.

Name	ZDF									
#	Objects				Items					
16	k	1	e	-	m	[580-75032]	d	-		
	LxH	[100-9000]× -			l×h	[1-1890]×[1-970]				
	Cost	-		Value	-					
URL	https://paginas.fe.up.pt/~esicup/datasets									

3. Instance Generators

In this section, we present, chronologically by year of appearance, six instances generators found in the literature for two-dimensional rectangular cutting and packing problems.

For each generator, Table 86 to Table 91 give the name of the instance generator, a brief description, the article(s) in which it was defined and if available online an internet link for download.

Table 86. Lodi *et al.* Instance Generator.

Name	Lodi <i>et al.</i>
Description	The authors presented an instance generator for the two-dimensional rectangular BPP according to the classes defined by Berkey and Wang [21] and Martello and Vigo [22].
Article(s)	Lodi <i>et al.</i> [89]
URL	https://paginas.fe.up.pt/~esicup/problem_generators

Table 87. Wang and Valenzela Instance Generator.

Name	Wang and Valenzuela
Description	The authors described in this article a recursive procedure to generate datasets of rectangular items to be placed(/extracted) with zero waste on(/from) a single rectangular large object.
Article(s)	Wang and Valenzuela [84]

Table 88. Hopper and Turton Instance Generator.

Name	Hopper and Turton
Description	The authors proposed a generator to address the two-dimensional rectangular (non-)guillotine Strip Packing and Bin Packing Problems.
Article(s)	Hopper and Turton [90]

Table 89. SLOPPGEN Instance Generator.

Name	SLOPPGEN
Description	The authors presented the SLOPPGEN generator for the two-dimensional rectangular SLOPP in which the large object includes one or several defective areas.
Article(s)	Neidlein and Wäscher [91], Neidlein <i>et al.</i> [92]

Table 90. ep2 (and ep3) Instance Generator.

Name	ep2 (and ep3)
Description	The authors presented the ep2 (and ep3) PG for two- and three- dimensional rectangular Knapsack Problems.
Article(s)	Egeblad and Pisinger [31]
URL	http://www.diku.dk/~pisinger/codes.html

Table 91. 2DCPackGen Instance Generator.

Name	2DCPackGen
Description	The authors presented the 2DCPackGen that generates instances for every type of two-dimensional rectangular Cutting and Packing problems according to the typology of Wäscher <i>et al.</i> [3]. The authors review in this article other generators found in the literature and describe their main limitations.
Article(s)	Silva <i>et al.</i> [93]
URL	https://paginas.fe.up.pt/~esicup/problem_generators

References

- [1] E. M. da C. Silva, F. Alvelos, and J. M. V. de Carvalho, “An integer programming model for two- and three-stage two-dimensional cutting stock problems,” *Eur. J. Oper. Res.*, vol. 205, no. 3, pp. 699–708, 2010.
- [2] G. F. Cintra, F. K. Miyazawa, Y. Wakabayashi, and E. C. Xavier, “Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation,” *Eur. J. Oper. Res.*, vol. 191, no. 1, pp. 61–85, 2008.
- [3] G. Wäscher, H. Haußner, and H. Schumann, “An improved typology of cutting and packing problems,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [4] M. Adamowicz and A. Albano, “A Solution of the Rectangular Cutting-Stock Problem,” *IEEE Trans. Syst. Man. Cybern.*, vol. SMC-6, no. 4, pp. 302–310, 1976.
- [5] Y. Cui and B. Huang, “Heuristic for constrained T-shape cutting patterns of rectangular pieces,” *Comput. Oper. Res.*, vol. 39, no. 12, pp. 3031–3039, 2012.
- [6] R. Andrade, E. G. Birgin, and R. Morabito, “Two-stage two-dimensional guillotine

- cutting stock problems with usable leftover,” *Int. Trans. Oper. Res.*, vol. 23, no. 1–2, pp. 121–145, 2016.
- [7] R. Andrade, E. G. Birgin, R. Morabito, and D. P. Ronconi, “MIP models for two-dimensional non-guillotine cutting problems with usable leftovers,” *J. Oper. Res. Soc.*, vol. 65, no. 11, pp. 1649–1663, 2014.
 - [8] A. Bortfeldt and H. Gehring, “New Large Benchmark Instances for the Two-Dimensional Strip Packing Problem with Rectangular Pieces,” *Proc. 39th Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 00, no. C, p. 30b, 2006.
 - [9] J. E. Beasley, “An algorithm for the two-dimensional assortment problem,” *Eur. J. Oper. Res.*, vol. 19, no. 2, pp. 253–261, 1985.
 - [10] R. Alvarez-Valdés, A. Parajón, and J. M. Tamarit, “A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems,” *Comput. Oper. Res.*, vol. 29, no. 7, pp. 925–947, 2002.
 - [11] Y. Cui, Z. Wang, and J. Li, “Exact and heuristic algorithms for staged cutting problems,” *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.*, vol. 219, no. 2, pp. 201–207, 2005.
 - [12] A. R. Babu and N. R. Babu, “Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms,” *Int. J. Prod. Res.*, vol. 37, no. 7, pp. 1625–1643, 1999.
 - [13] A. R. Babu and N. R. Babu, “A generic approach for nesting of 2-D parts in 2-D sheets using genetic and heuristic algorithms,” *CAD Comput. Aided Des.*, vol. 33, no. 12, pp. 879–891, 2001.
 - [14] B. E. E. Bengtsson, “Packing rectangular pieces—a heuristic approach,” *Comput. J.*, vol. 25, no. 3, pp. 253–257, 1982.
 - [15] E. K. Burke, G. Kendall, and G. Whitwell, “A New Placement Heuristic for the Orthogonal Stock-Cutting Problem,” *Oper. Res.*, vol. 52, no. 4, pp. 655–671, 2004.
 - [16] A. El-Bouri, J. Rao, N. Popplewell, and S. Balakrishnan, “An improved heuristic for the two-dimensional cutting stock problem with multiple sized stock sheets,” *Int. J. Ind. Eng. Theory Appl. Pract.*, vol. 13, no. 2, pp. 198–206, 2006.
 - [17] N. Christofides and C. Whitlock, “An Algorithm for Two-Dimensional Cutting Problems,” *Oper. Res.*, vol. 25, no. 1, pp. 30–44, 1977.
 - [18] Y. Cui and B. Huang, “Reducing the number of cuts in generating three-staged cutting patterns,” *Eur. J. Oper. Res.*, vol. 218, no. 2, pp. 358–365, 2012.
 - [19] V. Cung, M. Hifi, and B. Cun, “Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm,” *Int. Trans. Oper. Res.*, vol. 7, no. 3, pp. 185–210, 2000.
 - [20] F. Clautiaux, A. Jouplet, J. Carlier, and A. Moukrim, “A new constraint programming approach for the orthogonal packing problem,” *Comput. Oper. Res.*, vol. 35, no. 3, pp. 944–959, 2008.
 - [21] J. O. Berkey and P. Y. Wang, “Two-Dimensional Finite Bin-Packing Algorithms,” *J. Oper.*

Res. Soc., vol. 38, no. 5, p. 423, 1987.

- [22] S. Martello and D. Vigo, "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," *Manage. Sci.*, vol. 44, no. April 2015, pp. 388–399, 1998.
- [23] Y. Cui, "Heuristic and exact algorithms for generating homogenous constrained three-staged cutting patterns," *Comput. Oper. Res.*, vol. 35, no. 1, pp. 212–225, 2008.
- [24] Y. Cui and Y. Yang, "A recursive branch-and-bound algorithm for constrained homogenous T-shape cutting patterns," *Math. Comput. Model.*, vol. 54, no. 5–6, pp. 1320–1333, 2011.
- [25] Y. Cui and Z. Zhao, "Heuristic for the rectangular two-dimensional single stock size cutting stock problem with two-staged patterns," *Eur. J. Oper. Res.*, vol. 231, no. 2, pp. 288–298, 2013.
- [26] K. Ratanapan and C. H. Dagli, "An object-based evolutionary algorithm for solving rectangular piece nesting problems," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 1997, vol. 2, pp. 989–994.
- [27] K. Ratanapan and C. H. Dagli, "An object-based evolutionary algorithm: the nesting solution," in *IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 581–586.
- [28] C. H. Dagli and P. Poshyanonda, "New approaches to nesting rectangular patterns," *J. Intell. Manuf.*, vol. 8, no. 3, pp. 177–190, 1997.
- [29] K. A. Dowsland, "The Three-Dimensional Pallet Chart : An Analysis of the Factors Affecting the Set of Feasible Layouts for a Class of Two-Dimensional Packing Problems," *J. Oper. Res. Soc.*, vol. 35, no. 10, pp. 895–905, 1984.
- [30] R. M. S. A. El-Aal, "An interactive technique for the cutting stock problem with multiple objectives," *Eur. J. Oper. Res.*, vol. 78, no. 3, pp. 304–317, 1994.
- [31] J. Egeblad and D. Pisinger, "Heuristic approaches for the two- and three-dimensional knapsack packing problem," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1026–1049, 2009.
- [32] D. Fayard, M. Hifi, and V. Zissimopoulos, "An efficient approach for large-scale two-dimensional guillotine cutting stock problems," *J. Oper. Res. Soc.*, vol. 49, no. 12, pp. 1270–1277, 1998.
- [33] E. Ferreira and J. F. Oliveira, "A note on Fekete and Schepers' algorithm for the non-guillotinable two-dimensional packing problem." Technical report, FEUP, pp. 2–5, 2005.
- [34] M. Gardner, "Mathematical Games. The problem of Mrs. Perkins' quilt, and answers to last month's puzzles.,," *Sci. Am.*, vol. 2, 1966.
- [35] J. E. Beasley, "Algorithms for Unconstrained Two-Dimensional Guillotine Cutting," *J. Oper. Res. Soc.*, vol. 36, no. 4, pp. 297–306, 1985.
- [36] E. Hadjiconstantinou and N. Christofides, "An exact algorithm for general, orthogonal,

two-dimensional knapsack problems," *Eur. J. Oper. Res.*, vol. 83, no. 1, pp. 39–56, 1995.

- [37] J. Herz, "Recursive Computational Procedure for Two-dimensional Stock Cutting," *IBM J. Res. Dev.*, vol. 16, no. 5, pp. 462–469, 1972.
- [38] M. Hifi, "The DH/KD algorithm: a hybrid approach for unconstrained two-dimensional cutting problems," *Eur. J. Oper. Res.*, vol. 97, no. 1, pp. 41–52, 1997.
- [39] M. Hifi, "An improvement of Viswanathan and Bagchi's exact algorithm for constrained two-dimensional cutting stock," *Comput. Oper. Res.*, vol. 24, no. 8, pp. 727–736, 1997.
- [40] M. Hifi, "Exact algorithms for large-scale unconstrained two and three staged cutting problems," *Comput. Optim. Appl.*, vol. 18, no. 1, pp. 63–88, 2001.
- [41] E. Hopper, "Two-dimensional Packing utilising Evolutionary Algorithms and other Meta-Heuristic Methods," University of Wales, 2000.
- [42] E. Hopper and B. C. H. Turton, "An empirical study of meta-heuristics applied to 2D rectangular bin packing," *Stud. Inform. Universalis*, vol. 2, no. 1, pp. 77–106, 2001.
- [43] E. Hopper and B. C. H. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem," *Eur. J. Oper. Res.*, vol. 128, no. 1, pp. 34–57, 2001.
- [44] M. Hifi and V. Zissimopoulos, "Une amélioration de l'algorithme récursif de Herz pour le problème de découpe à deux dimensions," *RAIRO, Rech. Opérationnelle*, vol. 30, no. 2, pp. 111–125, 1996.
- [45] M. Hifi and V. Zissimopoulos, "A recursive exact algorithm for weighted two-dimensional cutting," *Eur. J. Oper. Res.*, vol. 91, no. 3, pp. 553–564, 1996.
- [46] S. Israni and J. Sanders, "Two-dimensional cutting stock problem research: A review and a new rectangular layout algorithm," *J. Manuf. Syst.*, vol. 1, pp. 169–182, 1982.
- [47] S. Imahori, M. Yagiura, S. Umetani, S. Adachi, and T. Ibaraki, "Local search algorithms for the two-dimensional cutting stock problem with a given number of different patterns," *Oper. Res. Comput. Sci. Interfaces Ser.*, vol. 32, 2005.
- [48] S. Jakobs, "On genetic algorithms for the packing of polygons," *Eur. J. Oper. Res.*, vol. 88, no. 1, pp. 165–181, 1996.
- [49] J. Q. Jiang, Y. C. Liang, X. H. Shi, and H. P. Lee, "A Hybrid Algorithm Based on PSO and SA and Its Application for Two-Dimensional Non-guillotine Cutting Stock Problem," in *Lecture Notes in Computer Science*, vol. 3037, no. ICCS 2004, IEEE, 2004, pp. 666–669.
- [50] R. E. Korf, M. D. Moffitt, and M. E. Pollack, "Optimal rectangle packing," *Ann. Oper. Res.*, vol. 179, no. 1, pp. 261–295, 2010.
- [51] B. Kröger, "Guillotineable bin packing: A genetic approach," *Eur. J. Oper. Res.*, vol. 84, no. 3, pp. 645–661, 1995.
- [52] K. K. Lai and J. W. M. Chan, "Developing a simulated annealing algorithm for the cutting stock problem," *Comput. Ind. Eng.*, vol. 32, no. 1, pp. 115–127, 1997.
- [53] T. W. Leung, C. K. Chan, and M. D. Troutt, "Application of a mixed simulated annealing-

genetic algorithm heuristic for the two-dimensional orthogonal packing problem," *Eur. J. Oper. Res.*, vol. 145, no. 3, pp. 530–542, 2003.

- [54] T. W. Leung, C. H. Yung, and M. D. Troutt, "Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem," *Comput. Ind. Eng.*, vol. 40, no. 3, pp. 201–214, 2001.
- [55] R. Morabito and M. N. Arenales, "Optimizing the cutting of stock plates in a furniture company," *Int. J. Prod. Res.*, vol. 38, no. 12, pp. 2725–2742, 2000.
- [56] R. Morabito, M. N. Arenales, and V. F. Arcaro, "An and-or-graph approach for two-dimensional cutting problems," *Eur. J. Oper. Res.*, vol. 58, no. 2, pp. 263–271, 1992.
- [57] D. Pisinger and M. Sigurd, "The two-dimensional bin packing problem with variable bin sizes and costs," *Discret. Optim.*, vol. 2, no. 2, pp. 154–167, 2005.
- [58] D. Mack and A. Bortfeldt, "A heuristic for solving large bin packing problems in two and three dimensions," *Cent. Eur. J. Oper. Res.*, vol. 20, no. 2, pp. 337–354, 2012.
- [59] R. Morabito and V. Garcia, "The cutting stock problem in a hardboard industry: A case study," *Comput. Oper. Res.*, vol. 25, no. 6, pp. 469–485, 1998.
- [60] R. Morabito and V. Pureza, "A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem," *Ann. Oper. Res.*, vol. 179, no. 1990, pp. 1–25, 2010.
- [61] C. L. Mumford-Valenzuela, J. Vick, and P. Y. Wang, "Heuristics for large strip packing problems with guillotine patterns: An empirical study," in *Metaheuristics: computer decision-making*, vol. 86, 2003, pp. 501–522.
- [62] J. E. Beasley, "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure," *Oper. Res.*, vol. 36, no. 1, pp. 49–64, 1985.
- [63] J. E. Beasley, "A population heuristic for constrained two-dimensional non-guillotine cutting," *Eur. J. Oper. Res.*, vol. 156, no. 3, pp. 601–627, 2004.
- [64] K. Novianingsih, R. Hadianti, and S. Uttunggadewa, "Column generation technique for solving two-dimensional cutting stock problems: method of stripe approach," *J. Indones. Math. Soc.*, vol. 13, no. 2, pp. 161–172, 2012.
- [65] J. F. Oliveira and J. S. Ferreira, "An improved version of Wang's algorithm for two-dimensional cutting problems," *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 256–266, 1990.
- [66] S. P. Fekete and J. Schepers, "On more-dimensional packing III: Exact Algorithms," ZPR Technical Report 97.290, 2000.
- [67] F. G. Ortmann, N. Ntene, and J. H. van Vuuren, "New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems," *Eur. J. Oper. Res.*, vol. 203, no. 2, pp. 306–315, 2010.
- [68] V. Parada, A. Gómes de Alvarenga, and J. de Diego, "Exact solutions for constrained two-dimensional cutting problems," *Eur. J. Oper. Res.*, vol. 84, no. 3, pp. 633–644, 1995.
- [69] E. Pinto and J. F. Oliveira, "Algorithm based on graphs for the non-guillotinable two-

dimensional packing problem," in *Second ESICUP Meeting*, 2005.

- [70] S. Martello and M. Monaci, "Models and algorithms for packing rectangles into the smallest square," *Comput. Oper. Res.*, vol. 63, pp. 161–171, 2015.
- [71] M. Russo, A. Sforza, and C. Sterle, "An exact dynamic programming algorithm for large-scale unconstrained two-dimensional guillotine cutting problems," *Comput. Oper. Res.*, vol. 50, pp. 97–114, 2014.
- [72] M. Hifi, "Exact algorithms for the guillotine strip cutting/packing problem," *Comput. Oper. Res.*, vol. 25, no. 11, pp. 925–940, 1998.
- [73] M. Hifi, "The Strip Cutting/Packing Problem: Incremental Substrip Algorithms-Based Heuristics," *Pesqui. Operacional, Spec. Issue Cut. Pack. Probl.*, vol. 19, pp. 169–188, 1999.
- [74] F. C. R. Spieksma, "Branch-and-bound algorithm for the two-dimensional vector packing problem," *Comput. Oper. Res.*, vol. 21, no. 1, pp. 19–25, 1994.
- [75] A. Caprara and P. Toth, "Lower bounds and algorithms for the 2-dimensional vector packing problem," *Discret. Appl. Math.*, vol. 111, no. 3, pp. 231–262, 2001.
- [76] B. A. Skalbeck and H. K. Schultz, "Reducing Trim Waste in Panel Cutting Using Integer and Linear Programming," *Proc. West. AIDS Conf.*, pp. 145–147, 1976.
- [77] Y. Shiomi *et al.*, "The solution of 2-dimensional rectangular cutting stock problem considering cutting process," *Proc. 3rd IEEE Int. Conf. Autom. Sci. Eng. IEEE CASE 2007*, pp. 140–145, 2007.
- [78] S. Tschöke and N. Holthöfer, "A New Parallel Approach to the Constrained Two-Dimensional Cutting Stock Problem," in *Workshop on Algorithms for Irregularly Structured Problems*, 1995, pp. 285–299.
- [79] H. H. Yanasse, A. S. I. Zinober, and R. G. Harris, "Two-dimensional cutting stock with multiple stock sizes," *J. Oper. Res. Soc.*, vol. 42, no. 8, pp. 673–683, 1991.
- [80] A. C. G. Vianna, M. N. Arenales, and M. C. N. Gramani, "Two-Stage and Constrained Two-Dimensional Guillotine Cutting Problems," *USP, no. Notas-Série Computação nº 69*, pp. 1–28, 2003.
- [81] V. S. Vassiliadis, "Two-dimensional stock cutting and rectangle packing: Binary tree model representation for local search optimization methods," *J. Food Eng.*, vol. 70, no. 3, pp. 257–268, 2005.
- [82] P. Venkateswarlu, "The trim-loss problem in a wooden container manufacturing company," *J. Manuf. Syst.*, vol. 20, no. 3, pp. 166–176, 2001.
- [83] P. Y. Wang, "Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems," *Oper. Res.*, vol. 31, no. 3, pp. 573–586, 1983.
- [84] P. Y. Wang and C. L. Valenzuela, "Data set generation for rectangular placement problems," *Eur. J. Oper. Res.*, vol. 134, no. 2, pp. 378–391, 2001.
- [85] L. Wei, W. C. Oon, W. Zhu, and A. Lim, "A skyline heuristic for the 2D rectangular

packing and strip packing problems," *Eur. J. Oper. Res.*, vol. 215, no. 2, pp. 337–346, 2011.

- [86] J. Wan, Y. Wu, and H. Dai, "A Pattern Combination Based Approach to Two-Dimensional Cutting Stock Problem," in *Advances in Natural Computation: First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings, Part III*, 2005, pp. 332–336.
- [87] D. Zhang, L. Wei, S. C. H. Leung, and Q. Chen, "A Binary Search Heuristic Algorithm Based on Randomized Local Search for the Rectangular Strip-Packing Problem," *INFORMS J. Comput.*, vol. 25, no. 2, pp. 332–345, 2013.
- [88] S. C. H. Leung and D. Zhang, "A fast layer-based heuristic for non-guillotine strip packing," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13032–13042, 2011.
- [89] A. Lodi, S. Martello, and D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems," *INFORMS J. Comput.*, vol. 11, no. 4, pp. 345–357, 1999.
- [90] E. Hopper and B. C. H. Turton, "Problem Generators for Rectangular Packing Problems," *Stud. Inform. Univ.*, vol. 2, pp. 123–136, 2002.
- [91] V. Neidlein and G. Wäscher, "SLOPPGEN: A problem generator for the two- dimensional rectangular single large object placement problem with a single defect," *FEMM Work. Pap.*, pp. 1–8, 2008.
- [92] V. Neidlein, A. Scholz, and G. Wäscher, "SLOPPGEN: A problem generator for the two- dimensional rectangular single large object placement problem with defects," *Int. Trans. Oper. Res.*, vol. 23, no. 1–2, pp. 173–186, 2016.
- [93] E. M. da C. Silva, J. F. Oliveira, and G. Wäscher, "2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems," *Eur. J. Oper. Res.*, vol. 237, no. 3, pp. 846–856, 2014.

Resources for Two-dimensional (and Three-dimensional) Cutting and Packing Solution Methods Research

Abstract We present a set of resources that we have created when studying two-dimensional rectangular cutting and packing solution methods. We have compiled and converted the datasets found in the literature into a common format to ease the data input to the solution methods implementations. We present graphical user interfaces for instance and cutting plan visualisation. Finally, we present the website developed that, besides hosting all resources that we have made available, contains a set of utilities that can help on the characterisation of the articles available in the literature for two-dimensional cutting and packing problems and the relations between them. These resources are available for use and open for contributions.

Keywords: Resources, Tools, Instances, Graphical User Interface, Rectangular, Two-dimensional, Three-dimensional, Cutting and Packing Problem

1. Introduction

Cutting and packing problems intent to place (either to cut or pack) a set of items into larger objects considering some objective, i.e., minimization of the number of objects used to place all given items or maximization of the area occupied by items placed, without overlapping, inside one object. The cutting and packing problem is a wide family of related problems, we refer to Wäscher *et al.* [1] for a great overview of this problem family and for a typology to classify the problems through their common characteristics.

Hopper and Turton [2] stated that: “*In general, during the development of packing algorithms it is useful to visualise final and intermediate layouts in order to judge the quality and the correctness of the packing routines more easily*”. As cutting and packing researchers, we also have found the need to create tools that helped us to understand in a simpler and clearer manner the problems in which we were focused on.

Most often researchers find themselves searching, adapting or creating new tools that facilitate the visualisation and analysis of the current state of their research, shifting the attention from the most important task, i.e., the research for new and better solution methods.

One common problem shared by researchers is to find and use the instances generated by other researchers when comparing results of their solution methods. Existent dataset libraries do not share the same file format to define the instances (in some case, even in the same library the format varies) and are not comprehensive enough forcing the researchers to search for another resource in order to find a particular dataset or instance. We have compiled more than 6000 instances (84 datasets) found in the literature and converted them into one specific format to help recognise the instance structure and to unify the data input for the solution method's implementations.

We present three graphical user interfaces (GUI) that allow to easily visualise the instances and the cutting plans obtained by the solution methods implementations.

We rely heavily on the key-value pair format JavaScript Object Notation¹ (JSON) to structure all our data, i.e., instance definition and cutting plans. The characteristics associated with this format justify our choice, being the most relevant in the context of this work the followings:

- Lightweight format for store and transport data.
- Easy for humans to read, i.e., self-describing.
- Easy for machines to parse and generate (most of the programming language have some sort of built-in or available library to easily read and write JSON files).

As we believe that any work only has value when effectively used and shared, the datasets and tools mentioned in this work are freely available (MIT License²) at the repository <https://github.com/Oscar-Oliveira>. We hope that other researchers and developers engage and contribute to the improvement of these tools and with the growth of the dataset library.

For those who just want to make use of the resources that we have made available, we have created a platform, i.e., website, for hosting these resources. This website also contains a set

¹ See <http://www.json.org/>.

² See <https://opensource.org/licenses/MIT> for license template and further resources on this license.

of tools that allows to visualise and filter the existent literature related to the two-dimensional rectangular cutting and packing problem.

For the best of our knowledge, no resource set, like the one presented in this paper, exists or have been made available so far.

The rest of the paper is organized as follows. Section 2 describes the datasets conversion to the JSON format. Section 3 presents the GUI to visualise the instances converted to the JSON format and the two-dimensional and three-dimensional cutting plan viewers. Section 4 describes the main features of the website for hosting the resources created for helping our (and future) research. Finally, conclusions and future work directions are given in the last section.

The examples given in the following sections consider the two-dimensional instance OF1 from Oliveira and Ferreira [3] and the three-dimensional instance 10 (Class 8, $m = 200$) of the dataset generated by Martello *et al.* [4]³.

2. Datasets

We have compiled more than 80 datasets usually used by the researchers when evaluating and comparing their solution method implementations. As was expected, many formats to define the instances characteristics have emerged from this diversity, and to ease the data input for the implementation of the solution methods, we converted all the instances to JSON format with the structure depicted in Figure 1.

The instances were converted considering the following set of rules:

- Objects with the same dimensions are aggregated.
- The cost equals the area of the object if the cost is not defined.
- Items with the same dimensions and value are aggregated.
- The demand equals 1 if the demand is not defined.
- The value equals the area of the items, if the value is not defined.

³ Generator available at <http://hjemmesider.diku.dk/~pisinger/codes.html>

Each instance converted to this format has a name, a set of objects and a set of items. Each object has a length, height, stock, and cost, while each item has length, height, demand, a maximum demand (for double-constrained problems) and value.

```
{
  "Name": STRING,
  "Objects": [
    {
      "Length": INT,
      "Height": INT,
      "Stock": null|INT,
      "Cost": INT|DOUBLE
    }
  ],
  "Items": [
    {
      "Length": INT|DOUBLE,
      "Height": INT|DOUBLE,
      "Demand": INT,
      "DemandMax": null|INT,
      "Value": INT|DOUBLE
    }
  ]
}
```

Figure 1. JSON structure.

The file of instance OF1⁴ is depicted in Figure 2.

```
70 40
10
29 5 1
9 39 4
55 9 1
31 15 1
11 16 2
23 21 3
29 14 4
16 19 3
9 36 2
22 4 2
```

Figure 2. Instance OF1 – Original format.

As it can be seen, the instance is not self-described, to parse of this instance we must rely upon the position of the data, i.e., the first line defines the object dimensions, the second line defines the number of items, then for each item/line, the item dimensions and the demand. In this type of format, the reader must be perfectly aware of the structure and slight variations that can occur (as it is frequent that instances in the same dataset have minor differences, e.g., two spaces or a tab rather than one space). The same instance converted to our JSON format is depicted in Figure 3. After converted the instances are more readable, self-explanatory and less error-prone.

⁴ Obtained from <ftp://cermsem.univ-paris1.fr/pub/CERMSEM/hifi/2Dcutting/>

```
{
  "Name": "OF1",
  "Objects": [
    { "Length": 70, "Height": 40, "Stock": null, "Cost": 2800 }
  ],
  "Items": [
    { "Length": 29, "Height": 5, "Demand": 1, "DemandMax": null, "Value": 145 },
    { "Length": 9, "Height": 39, "Demand": 4, "DemandMax": null, "Value": 351 },
    { "Length": 55, "Height": 9, "Demand": 1, "DemandMax": null, "Value": 495 },
    { "Length": 31, "Height": 15, "Demand": 1, "DemandMax": null, "Value": 465 },
    { "Length": 11, "Height": 16, "Demand": 2, "DemandMax": null, "Value": 176 },
    { "Length": 23, "Height": 21, "Demand": 3, "DemandMax": null, "Value": 483 },
    { "Length": 29, "Height": 14, "Demand": 4, "DemandMax": null, "Value": 406 },
    { "Length": 16, "Height": 19, "Demand": 3, "DemandMax": null, "Value": 304 },
    { "Length": 9, "Height": 36, "Demand": 2, "DemandMax": null, "Value": 324 },
    { "Length": 22, "Height": 4, "Demand": 2, "DemandMax": null, "Value": 88 }
  ]
}
```

Figure 3. Instance OF1 - Converted to JSON.

3. GUI for instance and cutting plan visualisation

In this section, we present the instance and cutting plan viewers. The usage of these tools is very similar, to visualise the content of a JSON file in a viewer, drag-and-drop the file inside the top-left corner box. At the top-right corner, the viewers present an icon, depicted in Figure 4, that provides more information and help.



Figure 4. Menu icon.

3.1. Instance viewer

To help visualise the converted instances, we have created an instance viewer, depicted in Figure 5 with instance OF1. The left column presents the characteristics of the objects. The centre column presents the items characteristics, and at the right column, the instance is illustrated with the objects colourized with a darker grey and the items with a lighter grey.

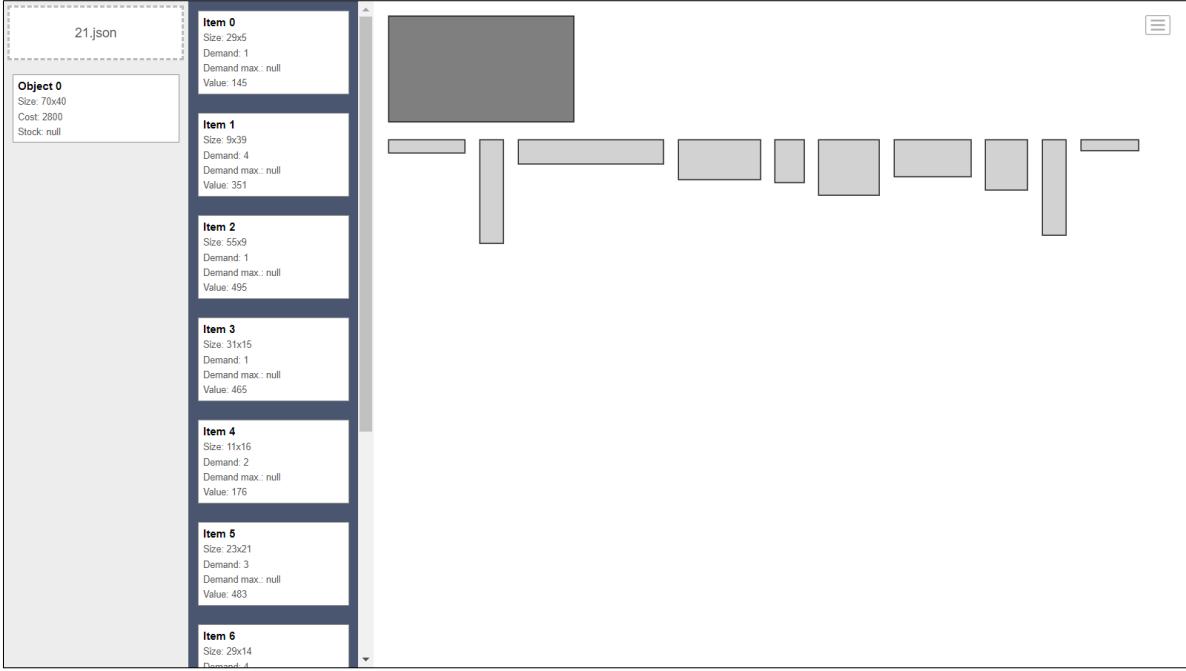


Figure 5. Instance OF1 – Rendered.

3.2. Two-dimensional cutting plan viewer

We provide a tool to help visualise two-dimensional cutting plans. Solution methods can export the solution obtained to a JSON file as depicted in Figure 6. As it can be observed, the key pattern can contain a set of patterns each of then seen as a shelf with position (x and y), dimension (w and h), frequency (f), pattern trim loss (tl) and first cut orientation (o). Noteworthy, shelves can contain other shelves. Solving a 2D 2-staged Single Large Object Placement Problem (SLOPP, see Wäscher *et al.* [1]) using the data of instance OF1, Figure 6 depicts the JSON file that contains the resulting cutting plan to be visualised using the two-dimensional viewer.

```
{
  "pattern": [
    { "x": 0, "y": 0, "w": 70, "h": 40, "f": 1, "tl": 87, "o": "Horizontal", "shelves": [
      { "x": 0, "y": 0, "w": 70, "h": 21, "shelves": [
        { "x": 0, "y": 0, "w": 23, "h": 21 },
        { "x": 23, "y": 0, "w": 23, "h": 21 },
        { "x": 46, "y": 0, "w": 23, "h": 21 }
      ]},
      { "x": 0, "y": 21, "w": 70, "h": 19, "shelves": [
        { "x": 0, "y": 0, "w": 16, "h": 19 },
        { "x": 16, "y": 0, "w": 16, "h": 19 },
        { "x": 32, "y": 0, "w": 16, "h": 19 },
        { "x": 48, "y": 0, "w": 11, "h": 16 },
        { "x": 59, "y": 0, "w": 11, "h": 16 }
      ]}
    ]}
  ]
}
```

Figure 6. 2D 2-staged SLOPP – 2D JSON.

The cutting plan viewer, depicted in Figure 7, presents in the left column the information of patterns contained in the JSON file.

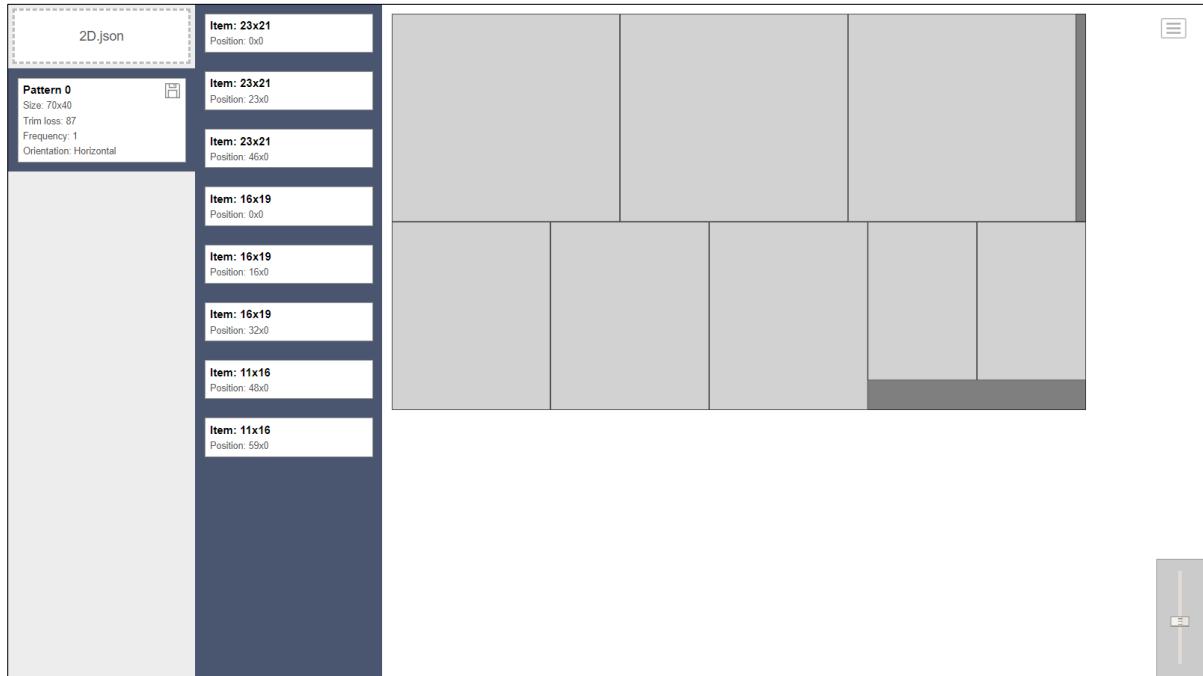


Figure 7. 2D 2-staged SLOPP – 2D render.

Clicking in the top-right disk icon of each pattern, depicted in Figure 8, allows to download an image of the respective pattern.



Figure 8. Disk icon.

The centre column presents the items contained in the currently selected pattern, identifying the dimension and position of each one of them. The right column illustrates the currently selected pattern. A zoom controller is present at the bottom-right corner of the two-dimensional cutting plan viewer.

3.3. Three-dimensional cutting plan viewer

As for the two-dimensional case, solution methods can export the obtained solution to a JSON file, as depicted in Figure 9, to be rendered in the three-dimensional cutting plan viewer.

Continuing with instance OF1 solved as a 2D 2-staged SLOPP, Figure 9 depicts the JSON file that contains the resulting cutting plan to be visualised in the three-dimensional viewer (Figure 10). This JSON file contains the bins dimensions (w, h, d), the frequency of cut (f), trim

loss (*t₁*) and the items to be placed inside the respective bin (*items*). For each item, we can find the identification (*i*), dimensions (*w*, *h*, *d*), position (*x*, *y*, *z*) and a Boolean flag to identify if the item was rotated by the solution method (*r*).

```
{"box": [
  {"w": 70, "h": 40, "d": 4, "f": 1, "t1": 87, "items": [
    {"i": 5, "w": 23, "h": 21, "d": 4, "x": 0, "y": 0, "z": 0, "r": 0},
    {"i": 5, "w": 23, "h": 21, "d": 4, "x": 23, "y": 0, "z": 0, "r": 0},
    {"i": 5, "w": 23, "h": 21, "d": 4, "x": 46, "y": 0, "z": 0, "r": 0},
    {"i": 7, "w": 16, "h": 19, "d": 4, "x": 0, "y": 21, "z": 0, "r": 0},
    {"i": 7, "w": 16, "h": 19, "d": 4, "x": 16, "y": 21, "z": 0, "r": 0},
    {"i": 7, "w": 16, "h": 19, "d": 4, "x": 32, "y": 21, "z": 0, "r": 0},
    {"i": 4, "w": 11, "h": 16, "d": 4, "x": 48, "y": 21, "z": 0, "r": 0},
    {"i": 4, "w": 11, "h": 16, "d": 4, "x": 59, "y": 21, "z": 0, "r": 0}
  ]}
]
```

Figure 9. 2D 2-staged SLOPP – 3D JSON.

The three-dimensional cutting plan viewer, as depicted in Figure 10, presents to the left the bins contained in the JSON file, the centre column the items contained inside the currently selected bin. Each item has a radio button associated that allows stopping the rendering at the current item to visualise the packing order followed by the solution method. At the right column, the packing is rendered allowing to rotate in all axis using the keyboard navigation keys (a.k.a., arrow keys) and mouse.

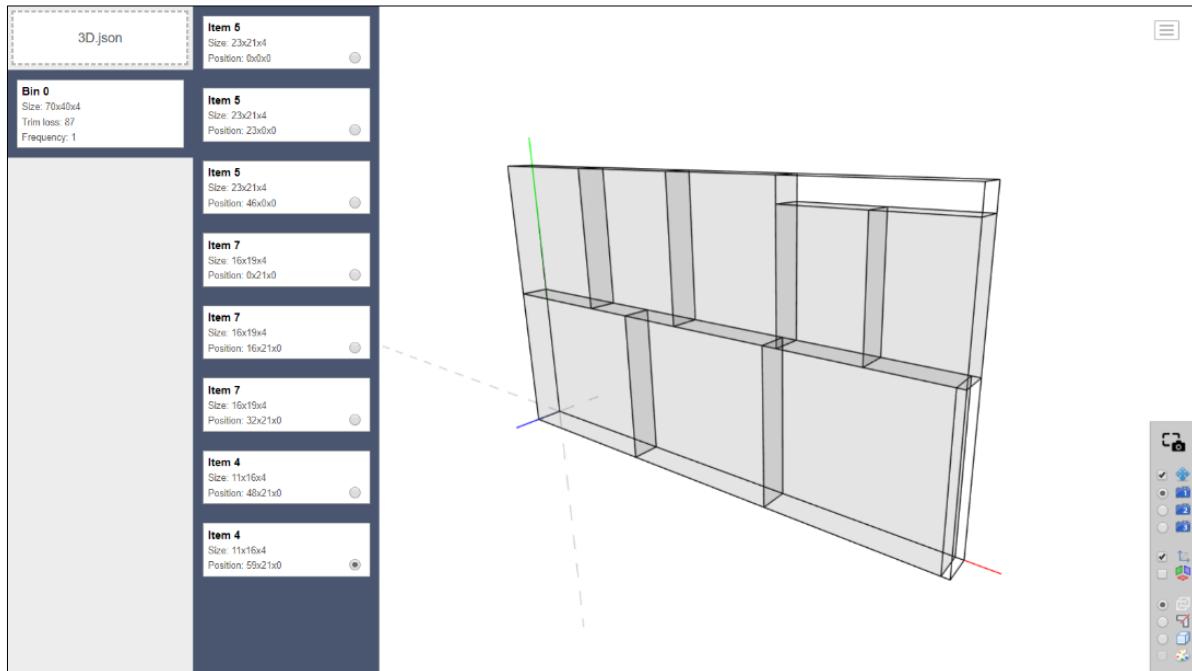


Figure 10. 2D 2-staged SLOPP – 3D render.

At the bottom-right corner a toolset is present that, in order of appearance from top to bottom, capture a picture of the currently selected bin, block the navigation, toggle camera type, toggle the visibility of the axis, toggle the visibility of the planes and the last three radio buttons allow to display the items as wireframe (Figure 11), as solid unoccupied space (Figure 12) and as solid items (Figure 13). With the latter display selected, the bottom checkbox assigns a randomly a colour using a grey scale for the items.

Figure 11 to Figure 13 depict the different visualisations of the three-dimensional instance solved as a three-dimensional Single Bin Size Bin Packing Problem (SBSBPP, see Wäscher *et al.* [1]).

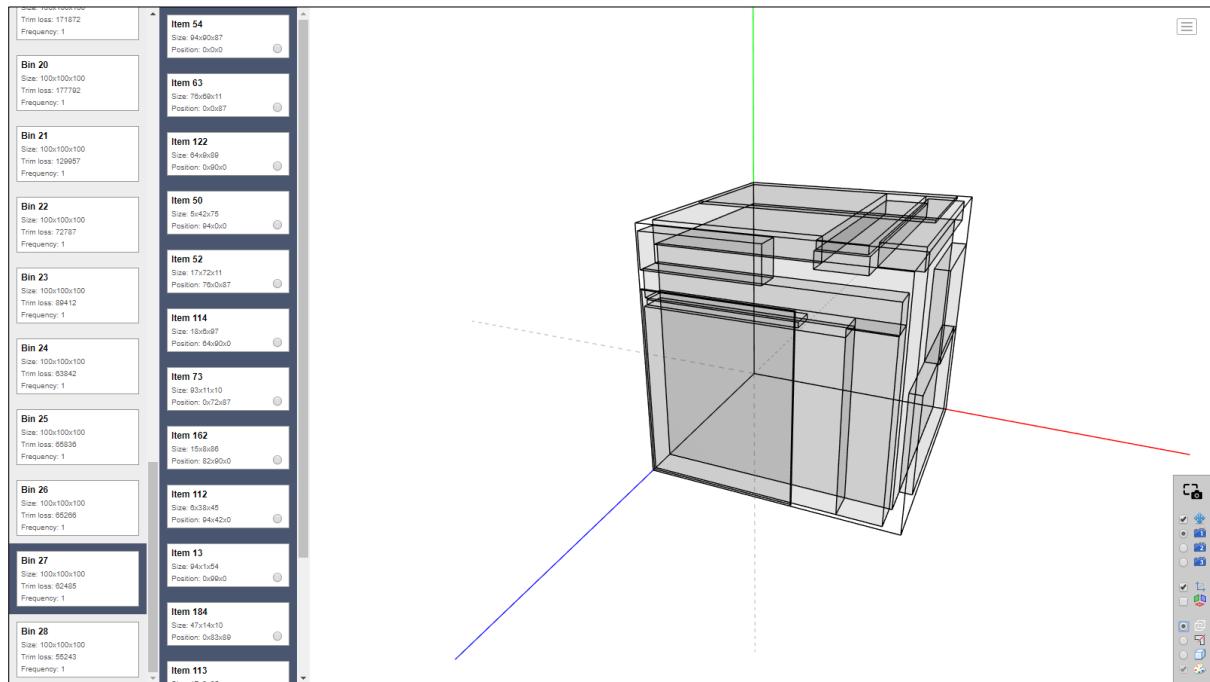


Figure 11. 3D SBSBPP – Wireframe render.

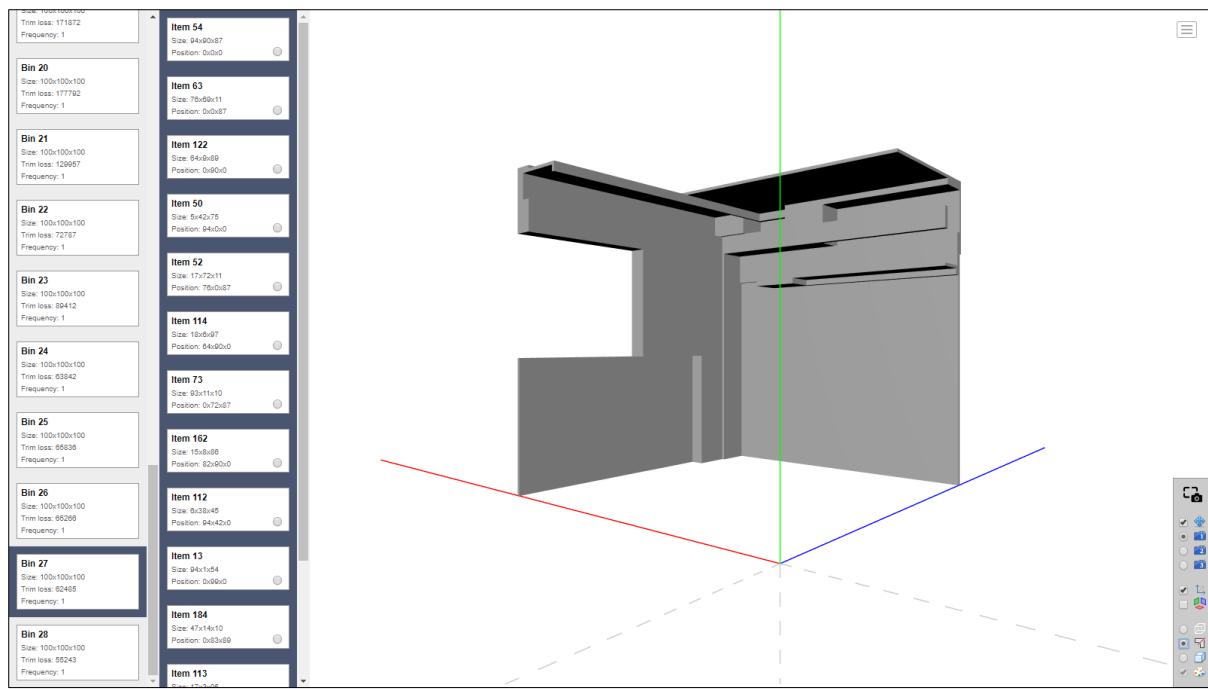


Figure 12. 3D SBSBPP – Solid unoccupied space render.

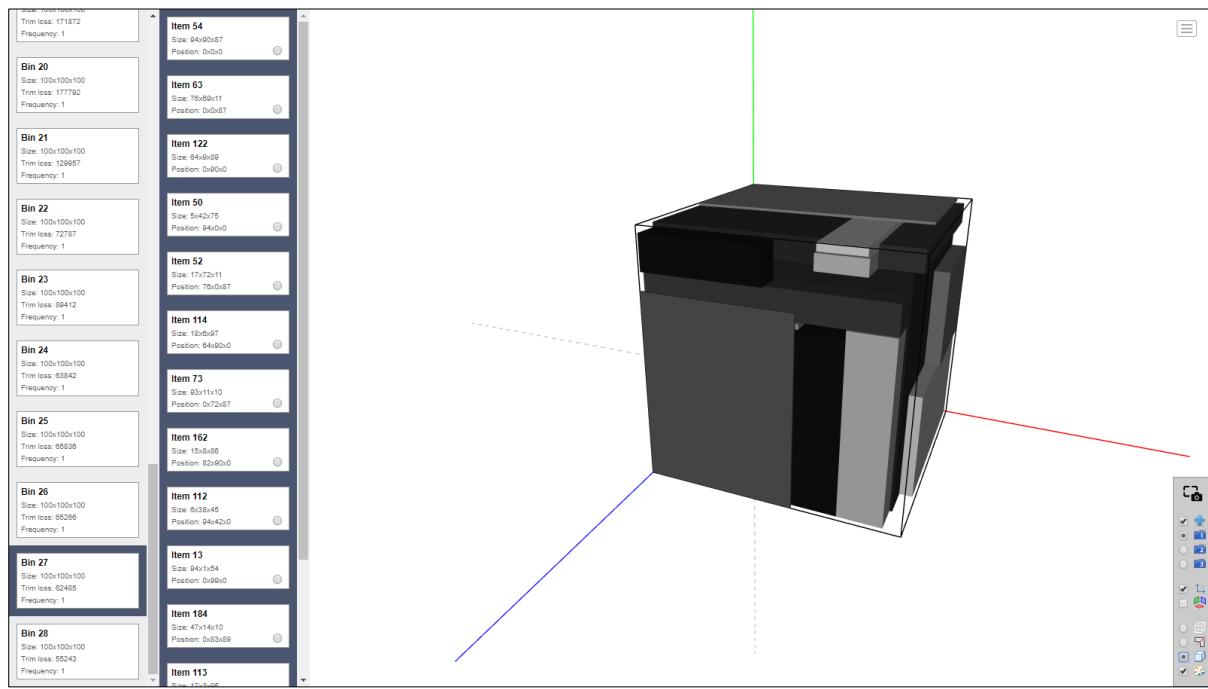


Figure 13. 3D SBSBPP – Solid render.

4. Website

The website created to group all resources developed during our research can be accessed through the following URL: <https://oscar-oliveira.github.io/2D-Cutting-and-Packing/>.

The website presents to the left a menu (black vertical bar, see Figure 14) with nine icons running from top to bottom letting the user switch between resources. In short, the menu items allow the access to:

-  Homepage
-  Literature review - Visualisations
-  Literature review - Quick view
-  Literature review - Filter
-  List of datasets
-  List of instance generators
-  2D instance viewer (see section 3.1.)
-  2D cutting plan viewer (see section 3.2.)
-  3D cutting plan viewer (see section 3.3.)

The homepage (see Figure 14) contains some metrics on the data gathered.



Figure 14. Homepage.

To minimize the work required updating the information, all data is stored in JSON format and used to generate the content. Four groups of data are stored:

- Keywords – List of keywords that can be used to characterise the articles.
- Datasets – List of datasets (see Section 2). For each dataset, the following information is stored: name, reference and URL from which the instances were obtained.
- Generators – List of the two-dimensional instance generators found in the literature. For each generator, the following information is stored: short name, reference, URL from which the source code was obtained and download URL.
- Articles – List of articles related to the two-dimensional cutting and packing problems. For each article, the following information is stored: name (key), year of publication, URL⁵, reference, list of keywords that characterise the article, list of datasets used by the authors, and list of articles (keys) used for evaluating the results obtained.

Next, we briefly describe the webpages that relates to the literature review and to the list of datasets and instance generators. We do not describe the last three menu items as they allow access to the GUI for instance and cutting plan visualisation described in Section 3.

4.1. Literature review - Visualisations

This webpage lets visualise the literature related to the two-dimensional cutting and packing problem. Observing Figure 15, it can be seen on the left side a vertical action bar that allows to choose: 1) the type of visualisation, 2) the order in which the articles appear, i.e., sorted by year of publication or by name, 3) the diagram size, 4) to download the diagram as a Scalable Vector Graphics⁶ (SVG) and 5) the action button. When the visualisation type *Comparison* is selected an additional option is made available that allows to choose the line tension between articles.

The types of visualisation generate diagrams that depicts the relation between: 1) the articles and keywords (type *Keywords*), 2) the articles and used datasets (type *Datasets*) and 3) the articles used for comparison when evaluating the results obtained (type *Comparison*). Rolling the cursor over a keyword, dataset or article highlight the existent relations. It is worth noting

⁵ Whenever available, we use the Digital Object Identifier (DOI, see <https://www.doi.org/>) based URL.

⁶ <https://www.w3.org/Graphics/SVG/>

that at the top-left corner of the diagram area, a legend appears with the currently selected item.

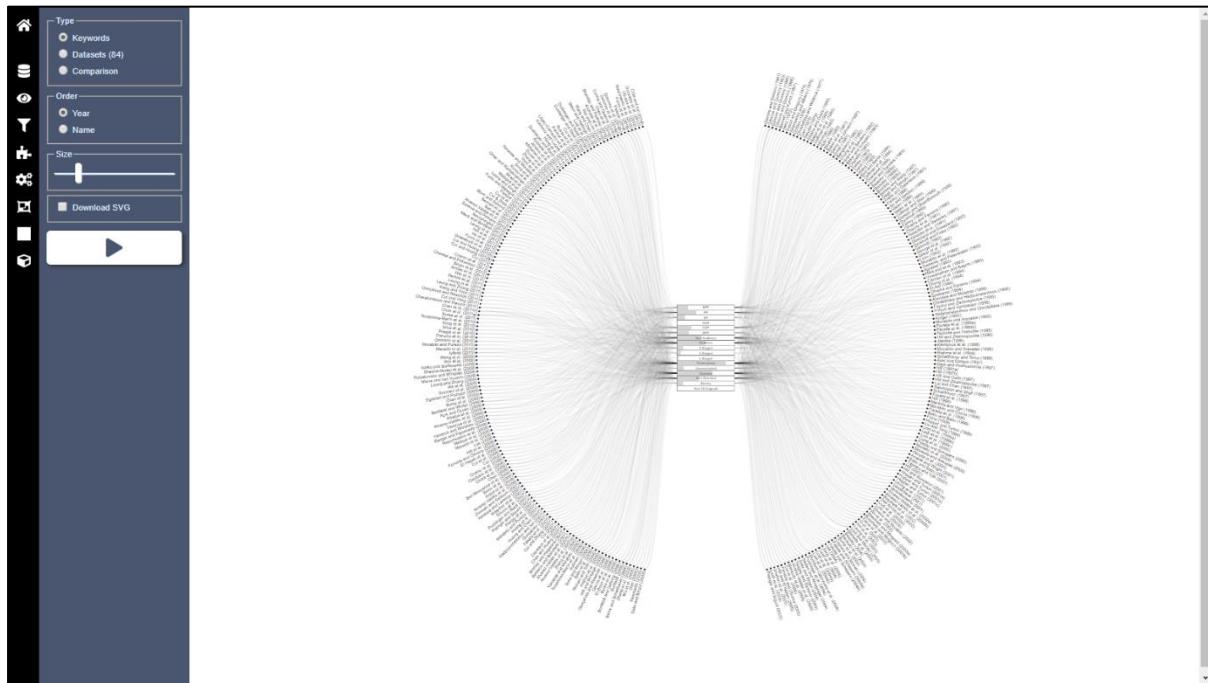


Figure 15. Literature review.

To characterize the articles, we have chosen the following keywords:

- IIPP, PP, KP, ODP, CSP and BPP - to identify the (basic) problems types following Wäscher's typology [1].
- Guillotine and Non-guillotine - to identify the cut type.
- 2-staged, 3-staged and k-staged - to identify the number of cuts stages.
- Constrained and Unconstrained - to identify if the demand is bounded or not.
- Oriented and Non-oriented - to identify if rotation of items can be allowed or not.
- Survey - to identify if the article is a survey (or a review).
- Non-orthogonal - to identify articles that deal with non-orthogonal placement.

Figure 16 depicts the articles that are characterised by the keyword Non-guillotine.

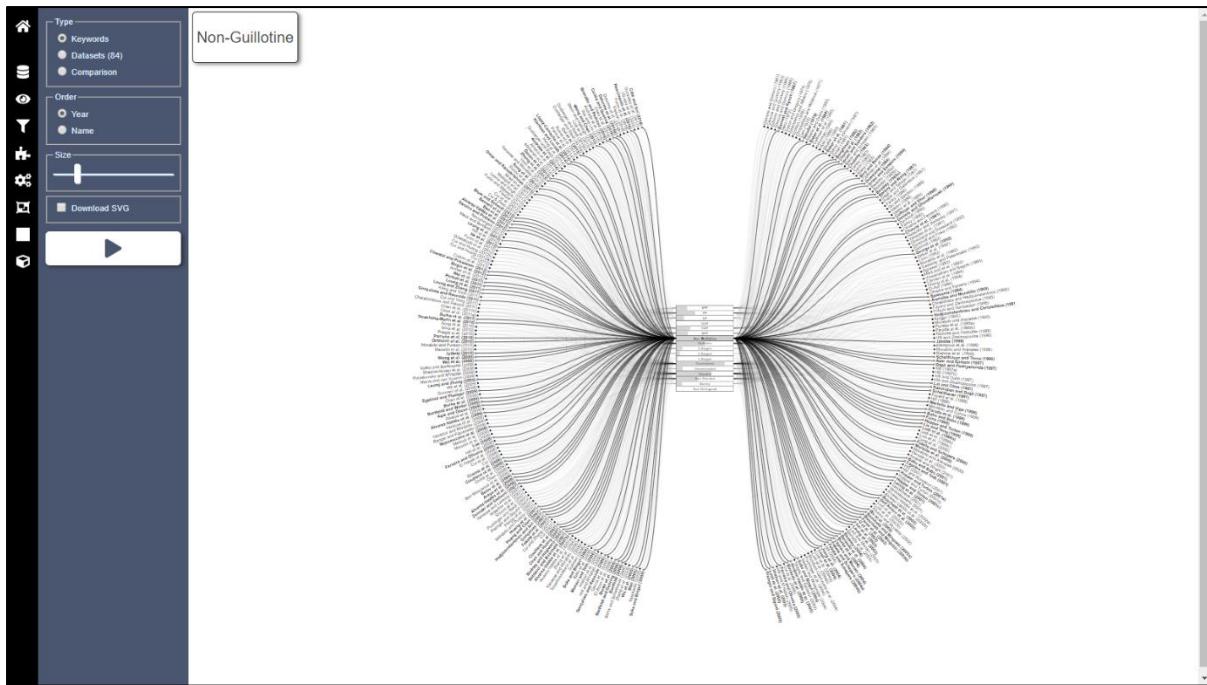


Figure 16. Type Keywords – Non-guillotine keyword selected.

Figure 17 illustrates the articles that use the dataset CGCUT.

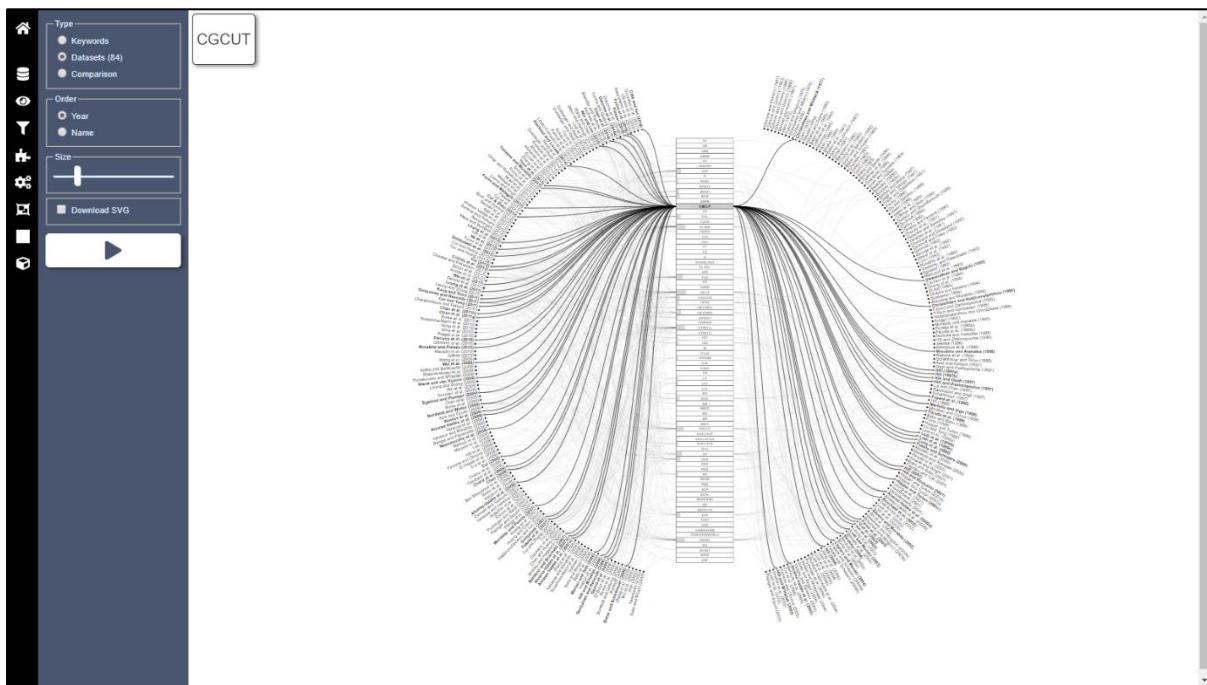


Figure 17. Type Datasets – CGCUT dataset selected.

Figure 18 highlights the comparisons made considering the article presented by Gonçalves and Resende [5]. This diagram highlights (black lines) the four articles that Gonçalves and Resende used to evaluate the results obtained by their heuristic and highlights (red line) that Kierkosz and Luczakone [6] used the results obtained in [5] for comparison.

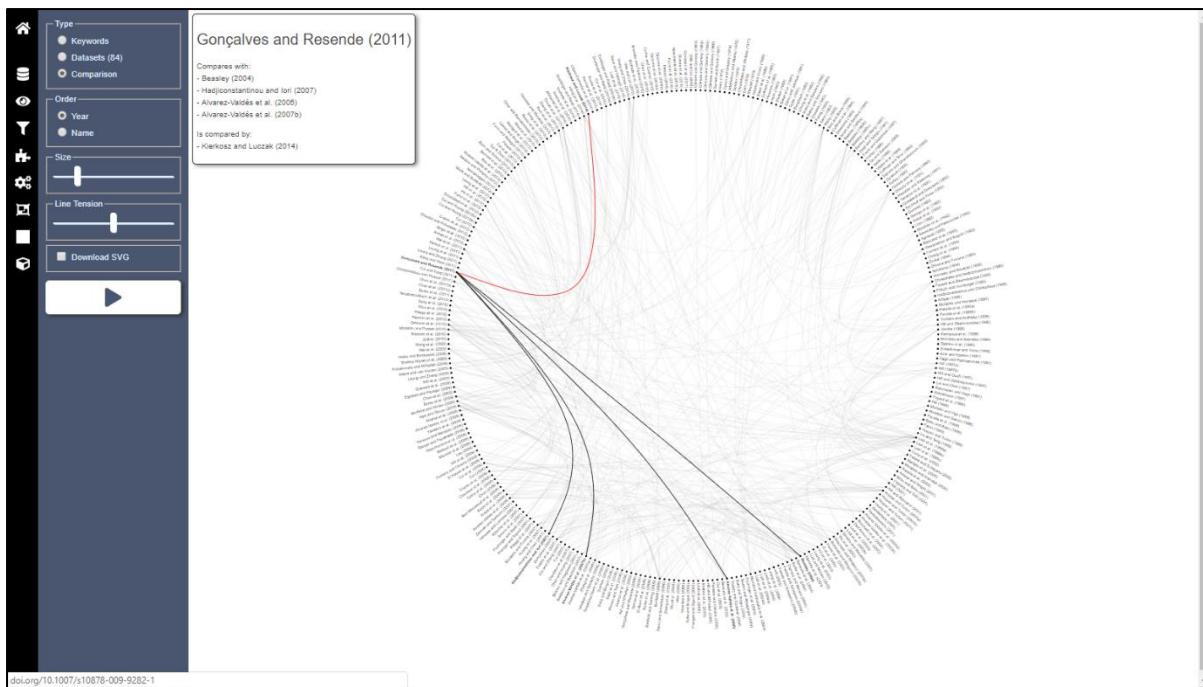


Figure 18. Type Comparison – Article Gonçalves and Resende (2011) selected.

4.2. Literature review - Quick view

Figure 19 illustrates the Quick view webpage and as in the previous diagrams, rolling the cursor over a keyword, dataset or article highlights the existent relations. Figure 20 illustrates the webpage with the dataset CGCUT selected, highlighting all articles that use this dataset.

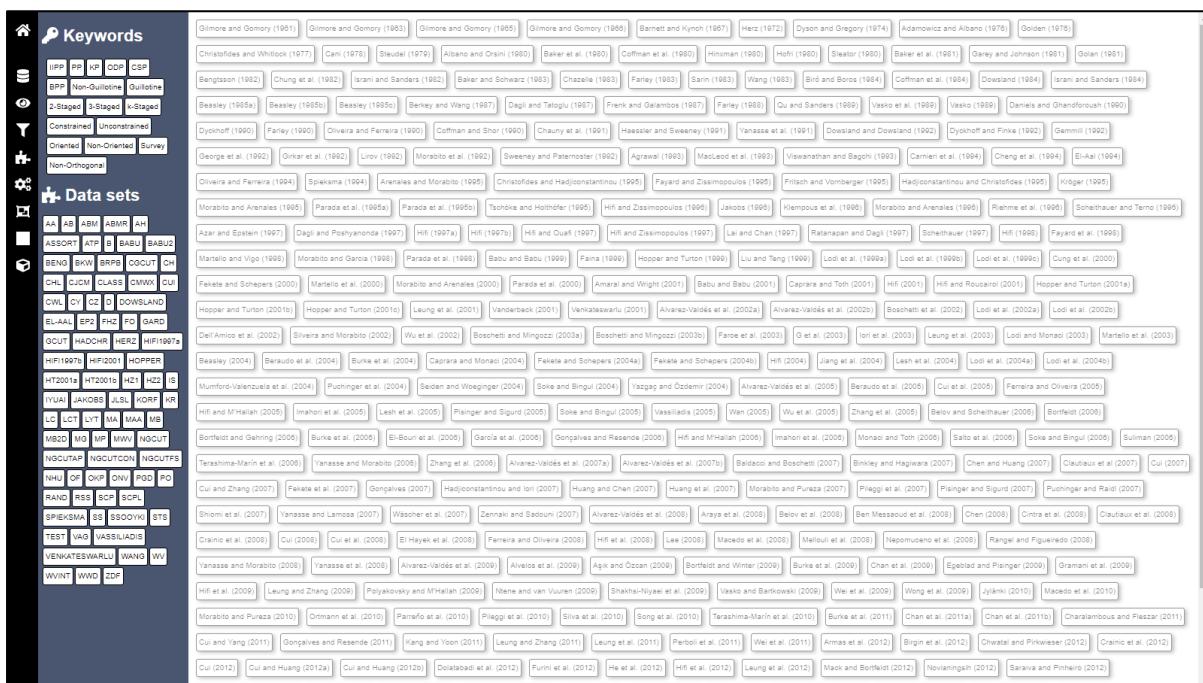


Figure 19. Quick view.

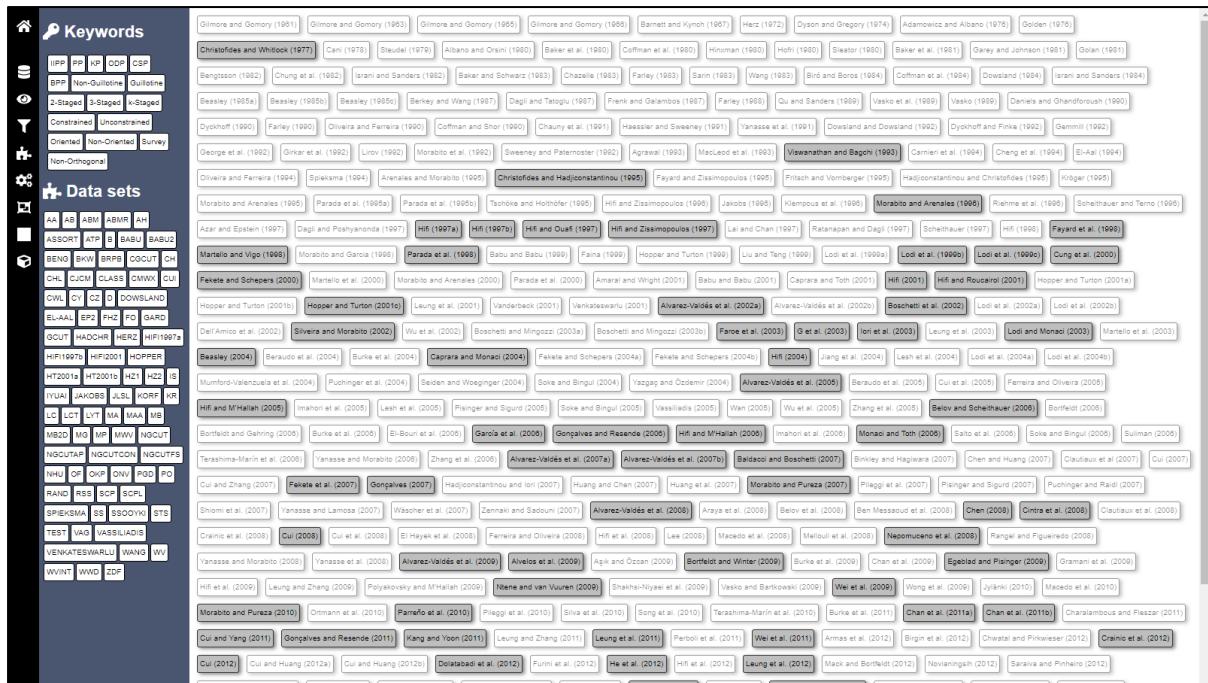


Figure 20. Quick view - CGCUT dataset selected.

4.3. Literature review - Filter

Figure 21 depicts the Filter webpage that allows to easily find articles using search words, keyword list (using logical operators is necessary) and decades selection. For each article that satisfies the search parameters, it is presented in a tabular format the publication year, the article reference, the keywords and the URL for access.

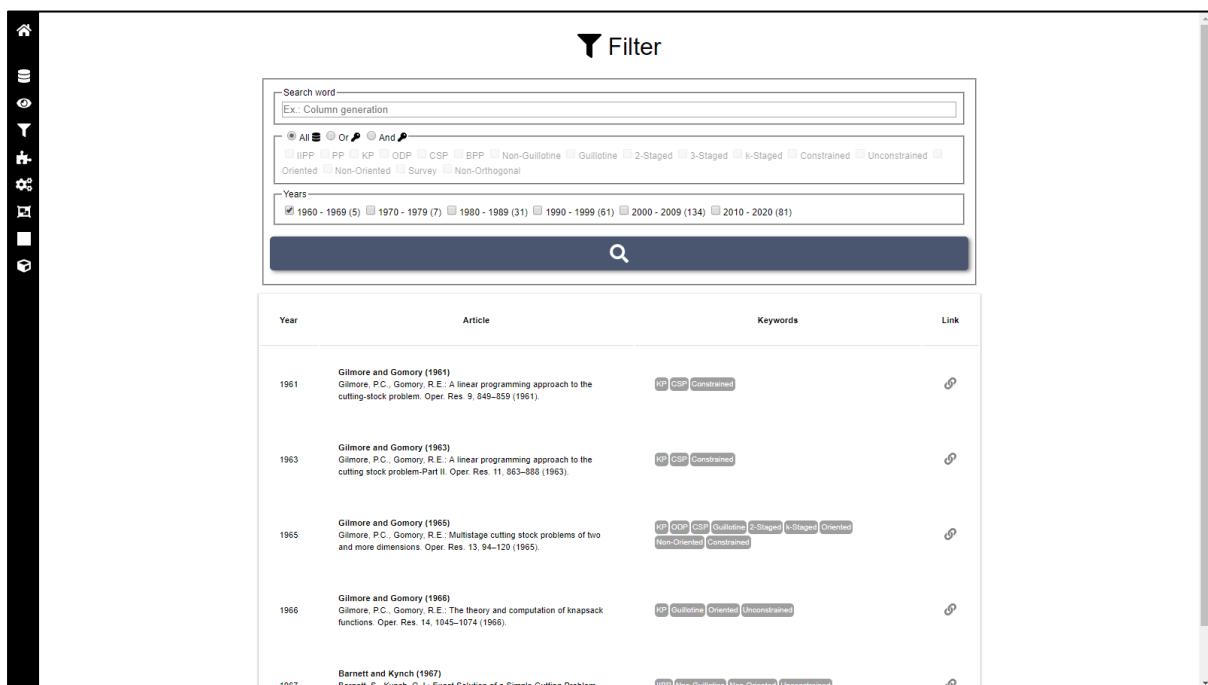
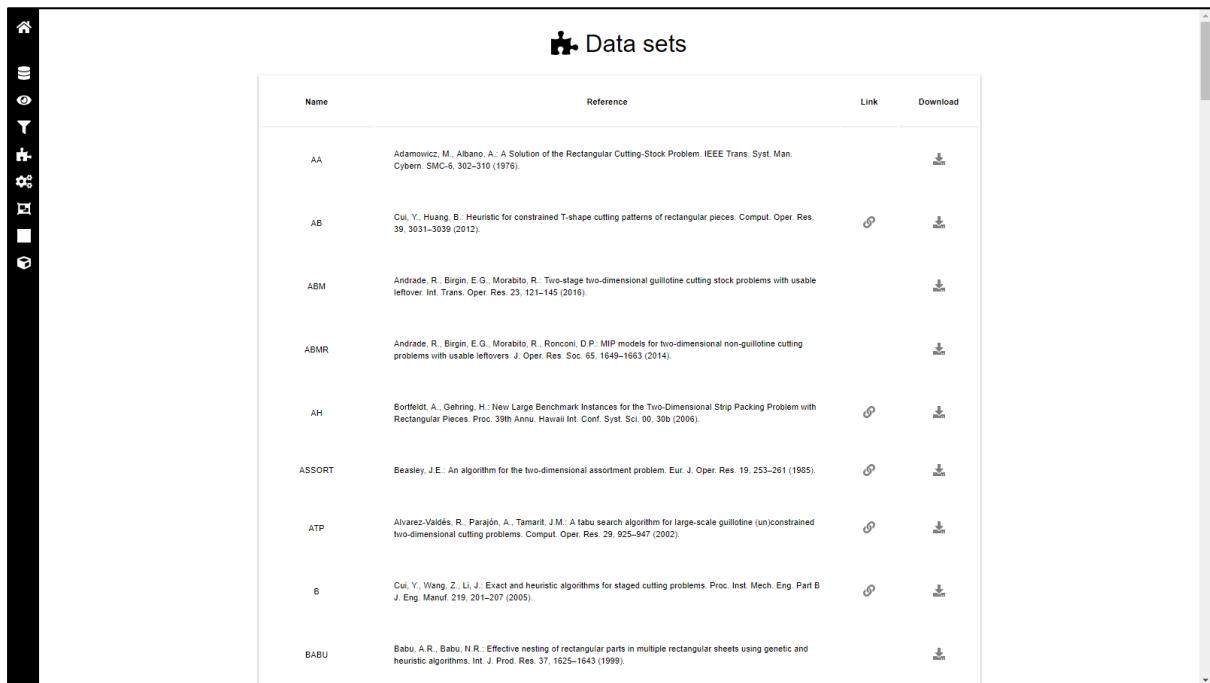


Figure 21. Filter.

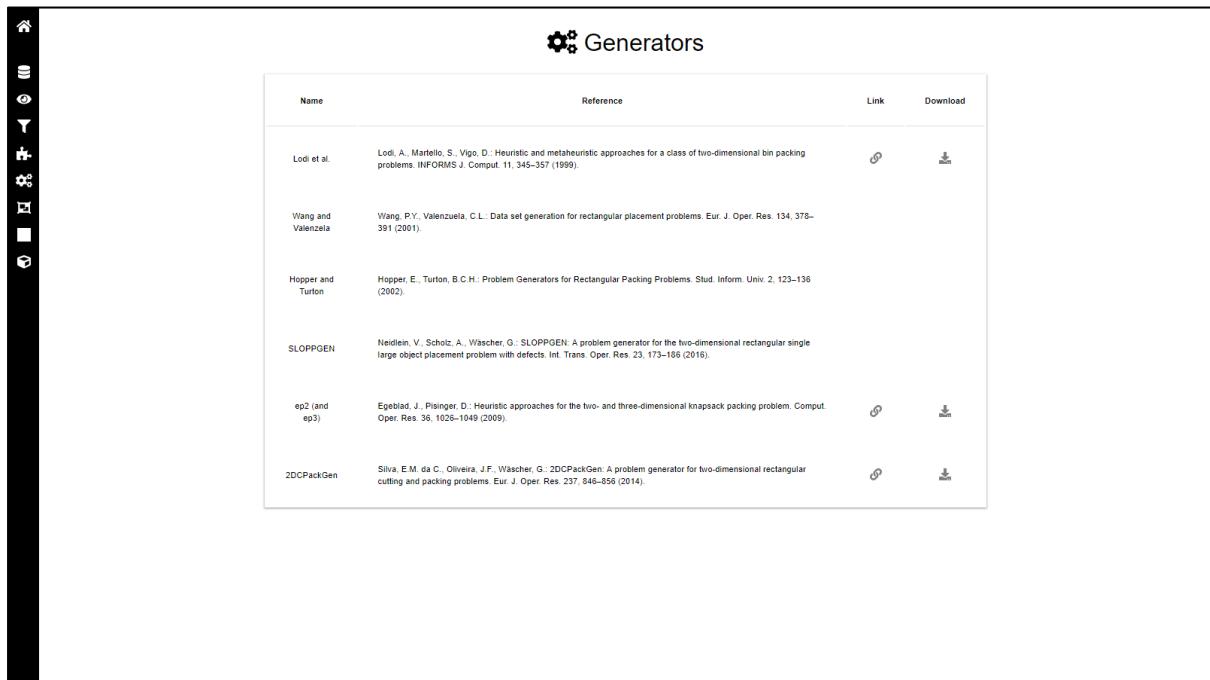
4.4. Lists of datasets and instance generators

Figure 22 and Figure 23 show the datasets and instance generators lists webpages that allows an easy access to these resources. The tables present for each entry (dataset or instance generator), the name, the reference, a link to the URL from which it was obtained and a link that allows to download the respective entry from our repository.



Name	Reference	Link	Download
AA	Adamowicz, M., Albano, A.: A Solution of the Rectangular Cutting-Stock Problem. <i>IEEE Trans. Syst. Man. Cybern. SMC-6</i> , 302–310 (1976).	🔗	⬇️
AB	Cui, Y., Huang, B.: Heuristic for constrained T-shape cutting patterns of rectangular pieces. <i>Comput. Oper. Res.</i> 39 , 3031–3039 (2012).	🔗	⬇️
ABM	Andrade, R., Birgin, E.G., Morabito, R.: Two-stage two-dimensional guillotine cutting stock problems with usable leftover. <i>Int. Trans. Oper. Res.</i> 23 , 121–145 (2016).	🔗	⬇️
ABMR	Andrade, R., Birgin, E.G., Morabito, R., Ronconi, D.P.: MIP models for two-dimensional non-guillotine cutting problems with usable leftovers. <i>J. Oper. Res. Soc.</i> 65 , 1649–1663 (2014).	🔗	⬇️
AH	Bortfeldt, A., Gehring, H.: New Large Benchmark Instances for the Two-Dimensional Strip Packing Problem with Rectangular Pieces. <i>Proc. 39th Annu. Hawaii Int. Conf. Syst. Sci.</i> 00 , 30b (2006).	🔗	⬇️
ASSORT	Beasley, J.E.: An algorithm for the two-dimensional assortment problem. <i>Eur. J. Oper. Res.</i> 19 , 253–261 (1985).	🔗	⬇️
ATP	Alvarez-Valdés, R., Parajón, A., Tamarit, J.M.: A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems. <i>Comput. Oper. Res.</i> 29 , 925–947 (2002).	🔗	⬇️
B	Cui, Y., Wang, Z., Li, J.: Exact and heuristic algorithms for staged cutting problems. <i>Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.</i> 219 , 201–207 (2005).	🔗	⬇️
BABU	Babu, A.R., Babu, N.R.: Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms. <i>Int. J. Prod. Res.</i> 37 , 1625–1643 (1999).	🔗	⬇️

Figure 22. Datasets list.



Name	Reference	Link	Download
Lodi et al.	Lodi, A., Martello, S., Vigo, D.: Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. <i>INFORMS J. Comput.</i> 11 , 345–357 (1999).	🔗	⬇️
Wang and Valenzuela	Wang, P.Y., Valenzuela, C.L.: Data set generation for rectangular placement problems. <i>Eur. J. Oper. Res.</i> 134 , 378–381 (2001).	🔗	⬇️
Hopper and Turton	Hopper, E., Turton, B.C.H.: Problem Generators for Rectangular Packing Problems. <i>Stud. Inform. Univ.</i> 2 , 123–136 (2002).	🔗	⬇️
SLOPPGEN	Neidlein, V., Scholz, A., Wäscher, G.: SLOPPGEN: A problem generator for the two-dimensional rectangular single large object placement problem with defects. <i>Int. Trans. Oper. Res.</i> 23 , 173–195 (2016).	🔗	⬇️
ep2 (and ep3)	Egebärd, J., Pisinger, D.: Heuristic approaches for the two- and three-dimensional knapsack packing problem. <i>Comput. Oper. Res.</i> 36 , 1026–1049 (2009).	🔗	⬇️
2DCPackGen	Silva, E.M. da C., Oliveira, J.F., Wäscher, G.: 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. <i>Eur. J. Oper. Res.</i> 237 , 846–856 (2014).	🔗	⬇️

Figure 23. Instance generators list.

5. Conclusion

We present a set of resources as we consider that they can be useful for researchers and practitioners whose interest lies in the study of cutting and packing problems. We have made available a wide set of datasets found in the literature, all converted into the same format to provide a consistent experience when researching for new solution methods. We, also, have made available three graphical user interfaces, one to visualise the instances and the other two to visualise the solutions obtained in a two-dimensional or three-dimensional view.

The last resource presented is a website, that allows easy access to all resource created. This website besides hosting the datasets, generators and GUI, contains a set of utilities for helping the analysis of the literature related to the two-dimensional cutting and packing problems. Noteworthy that an extension of this website can be developed to characterise other combinatorial problems with minor changes. We expect that these resources allow the researchers to focus on the actual research and not in the tool making as they occupy a great amount of time and resources. As we make these resources available, we hope for the contribution of others to improve the work made so far. A great amount of work can be undergone to improve these resources, we have identified a few, such as, unify the viewers into one tool, unify the JSON format and validate the JSON structure through a JSON schema⁷ (the three-dimensional viewer already uses a JSON schema to validate the JSON files). We envision a wide group of researchers discussing and contributing to improve these tools allowing to regain the focus in the research of solution methods.

References

- [1] G. Wäscher, H. Haußner, and H. Schumann, “An improved typology of cutting and packing problems,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [2] E. Hopper and B. C. H. Turton, “Problem Generators for Rectangular Packing Problems,” *Stud. Inform. Univ.*, vol. 2, pp. 123–136, 2002.
- [3] J. F. Oliveira and J. S. Ferreira, “An improved version of Wang’s algorithm for two-dimensional cutting problems,” *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 256–266, 1990.
- [4] S. Martello, D. Pisinger, and D. Vigo, “The Three-Dimensional Bin Packing Problem,”

⁷ <https://json-schema.org/>

Oper. Res., vol. 48, no. 2, pp. 256–267, 2000.

- [5] J. F. Gonçalves and M. G. C. Resende, “A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem,” *J. Comb. Optim.*, vol. 22, no. 2, pp. 180–201, 2011.
- [6] I. Kierkosz and M. Luczak, “A hybrid evolutionary algorithm for the two-dimensional packing problem,” *Cent. Eur. J. Oper. Res.*, vol. 22, no. 4, pp. 729–753, 2014.

Part III – Heuristics

Adaptive Sequence-based Heuristic for Two-Dimensional Non-Guillotine Packing Problems

Abstract We present heuristics for two related two-dimensional non-guillotine packing problems. The first problem aims to pack a set of items into the minimum number of larger identical bins, while the second aims to pack the items that generates most value into only one bin. Our approach successively creates sequences of items that defines a packing order considering the knowledge obtained from sequences generated previously. Computational experiments demonstrated that the proposed heuristics are very effective in terms of solution quality and with small computing times.

Keywords: Two-dimensional, Rectangular, Non-guillotine, Knapsack Problem, Bin Packing, Heuristics

1. Introduction

We present heuristics to solve two related non-guillotine packing problems both considering a finite set of m rectangular items types with associated length (l_i), height (h_i), value (v_i), and demand (d_i). In the first problem, the set of items must be packed into the minimum number of identical bins, while the second one, aims to maximize the value of the items packed into one bin. The items must be packed orthogonally and cannot be rotated. All bins are rectangular and have identical length (L) and height (H).

Giving that, the proposed heuristics deal with each demanded item individually, the first problem, following the typology of Wäscher *et al.* [1], is classified as Single Bin Size Bin Packing Problem (SBSBPP) while the second one is classified as Single Knapsack Problem (SKP).

The remaining of the paper is organized as follows. Section 2 gives an overview of the solutions approaches proposed in the literature to solve both problems. In Section 3, a description of our heuristics, hereafter denoted as Adaptive Sequence-based Heuristics (ASH), is presented, and in Section 4, computational results are reported comparing our approach

to other solution methods from the literature. Finally, conclusions and future work directions are given in Section 5.

2. Literature Review

Since the work of Gilmore and Gomory [2] the interest on cutting and packing problems has been growing, due to its complexity (NP-hard, see Garey and Johnson [3]) and great practical applicability (see Singh and Jain [4]).

The next two sub-sections present the most relevant solution methods proposed for the Single Bin Size Bin Packing Problem (SBSBPP) and Single Knapsack Problem (SKP), respectively.

2.1. Single Bin Size Bin Packing Problem

Berkey and Wang [5] studied the performance of heuristics for the SBSBPP adapted from heuristics proposed in the literature for the Open Dimensional Problem (ODP, see Wäscher *et al.* [1]).

Regarding the lower bounds, Martello and Vigo [6] showed that the Continuous Lower Bound ($CLB = \lceil \frac{\sum_{i=1}^m l_i h_i}{LH} \rceil$) for the SBSBPP has a worst-case performance ratio of $\frac{1}{4}$ and presented new lower bounds that are used in a Branch-and-Bound algorithm. In Boschetti and Mingozzi [8] new lower bounds are proposed and in [9] the same authors presented a heuristic called HBP, for the SBSBPP. HBP generates solutions considering the current allocation method and updating the value of the items at the end of each iteration.

Lodi *et al.* [7] presented heuristics to solve the 2D (non-)oriented (non-)guillotine SBSBPP and a Unified Tabu Search Framework that is adaptable for each specific problem changing uniquely the inner heuristic to explore the neighbourhood.

A heuristic algorithm based on Guided Local Search (GLS, see Voudouris and Tsang [11]) for the 3D SBSBPP was proposed by Faroe *et al.* [10]. This approach was adapted by the authors for the 2D case providing the same depth for bins and items. The algorithm starts with an upper bound on the number of available bins and iteratively decreases this number for the next cutting plan generation. This process is repeated until the time limit is reached or the current solution is the same as the calculated lower bound.

Monaci and Toth [12] presented the Set-Covering Heuristic (SCH) formulating the problem as a Set-Covering Problem. SCH generates, through heuristic procedures, a large set of columns to define the Set-Covering instance, then the problem is solved by means of a Lagrangean-based heuristic.

A Greedy Randomized Adaptive Search Procedure (GRASP, see Feo and Resende [14]) with Variable Neighbourhood Descent (VND, see Hansen and Mladenovic [15]) for the 2D and 3D SSBPP was proposed by Parreño *et al.* [13].

In Blum and Schmid [16] a hybrid algorithm called EA-LGFi is presented. In EA-LGFi an evolutionary approach is used to generate the input sequences to the LGFi heuristic proposed by Wong *et al.* [17] in order to perform the placement of the items.

2.2. Single Knapsack Problem

Exact approaches to solve the SKP have been proposed in the literature by several authors. Beasley [18] presented a Branch-and-Bound algorithm that uses an upper bound obtained through the Lagrangean relaxation of a 0-1 integer linear programming formulation of the problem and a Subgradient Optimization procedure to minimize its value.

An integer programming formulation where binary variables specify the item position relatively to another item was proposed by Scheithauer and Terno [19]. The authors presented a Branch-and-Bound in which Subgradient Optimization is used to minimize the upper bound obtained through the Lagrangean Relaxation of the problem. The authors perform reduction tests that limit the size of the tree, therefore, reducing the computational efforts required by the algorithm.

Fekete and Schepers [20][21][22] presented a tree-search algorithm for the d -dimensional Knapsack Problem using a graph-theoretical characterization of packings. In this characterization, if no overlapping occurs in both x and y axis projection graphs, the pattern is feasible.

Boschetti *et al.* [23] presented new upper bounds obtained through the relaxation of their integer programming formulation of the problem.

Four exact algorithms based on the relaxation of SKP given by the one-dimensional case are presented in Caprara and Monaci [24]. Exact algorithms were also proposed for the Knapsack problem in which only guillotine cuts are allowed, without a limitation on the number of stages, in Dolatabadi *et al.* [25] and more recently in Fleszar [26] and in Furini *et al.* [27].

Heuristics methods have, also, been proposed to solve the SKP. Lai and Chan [28] presented a heuristic based on the Simulated Annealing using an ordered list of items that encodes the order in which the items will be packed into the bin through a placement algorithm. The placement algorithm packs each item, in turn, without overlapping, into the empty rectangular space (ERS) that is closest to the bin bottom-left corner. This heuristic keeps track of the ERS through a called Difference Process that creates interval lists after each item packing. In Leung *et al.* [42] a Simulated Annealing algorithm is also used, combined with a greedy strategy.

Leung *et al.* [29] performed a comparison of the Genetic Algorithm with bottom-left approach proposed for the Strip Packing Problem by Jakobs [30] and the Simulated Annealing with Difference Process proposed by Lai and Chan [28]. In Leung *et al.* [31] a pure Genetic Algorithm and a hybrid approach (Genetic Algorithm with Simulated Annealing) called MSAGA are compared, aiming to verify if the latter could prevent the early convergence observed in the pure approach. The decoder, i.e., placement algorithm, uses the Difference Process.

A Genetic Algorithm based on a new non-linear mathematical formulation was presented in Beasley [32]. In Alvarez-Valdés *et al.* [33] a GRASP is proposed and in Alvarez-Valdés *et al.* [34] a Tabu Search (see Glover [35]) to solve the SKP is presented.

Hadjiconstantino and Iori [36] presented a greedy heuristic and a hybrid Genetic Algorithm. The greedy heuristic called HC_{HV} packs the items at the bottom-left position possible, alternatively on top of each other or side by side until no more items fit in the current direction. In the genetic approach, the items are placed at the position where the highest fraction of its perimeter touches edges, either of another item or bin boundaries.

A hybrid Genetic Algorithm based on random keys was proposed in Gonçalves [37]. The genetic part of this heuristic is responsible to generate and evolve the sequence of items that defines the packing order. The placement method makes use of the Difference Process proposed by Lai and Chan [28] to keep track of the ERS created after each placement. In

Gonçalves and Resende [39] a parallel implementation of a multi-population Genetic Algorithm where the chromosome encodes the items sequence and the associated placement rule (bottom-left or left-bottom) is presented. As in the hybrid Genetic Algorithm based on random keys, the Difference Process is used again to keep track of the ERS that are created after placing an item.

Bortfeldt and Winter [38] also presented a Genetic Algorithm approach to solve the (un)constrained (non-)guillotine SKP and Single Large Object Placement Problem (SLOPP, see Wäscher *et al.* [1]). In Kierkosz and Luczak [43] an Evolutionary Algorithm is presented, using as a placement algorithm a tree-search that places the items using a bottom-left strategy.

Wei *et al.* [40] presented a Tabu Search using the concept of Skyline representation which is a sequence of line segments that expresses the rectilinear contour of the current packed items.

A two-stage heuristic for solving the SKP is presented in He *et al.* [41]. In the first phase, a solution is generated placing the items in the empty space with the highest fit degree (calculated with the number of touching edges and a smooth degree that considers the number of empty spaces remaining if the item is placed at this position), then the solution is improved through a partial tree-search.

3. Adaptive Sequence-based Heuristic (ASH)

We propose a Multi-start heuristic (see Martí *et al.* [44]) that iteratively creates a new sequence of items used to define the packing order.

The main concept behind the proposed heuristics is that if a good solution was packed using some ordering (S_{base}), it is possible that a better solution exists changing the order of few items, creating a new packing sequence $S_{current}$. If no improvement is obtained with this new ordering, it may be the case that an ordering with more changes can lead to a better solution. So, we incrementally allow more changes to the base ordering. When an ordering generates a new best solution, this will replace the base ordering to be used in the next iterations. The main steps of the proposed heuristics are shown in Algorithm 1.

Algorithm 1. ASH main steps.

```

 $S_{base} \leftarrow$  Items ordered by efficiency with value as tiebreaker
 $\alpha \leftarrow \alpha_{min}$ 
Generate a new solution with sequence  $S_{base}$ 
While stopping criteria are not met do
     $S_{current} \leftarrow$  Generate a new sequence of items based on  $S_{base}$  and  $\alpha$ 
    Generate a new solution with sequence  $S_{current}$ 
    If a new best solution is found then
         $S_{base} \leftarrow S_{current}$ 
         $\alpha \leftarrow \alpha_{min}$ 
    Otherwise
         $\alpha \leftarrow \min\{\alpha + \alpha_{inc}, \alpha_{max}\}$ 

```

Starting with S_{base} , the sequence of items ordered by efficiency $e_i = \frac{v_i}{(l_i \times h_i)}$ (see Alvarez-Valdés *et al.* [33]) using v_i as a tiebreaker, at each iteration, a new sequence $S_{current}$ is generated using the current probability α . The sequences are generated (see Algorithm 2) based on the algorithm proposed by Lesh *et al.* [45] which creates a new sequence adding with a probability of α , one element at a time from the input sequence to the new sequence.

Algorithm 2. Sequence generator.

```

Input: sequence  $In$ , probability  $\alpha$ 
 $Out \leftarrow \emptyset$ 
 $n \leftarrow |In|$ 
for  $i \leftarrow 1, \dots, n$  do
     $j \leftarrow 1$ 
     $Out_i \leftarrow \emptyset$ 
    while  $Out_i = \emptyset$  do
        if  $\alpha \leq$  generated random value then
             $Out_i \leftarrow In_j$ 
             $In \leftarrow In \setminus \{In_j\}$ 
         $j \leftarrow (j \bmod |In|) + 1$ 
return  $Out$ 

```

If the solution generated with $S_{current}$ is the best one so far, this ordering replaces S_{base} and α is set to its minimum value α_{min} . Otherwise, S_{base} is inalterable and α is updated to $\min\{\alpha + \alpha_{inc}, \alpha_{max}\}$.

When a new best solution is found, we seek to intensify the search near to this solution, which is accomplished by setting α to its minimum value, aiming to generate sequences that are very similar to the base. While higher α allows to diversify the search space producing sequences that differ incrementally more from the base sequence.

Following the ordering of $S_{current}$, the items are packed into the bin, one at a time on the empty rectangle space (ERS) with the smallest area with enough space to fit the item. To

prevent the creation of very small empty spaces, after packing an item, we try to pack items that fit exactly into existing ERS.

We keep track of the ERS resulting from the packing of items using the Difference Process (Lai and Chan [28]). This process, first, places the box inside the given ERS, then generates the new ERSs that result from the intersection of the box with the existing ERS and removes intersected ERSs. The last phase removes the ERSs that are infinitely thin or are totally inscribed by other ERSs. The Difference Process is illustrated in Figure 1, in which the darker rectangles depict the available ERS at the beginning of the process (a) and at the end of each item placement (b and c).

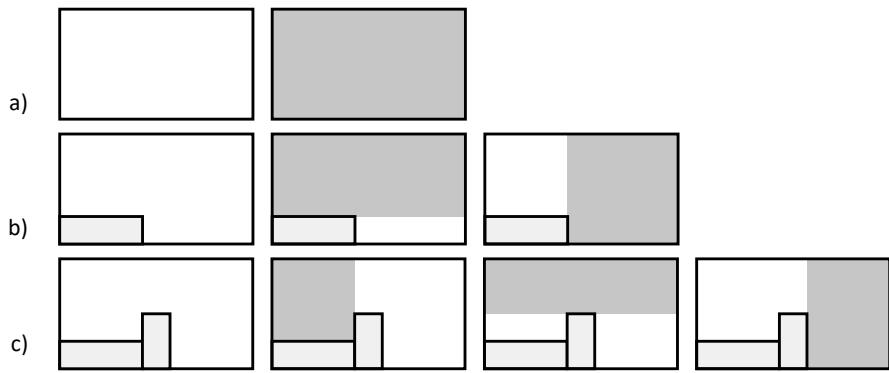


Figure 1. Difference Process.

The main difference between solving SSBPP and SKP instances is that for SSBPP, new patterns are created and added to the solution until all items are packed, while for the SKP, a solution is created considering only one bin.

ASH iterates until a maximum number of iterations has been performed or the optimality is guaranteed (if the solution value is equal to the Continuous Lower Bound (*CLB*) for the SSBPP, and if all the items are packed inside the bin for the SKP).

4. Computational Results

The proposed heuristic was implemented in C and the tests were run on a computer with an Intel Core i7-4800MQ at 2.70 GHz with 8 Gb RAM and operating system Linux Ubuntu 18.04.

The next two sub-sections present the computational results for the Single Bin Size Bin Packing Problem (SSBPP) and Single Knapsack Problem (SKP), respectively. For both problems, the values of α_{min} , α_{max} , and α_{inc} were set through experiments carried out

considering different options, and the datasets considered correspond to the ones used by the algorithms that were used for comparison.

4.1. Single Bin Size Bin Packing Problem

We have considered to evaluate the results of our heuristic for solving the SSBPP, the set of instances, usually referred to as CLASS, generated by Berkey and Wang [5] (CLASS 1 to 6) and by Martello and Vigo [6] (CLASS 7 to 10). This set is divided into 10 classes of 50 instances. Each class contains 5 subsets composed of 10 instances for each value of $m \in [20, 40, 60, 80, 100]$, with $m = \sum_i d_i$. The bin dimensions range from 10×10 to 300×300 .

Each instance was run only once, generating at most 2000 cutting plans with α_{min} , α_{max} , and α_{inc} set to 0.1, 0.65, and 0.002, respectively.

Table 1 presents the results solving all the instances of the dataset CLASS by the heuristics HBP [9], GLS [10], SCH [12], GRASP/VND [13], EA_LGFi [16], and in the last column by ASH. The first line denotes the number of bins needed for the 500 instances, and the average and maximum time to solve a subset. The results from HBP and GLS were retrieved directly from Boschetti and Mingozi [9] and those of SCH, GRASP/VND and EA_LGFi were retrieved directly from Blum and Schmid [16].

Table 1. Summary of the results obtained.

Algorithm	HBP	GLS	SCH	GRASP/VND	EA_LGFi	ASH
Number of bins	7275	7284	7243	7241	7239	7269
Average time	21.01	1000	7.89	2.20	1.77	0.04
Maximum time	114.07	1000	55.77	12.00	27.33	0.11

Observing Table 1, the best results are clearly obtained by SCH, GRASP/VND and EA_LGFi. The results obtained by those approaches and by ASH solving Berkey and Wang instances are given in Table 2, and in Table 3 the results solving Martello and Vigo instances.

The first column of Table 2 and Table 3 gives the class number, the second column denotes the number of items of each subset, then for each of the heuristics the number of bins and the average time, in seconds, needed to solve each subset.

Table 2. Results for Berkey and Wang instances.

CLASS	m	SCH		GRASP/VND		EA_LGFi		ASH	
		Bins	t (s)	Bins	t (s)	Bins	t (s)	Bins	t (s)
1	20	71	0.06	71	0.00	71	0.00	71	0.01
	40	134	2.42	134	0.00	134	0.00	134	0.02
	60	200	7.26	200	4.50	200	0.01	200	0.04
	80	275	4.63	275	1.50	275	0.00	275	0.07
	100	317	5.21	317	0.00	317	0.00	317	0.08
2	20	10	0.06	10	0.00	10	0.00	10	0.00
	40	19	0.67	19	0.00	19	0.00	19	0.00
	60	25	0.07	25	0.00	25	0.00	25	0.00
	80	31	0.07	31	0.00	31	0.00	32	0.01
	100	39	0.79	39	0.00	39	0.00	39	0.00
3	20	51	0.07	51	0.00	51	0.02	51	0.01
	40	94	2.66	94	3.00	94	0.01	94	0.02
	60	139	6.21	139	4.60	139	0.27	140	0.06
	80	189	8.80	189	4.10	189	20.68	192	0.08
	100	223	12.80	223	4.90	224	26.17	225	0.11
4	20	10	0.06	10	0.00	10	0.00	10	0.00
	40	19	0.07	19	0.00	19	0.00	19	0.00
	60	25	6.15	25	3.00	23	12.18	25	0.02
	80	32	10.35	31	1.90	31	0.00	32	0.03
	100	38	4.72	38	1.50	37	0.00	38	0.03
5	20	65	0.06	65	0.00	65	0.00	65	0.00
	40	119	1.98	119	0.00	119	0.03	119	0.02
	60	180	1.93	180	1.50	180	0.14	181	0.06
	80	247	20.66	247	9.00	247	0.03	247	0.09
	100	282	18.50	282	5.20	284	27.33	287	0.11
6	20	10	0.06	10	0.00	10	0.00	10	0.00
	40	17	6.85	17	3.00	17	0.03	18	0.03
	60	21	0.66	21	0.10	21	0.00	22	0.01
	80	30	0.23	30	0.00	30	0.00	30	0.00
	100	34	6.29	34	3.00	32	0.58	34	0.00

ASH could not provide better results than those obtained by the best approaches already mentioned, but it is extremely effective considering the execution time presented.

Noteworthy that the gap between the average and maximum time presented by ASH is very tight when compared with the other approaches (e.g., 1.77 seconds of average time to 27.33 seconds for the EA_LGFi). This tight gap makes ASH a robust heuristic with a predictable execution time.

Our approach is very fast, simple to implement and can be an effective approach to solve the SSBPP or to be used in a bounding scheme on more complex solution methods.

Table 3. Results for Martello and Vigo instances.

CLASS	<i>m</i>	SCH		GRASP/VND		EA_LGFi		ASH	
		Z	time(s)	Z	time(s)	Z	time(s)	Z	time(s)
7	20	55	0.13	55	0.00	55	0.00	55	0.01
	40	111	3.02	111	3.00	111	0.01	112	0.03
	60	158	8.85	159	4.50	159	0.00	159	0.06
	80	232	54.79	232	12.00	232	0.00	232	0.08
	100	271	25.06	271	3.10	271	0.01	273	0.10
8	20	58	0.06	58	0.00	58	0.03	58	0.01
	40	113	0.96	113	1.50	113	0.00	113	0.03
	60	162	9.05	161	4.20	161	0.02	162	0.06
	80	224	11.60	224	1.60	224	0.00	226	0.09
	100	279	47.13	278	6.10	277	0.25	278	0.00
9	20	143	0.06	143	0.00	143	0.00	143	0.01
	40	278	0.07	278	0.00	278	0.00	278	0.03
	60	437	0.07	437	0.10	437	0.00	437	0.07
	80	577	0.08	577	0.00	577	0.00	577	0.10
	100	695	0.11	695	0.00	695	0.00	695	0.00
10	20	42	0.12	42	0.00	42	0.02	43	0.01
	40	74	0.11	74	0.00	74	0.00	74	0.02
	60	101	8.89	100	4.50	101	0.71	102	0.05
	80	128	38.26	129	9.40	128	0.06	130	0.08
	100	159	55.77	159	9.20	160	0.08	161	0.11

4.2. Single Knapsack Problem

To evaluate the performance of ASH for the SKP, we have considered the following benchmark datasets, NGCUT (Beasley [18]), HADCHR (Hadjiconstantinou and Christofides [46]), WANG (Wang [47]), CGCUT (Christofides and Whitlock [48]), OKP (Fekete *et al.* [49]), LC (Lai and Chan [28]), JAKOBS (Jakobs [30]), LCT (Leung *et al.* [31]), and HT (Hopper and Turton [50]).

The details of the datasets are given in Table 4 which presents, in order of appearance, the dataset name, the number of instances, the range of the bins and items dimensions, and the range of the number of demanded items (*m*). The first five datasets are referred to as Problems from literature and the remaining four referred to as Zero-waste problems.

The results obtained by ASH are shown in Table 5 and Table 6. Each instance was run only once, generating at most 2000 solutions with α_{min} , α_{max} , and α_{inc} set to 0.1, 0.9, and 0.005 respectively.

Table 4. Datasets.

Dataset	Number of instances	Bin $w \times h$	Items $w \times h$	m
NGCUT	12	[10-30]x[10-30]	[1-30]x[1-30]	[7-22]
HADCHR	2	30x30	[1-22]x[4-21]	[7-15]
WANG	1	70x40	[9-33]x[11-43]	42
CGCUT	1	40x70	[9-33]x[11-43]	62
OKP	5	100x100	[1-100]x[1-99]	[30-97]
LC	3	400x[200-400]	[30-200]x[30-150]	[10-20]
JAKOBS	5	[65-120]x[45-80]	[5-40]x[4-36]	[20-30]
LCT	2	[150-160]x[110-120]	[6-48]x[4-40]	[40-50]
HT	21	[20-160]x[20-240]	[1-72]x[1-113]	[16-197]

The content of Table 5 and Table 6 is the following. The columns identify, in order of appearance, the dataset name, the instance number, the optimal solution, and the result obtained by GRASP [33], Tabu Search [34], the Parallel Multi-population Genetic Algorithm (MPGA) [39], and finally, by ASH. For each dataset, the last rows present the number of optimal solutions obtained and the average time needed to solve each instance. Table 6 includes two more rows, the maximum time spent to solve one instance and the average gap between the optimal solution and the solution obtained.

Table 5. Computational results - Problems from literature.

Set	#	Z*	GRASP	TABU	MPGA	ASH
NGCUT	1	164	164	164	164	164
	2	230	230	230	230	230
	3	247	247	247	247	247
	4	268	268	268	268	268
	5	358	358	358	358	358
	6	289	289	289	289	289
	7	430	430	430	430	430
	8	834	834	834	834	834
	9	924	924	924	924	924
	10	1452	1452	1452	1452	1452
	11	1688	1688	1688	1688	1688
	12	1865	1865	1865	1865	1865
		Optimums	12	12	12	12
		Avg. time (s)	0.07	0.03	0.01	0.01
HADCHR	3	1178	1178	1178	1178	1178
	11	1270	1270	1270	1270	1270
			Optimums	2	2	2
			Avg. time (s)	0.00	0.00	0.01
WANG		2726	2726	2726	2726	2721
			Optimums	1	1	0
			Avg. time (s)	0.77	0.11	0.02
						0.01
OCGCUT	3	1860	1860	1860	1860	1860
			Optimums	1	1	1
			Avg. time (s)	0.39	0.06	0.02
OKP	1	27718	27589	27718	27718	27486
	2	22502	21976	22502	22502	22119
	3	24019	23743	24019	24019	24019
	4	32893	32893	32893	32893	32893
	5	27923	27923	27923	27923	27923
			Optimums	2	5	3
			Avg. time (s)	2.03	1.25	0.03

Table 5 presents the results obtained by solving the datasets on Problems from the literature. The results show that both TABU and MPGA obtained all optimal solutions and that ASH obtained results close to those obtained by GRASP. Although execution time is not easy to compare, our time remains constantly small on all datasets, while the other approaches it increases greatly as the problem size grows.

The results for the Zero-waste problems are given in Table 6. The results show that ASH, when compared with the GRASP, obtains a higher number of optimal solutions, and only for one dataset, the average gap obtained is higher.

Table 6. Computational results - Zero-waste problems.

Set	#	Z*	GRASP	TABU	MPGA	ASH
LC	1	80000	80000	80000	80000	80000
	2	79000	79000	79000	79000	79000
	3	160000	154600	154600	154600	154600
	Optimums		2	3	3	2
	Avg. time (s)		1.37	0.13	1.26	0.01
	Max. time (s)		4.12	0.38	3.10	0.02
	Avg. GAP (%)		1.13	0	0	1.13
JAKOBS	1	5600	5447	5600	5600	5600
	2	5600	5455	5512	5540	5263
	3	5400	5328	5400	5400	5400
	4	4050	3978	4050	4050	4050
	5	2925	2871	2925	2925	2871
	Optimums		0	4	4	3
	Avg. time (s)		12.68	4.54	4.29	0.02
LCT	Max. time (s)		15.44	16.88	11.52	0.04
	Avg. GAP (%)		2.06	0.21	0.21	1.57
	1	16500	15856	16280	16340	15876
	2	19200	18628	19044	19116	18516
	Optimums		0	0	0	0
	Avg. time (s)		111.39	58.16	19.45	0.05
	Max. time (s)		132.26	63.95	23.71	0.09
HT	Avg. GAP (%)		3.44	1.07	0.70	3.67
	1	400	400	400	400	400
	2	400	386	400	400	386
	3	400	400	400	400	400
	4	600	590	600	600	594
	5	600	597	600	600	600
	6	600	600	600	600	600
	7	1800	1765	1800	1800	1773
	8	1800	1755	1800	1796	1758
	9	1800	1774	1800	1800	1764
	10	3600	3528	3580	3591	3536
	11	3600	3524	3564	3588	3542
	12	3600	3544	3580	3594	3560
	13	5400	5308	5342	5396	5328
	14	5400	5313	5361	5400	5319
	15	5400	5312	5375	5392	5338
	16	9600	9470	9548	9582	9467
	17	9600	9453	9448	9595	9501
	18	9600	9450	9565	9582	9506
	19	38400	37661	38026	38146	37771
	20	38400	37939	38145	38374	38074
	21	38400	37745	37867	38254	37866
	Optimums		3	9	9	4
	Avg. time (s)		612.00	572.30	118.30	0.49
	Max. time (s)		3760.14	5615.75	808.03	2.64
	Avg. GAP (%)		1.50	0.47	0.13	1.24

The average time of ASH maintains very low, while for the three other approaches grows greatly needing in some cases minutes to solve one instance.

Considering the other approaches, ASH is much simpler to implement and to parametrize (i.e., maximum number of iterations, and minimum, maximum and increment of the probability α). GRASP needs a Restricted Candidate List and improvement methods, Tabu Search requires neighbourhood structures and memory strategies, and Genetic Algorithms needs chromosomes and an evolutionary process. The results show that this heuristic, although simple, can generate high-quality solutions using small computing times.

5. Conclusion

We have developed the ASH to solve the non-guillotine Single Bin Size Bin Packing Problem and the non-guillotine Single Knapsack Problem. The ASH iteratively creates new sequences that define the packing order, incorporating knowledge from the previous packing order and during the packing procedure the Difference Process is used to keep track of the ERS resulting from packing the items.

Extensive computational experiments have been performed for both problems with well-known problem instances from the literature. The computational results show that our approach is competitive with other proposed solution methods for the problems considered. Noteworthy the implementation simplicity and the little parameterization required by this approach. We intend to extend the proposed method to the solution of other cutting and packing problems.

References

- [1] G. Wäscher, H. Haußner, and H. Schumann, “An improved typology of cutting and packing problems,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [2] P. C. Gilmore and R. E. Gomory, “A linear programming approach to the cutting-stock problem,” *Oper. Res.*, vol. 9, no. 6, pp. 849–859, 1961.
- [3] M. R. Garey and D. S. Johnson, “A Guide to the Theory of NP-Completeness,” *Mathematical Sciences*. W.H. Freeman and Company, San Francisco, 1979.
- [4] K. Singh and L. Jain, “Industrial Scope of 2D packing problems,” *Natl. J. Syst. Inf. Technol.*, vol. 2, no. 2, pp. 224–237, 2009.

- [5] J. O. Berkey and P. Y. Wang, "Two-Dimensional Finite Bin-Packing Algorithms," *J. Oper. Res. Soc.*, vol. 38, no. 5, p. 423, 1987.
- [6] S. Martello and D. Vigo, "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," *Manage. Sci.*, vol. 44, no. April 2015, pp. 388–399, 1998.
- [7] A. Lodi, S. Martello, and D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems," *INFORMS J. Comput.*, vol. 11, no. 4, pp. 345–357, 1999.
- [8] M. A. Boschetti and A. Mingozzi, "The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case," *Q. J. Belgian, French Ital. Oper. Res. Soc.*, vol. 1, no. 1, pp. 27–42, 2003.
- [9] M. A. Boschetti and A. Mingozzi, "The Two-Dimensional Finite Bin Packing Problem. Part II: New lower and upper bounds," *Q. J. Belgian, French Ital. Oper. Res. Soc.*, vol. 1, no. 2, pp. 135–147, 2003.
- [10] O. Faroe, D. Pisinger, and M. Zachariasen, "Guided Local Search for the Three-Dimensional Bin-Packing Problem," *INFORMS J. Comput.*, vol. 15, no. 3, pp. 267–283, 2003.
- [11] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *Oper. Res.*, vol. 113, pp. 469–499, 1999.
- [12] M. Monaci and P. Toth, "A Set-Covering-Based Heuristic Approach for Bin-Packing Problems," *Informs J. Comput.*, vol. 18, no. 1, pp. 71–85, 2006.
- [13] F. Parreño, R. Alvarez-Valdés, J. F. Oliveira, and J. M. Tamarit, "A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing," *Ann. Oper. Res.*, vol. 179, no. 1, pp. 203–220, 2010.
- [14] T. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *J. Glob. Optim.*, pp. 109–133, 1995.
- [15] P. Hansen and N. Mladenović, "Variable neighborhood search," *Search Methodol. Introd. Tutorials Optim. Decis. Support Tech.*, pp. 211–238, 2005.
- [16] C. Blum and V. Schmid, "Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm," *Procedia Comput. Sci.*, vol. 18, pp. 899–908, 2013.
- [17] L. Wong, L. S. Lee, and U. P. M. Serdang, "Heuristic Placement Routines for Two-Dimensional Bin Packing Problem," *J. Math. Stat.*, vol. 5, no. 4, pp. 334–341, 2009.
- [18] J. E. Beasley, "An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure," *Oper. Res.*, vol. 36, no. 1, pp. 49–64, 1985.
- [19] G. Scheithauer and J. Terno, "Modeling of packing problems," *Optimization*, vol. 28, no. 1, pp. 63–84, 1993.
- [20] S. P. Fekete and J. Schepers, "On more-dimensional packing I: Modeling," ZPR Technical Report 97.288, 2000.
- [21] S. P. Fekete and J. Schepers, "On more-dimensional packing II: Bounds," ZPR Technical

Report 97.289, 2000.

- [22] S. P. Fekete and J. Schepers, "On more-dimensional packing III: Exact Algorithms," ZPR Technical Report 97.290, 2000.
- [23] M. A. Boschetti, A. Mingozzi, and E. Hadjiconstantinou, "New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem," *IMA J. Manag. Math.*, vol. 13, no. 2, pp. 95–119, 2002.
- [24] A. Caprara and M. Monaci, "On the two-dimensional Knapsack Problem," *Oper. Res. Lett.*, vol. 32, no. 1, pp. 5–14, 2004.
- [25] M. Dolatabadi, A. Lodi, and M. Monaci, "Exact algorithms for the two-dimensional guillotine knapsack," *Comput. Oper. Res.*, vol. 39, no. 1, pp. 48–53, 2012.
- [26] K. Fleszar, "An Exact Algorithm for the Two-Dimensional Stage-Unrestricted Guillotine Cutting/Packing Decision Problem," *INFORMS J. Comput.*, vol. 28, no. 4, pp. 703–720, 2016.
- [27] F. Furini, E. Malaguti, and D. Thomopoulos, "Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming," *INFORMS J. Comput.*, vol. 28, no. 4, pp. 736–751, 2016.
- [28] K. K. Lai and J. W. M. Chan, "Developing a simulated annealing algorithm for the cutting stock problem," *Comput. Ind. Eng.*, vol. 32, no. 1, pp. 115–127, 1997.
- [29] T. W. Leung, C. H. Yung, and M. D. Troutt, "Applications of genetic search and simulated annealing to the two-dimensional non-guillotine cutting stock problem," *Comput. Ind. Eng.*, vol. 40, no. 3, pp. 201–214, 2001.
- [30] S. Jakobs, "On genetic algorithms for the packing of polygons," *Eur. J. Oper. Res.*, vol. 88, no. 1, pp. 165–181, 1996.
- [31] T. W. Leung, C. K. Chan, and M. D. Troutt, "Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem," *Eur. J. Oper. Res.*, vol. 145, no. 3, pp. 530–542, 2003.
- [32] J. E. Beasley, "A population heuristic for constrained two-dimensional non-guillotine cutting," *Eur. J. Oper. Res.*, vol. 156, no. 3, pp. 601–627, 2004.
- [33] R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit, "A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems," *J. Oper. Res. Soc.*, vol. 56, no. 4, pp. 414–425, 2005.
- [34] R. Alvarez-Valdés, F. Parreño, and J. M. Tamarit, "A tabu search algorithm for a two-dimensional non-guillotine cutting problem," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1167–1182, 2007.
- [35] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.
- [36] E. Hadjiconstantinou and M. Iori, "A hybrid genetic algorithm for the two-dimensional single large object placement problem," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1150–1166, 2007.

- [37] J. F. Gonçalves, “A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1212–1229, 2007.
- [38] A. Bortfeldt and T. Winter, “A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces,” *Int. Trans. Oper. Res.*, vol. 16, no. 6, pp. 685–713, 2009.
- [39] J. F. Gonçalves and M. G. C. Resende, “A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem,” *J. Comb. Optim.*, vol. 22, no. 2, pp. 180–201, 2011.
- [40] L. Wei, W. C. Oon, W. Zhu, and A. Lim, “A skyline heuristic for the 2D rectangular packing and strip packing problems,” *Eur. J. Oper. Res.*, vol. 215, no. 2, pp. 337–346, 2011.
- [41] K. He, W. Huang, and Y. Jin, “An efficient deterministic heuristic for two-dimensional rectangular packing,” *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1355–1363, 2012.
- [42] S. C. H. Leung, D. Zhang, C. Zhou, and T. Wu, “A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem,” *Comput. Oper. Res.*, vol. 39, no. 1, pp. 64–73, 2012.
- [43] I. Kierkosz and M. Luczak, “A hybrid evolutionary algorithm for the two-dimensional packing problem,” *Cent. Eur. J. Oper. Res.*, vol. 22, no. 4, pp. 729–753, 2014.
- [44] R. Martí, M. G. C. Resende, and C. C. Ribeiro, “Multi-start methods for combinatorial optimization,” *Eur. J. Oper. Res.*, vol. 226, no. 1, pp. 1–8, 2013.
- [45] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher, “New heuristic and interactive approaches to 2D rectangular strip packing,” *J. Exp. Algorithmics*, vol. 10, no. 1, p. 1.2, 2005.
- [46] E. Hadjiconstantinou and N. Christofides, “An exact algorithm for general, orthogonal, two-dimensional knapsack problems,” *Eur. J. Oper. Res.*, vol. 83, no. 1, pp. 39–56, 1995.
- [47] P. Y. Wang, “Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems,” *Oper. Res.*, vol. 31, no. 3, pp. 573–586, 1983.
- [48] N. Christofides and C. Whitlock, “An Algorithm for Two-Dimensional Cutting Problems,” *Oper. Res.*, vol. 25, no. 1, pp. 30–44, 1977.
- [49] S. P. Fekete, J. Schepers, and J. C. van der Veen, “An Exact Algorithm for Higher-Dimensional Orthogonal Packing,” *Oper. Res.*, vol. 55, no. 3, pp. 569–587, 2007.
- [50] E. Hopper and B. C. H. Turton, “An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem,” *Eur. J. Oper. Res.*, vol. 128, no. 1, pp. 34–57, 2001.

Adaptive Sequence-based Heuristic for Two-Dimensional Guillotine Cutting Problems

Abstract We present heuristics for two related two-dimensional guillotine cutting problems. The first problem aims to minimize the number of identical objects required to extract all demanded items, while the second problem aims to maximize the value of the items that are extracted from one object. The proposed heuristics create iteratively a new sequence of items types that defines the cutting order to generate a new cutting plan. The heuristics are adaptive in the sense that try to retain, in the next sequence to be generated and evaluated, characteristics of previous sequences that provided good results. Also, for one of the problems considered in this work, a Path Relinking procedure is combined with the proposed heuristic to improve the results. Computational results show that these heuristics can generate high-quality solutions using small computing times and are competitive with other approaches from the literature.

Keywords: Two-dimensional, Rectangular, Guillotine, Placement Problem, Cutting Problem, Heuristics

1. Introduction

Given a set of items types with a size of m , cutting problems intent to extract those items from larger objects considering the objective to attain and the associated constraints. As we consider the two-dimensional rectangular case, objects and item types i (with $i = 1..m$) are characterized by length and height ($L \times H$, and $l_i \times h_i$, respectively).

If each item type i has an associated upper bound d_i on the number of units that a solution can contain, the problem is referred to as constrained, otherwise is unconstrained. Each item type as, also an associated value v_i and the problem is referred to as unweighted if all item types' values are equal to its area, otherwise is referred to as weighted problem.

Cut constraints are also considered in this paper. The cutting patterns generated must only consider guillotine cuts, i.e., all cuts must be performed in a straight line from one edge of the object to the opposite one. The cut sequence must be performed in stages, i.e., perpendicular rotation of the blades at each stage is considered. The cutting patterns generated consider horizontal first stage cuts, meaning that the object will be cut, while possible, into horizontal strips and the height of the strips is defined by the taller item included, usually referred to as restricted problem, i.e., at least one item that can be extracted with just one additional cut, without trimming cut. The second stage is vertical and will perform the same action but now considering the strips created at the first stage. The process continues until the maximum number of stages is met. When a bound on the maximum number of stages (k) exists, the problem is referred to as k -staged, otherwise referred to as non-staged. Figure 1 depicts to the left a two-staged pattern and to the right a three-staged pattern.

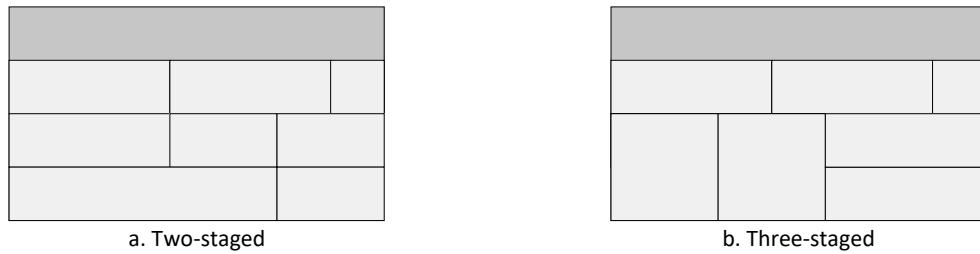


Figure 1. Two- and three-staged cuts.

If a problem considers that an extra trimming cut is allowed, the problem is referred to as non-exact, otherwise referred to as exact. Figure 2 depicts to the left an exact two-staged pattern and to the right a non-exact two-staged pattern in which the darker grey identifies the trimming area.

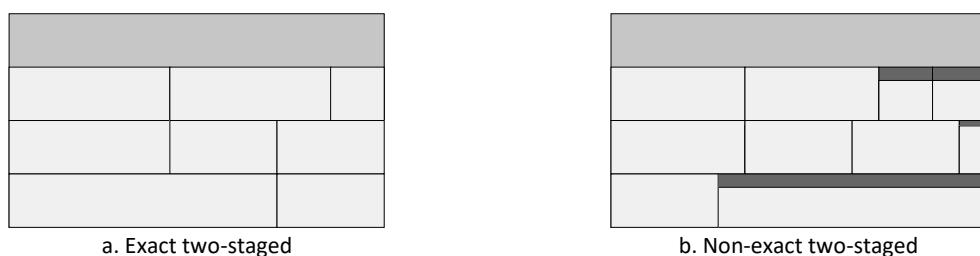


Figure 2. Exact and non-exact problems.

In this paper, we address two related problems belonging to the cutting and packing problem family. Following the typology of Wäscher *et al.* [1], the first problem is classified as Single Stock Size Cutting Stock Problem (SSSCSP) and aims to cut from an unlimited number of

identical objects an entire set of small item types, targeting the minimization of the number of objects used. The second problem is classified as Single Large Object Placement Problem (SLOPP) and aims to maximize the sum of the values of the items to be extracted from one object. Considering the cut and demand constraints, the problems considered in this paper are denoted as (non-)exact two- and three-staged SSSCSP, (non-)exact two-staged constrained SLOPP, and non-staged unconstrained SLOPP. We consider that the items cannot be rotated and must be orthogonally cut from the object.

Since the seminal work of Gilmore and Gomory in [2] and [3], proposing a Column Generation approach for the Cutting Stock Problem, the interest on the cutting and packing topic has been growing in the literature. Mainly due to its complexity (NP-hard, see Garey and Johnson [4]) and due to the high impact on practice in many areas, e.g., industry (wood, metal, and glass) and logistic (packing of boxes and load of containers).

The remaining of the paper is organized as follows. Section 2 presents, chronologically by year of publication, the most relevant articles found in the literature for the problems considered in this paper. Section 3 presents a description of our heuristics, hereafter denoted as Adaptive Sequence-based Heuristics (ASH), and in Section 4, computational results are reported comparing our approach to other solution methods from the literature. Finally, conclusions and future work directions are given in Section 5.

2. Literature Review

Gilmore and Gomory [5] extended for the two-dimensional CSP the Column Generation approach taken in [2] and [3]. The authors presented an algorithm for the unconstrained two-staged Single Knapsack Problem (SKP, see Wäscher *et al.* [1]) solving a one-dimensional SKP for each one of the item types to create strips. The strips created are then used to fill the object through the resolution of a one-dimensional SKP.

A recursive algorithm for the unconstrained SLOPP making use of discretization points to reduce the search space was proposed in Herz [6].

In Christofides and Whitlock [7] a tree-search for the constrained SLOPP is presented. Improvements to this work were presented by Christofides and Hadjiconstantinou [8], and by Hifi and Zissimopoulos [9].

Chung *et al.* [10] presented a two-phase approximation algorithm, called Hybrid First-Fit (HFF) for the Single Bin Size Bin Packing Problem (SBSBPP, see Wäscher *et al.* [1]). The items are ordered by non-increasing heights and are packed into strips following a First-Fit policy, then the strips are packed into an object using the same policy.

Two algorithms to solve the constrained SLOPP were presented by Wang [11]. The proposed enumeration approach builds successively a bigger rectangle combining smaller rectangles, usually denominated as a bottom-up approach, as opposed to the top-down approach taken by Christofides and Whitlock [7]. Improvements to Wang's algorithm were proposed by Vasko [12], and Oliveira and Ferreira [13].

Both exact and heuristic algorithms for the unconstrained SLOPP based on Dynamic Programming (see Bellman [15]) were proposed by Beasley [14].

Berkey and Wang [16] studied the adaptation for the SBSBPP of several heuristics found in the literature for the Open Dimensional Problem (ODP, see Wäscher *et al.* [1]), namely, Finite Next-fit (FNF), Finite First-Fit (FFF), Finite Best-strip (FBS), Finite Bottom-left (FBL) and Hybrid First-Fit (HFF). The level approach FNF fills the bins by packing one item at a time at the current level, considering the items pre-ordered by non-increasing heights. New bins and new levels are only started when needed. While the FNF only evaluates one bin at a time, the FFF, evaluates all previously opened bins, packing the current item into the lowest level of the first bin in which it fits. The FBS and HFF use a similar approach where strips are created and then considered to fill the bins, the FBS uses a best-fit policy while the HFF uses the first-fit policy. The FBL is not a level-approach and packs the items, one at a time, at the bottom-left position possible considering all the started bins.

A heuristic for the unconstrained SLOPP that combines depth-first and hill-climbing search strategies using an And/Or-Graph (see Chang and Slagle [18]) to represent the solutions was presented in Morabito *et al.* [17].

A Branch-and-Bound (see Land and Doig [20], and Agin[21]) procedure to solve the constrained SLOPP taking a bottom-up approach was proposed in Viswanathan and Bagchi [19]. Improvements to this procedure were proposed in Hifi [22] and in Cung *et al.* [23].

An approximation algorithm for the unconstrained SLOPP making use of a selection of strips obtained by solving a sequence one-dimensional SKP through Dynamic Programming is presented in Fayard and Zissimopoulos [24]. A generalization of the approach to deal with large-scale (un)constrained (un)weighted SLOPP is proposed later in Fayard *et al.* [25].

Morabito and Arenales [26] compared the results obtained by the approaches proposed by Beasley [14] and by Gilmore and Gomory [5] to solve large-scale unconstrained SLOPP. Beasley's algorithm was not able to obtain better results than the ones proposed in [5] (which considers the two-staged case but produces feasible solutions for the non-staged) since the discretization points removed to reduce the search space can in some cases be necessary to achieve the optimal solution.

In Morabito and Arenales [27], an algorithm for the constrained k -staged SLOPP that combines backtracking and hill-climbing search strategies using a And/Or-Graph representation is presented.

Heuristics to the (non-)guillotine SBSBPP are presented in Lodi *et al.* [28]. A Knapsack Packing heuristic for the two-staged problem is presented, the strips are created iteratively, first packing the highest unpacked item, then a knapsack problem is solved to fill the rest of the strip considering all unpacked items that do not exceed the height of the strip, i.e., height of the first element. The generated strips are then used to fill the bins solving the associated one-dimensional SBSBPP. In the same work, a Unified Tabu Search Framework is proposed, this procedure is adaptable for each specific problem by changing uniquely the inner heuristic to explore the neighbourhood.

Hifi [29] presented an exact algorithm for large-scale unconstrained two- and three-staged SLOPP. In Hifi and Roucairol [30] both approximate and exact algorithms to solve the (un)weighted two-staged SLOPP are presented. The first algorithm, based on the algorithm proposed by Gilmore and Gomory [5] for the unconstrained case, builds horizontal and vertical strips, then selects the best strips to generate the cutting pattern solving a series of one-dimensional SKP through a Dynamic Programming procedure. This heuristic is used to

create an initial lower bound to a Branch-and-Bound procedure to solve to optimality the problem.

Constructive heuristics, a Greedy Randomized Adaptive Search Procedure (GRASP, see Feo and Resende [32]), a Tabu Search (see Glover [33]), and a Path Relinking (see Glover [34]) to solve large-scale (un)constrained (un)weighted SLOPP were proposed in Alvarez-Valdés *et al.* [31].

Two integer linear programming models for the two-staged SLOPP are presented in Lodi and Monaci [35] and in Lodi *et al.* [36] new integer programming models and bounds are proposed for the two-staged ODP and SSBPP.

Hifi and M'Hallah [37] presented an exact algorithm for the constrained two-staged SLOPP based on the one proposed by Hifi and Roucairol [30] but with different bounds and with new pruning strategies to avoid duplicated patterns.

Branch-and-Cut-and-Price procedures (BCP, see Jünger and Thienel [39]) for the one-dimensional SSSCSP and for the two-dimensional two-staged SLOPP are proposed by Belov and Scheithauer [38]. The Branch-and-Cut-and-Price algorithm combines Branch-and-Bound, Cutting Planes and Column Generation.

In Hifi and M'Hallah [40] three algorithms for the constrained two-staged SLOPP, namely, the Strip Generation Algorithm (SGA), Extended SGA (ESGA), and the Hill-Climbing ESGA (HESGA) are presented. The SGA first generates a set of uniform strips and general strips and then searches for good combinations of strips to fill the object. The ESGA fills one section of the object with the SGA, then uses a procedure that takes into consideration discretization points to fill the second section. HESGA combines the ESGA with hill-climbing strategies.

Alvarez-Valdés *et al.* [41] presented two GRASP and a Path Relinking approach for the two-staged SLOPP. One of the GRASP approaches is based on items, while the other is based on strips generated by solving knapsack problems. Making use of the high-quality solutions obtained by the GRASP based on strips and the more diverse set of solutions obtained by the GRASP based on items, the authors presented a Path Relinking.

An integer linear programming model and a Branch-and-Price algorithm for the three-staged SSBP is presented in Puchinger and Raidl [42]. A Branch-and-Price for the SSBP was also presented by Pisinger and Sigurd [43].

Cintra *et al.* [44] proposed Dynamic Programming algorithms for the k - and non-staged SLOPP. The SLOPP algorithms are then used to solve the sub-problem in a Column Generation approach for solving the SSSCSP, ODP and Multiple Stock Size Cutting Stock Problem (MSSCSP, see Wäscher *et al.* [1]).

A Beam Search (see Ow and Morton [46]) based algorithm for the two-staged SLOPP is proposed in Hifi *et al.* [45]. The Beam Search is based on the best-first search in which at each step of the search only the most promising n (n as the beam width) nodes are considered for branching.

A Variable Neighbourhood Descent (VND, see Hansen and Mladenovic [48]) for the two- and three-staged SSBP is presented in Alvelos *et al.* [47]. The VND is an iterative improvement method that systematically switches between neighbourhoods.

Regarding mathematical formulations, Macedo *et al.* [49] presented an Arc-Flow model for the two-staged SSSCSP based on the work of Valério de Carvalho [50] for the one-dimensional case and Silva *et al.* [51] proposed integer programming models for the two- and three-staged (non-) exact SSSCSP (with extensions for the MSSCSP).

Chan *et al.* [52] presented a heuristic for the two- and three-staged SSBP called Stochastic Neighbourhood Structures (SNS). The SNS, based on the VND, requires that all neighbourhood structures explored to be stochastics.

In Cui and Zhao [53] an algorithm for the two-staged SSSCSP, denoted as Repeated Constrained Column-Generation (RCCG), that uses the Column Generation approach to solve the sub-problems and the residual problems is proposed.

An algorithm for the (un)weighted SLOPP that combines the bottom-up and top-down approaches is proposed in Wei and Lim [54].

3. Adaptive Sequence-based Heuristic (ASH)

We propose a Multi-start heuristic (see Martí *et al.* [55]) that iteratively creates a given number of solutions (cutting plans), each of them considering a new ordering of items types used to generate the pattern(s) to include into the solution. The main difference between solving SSSCSP and SLOPP instances is that for SSSCSP new patterns are created and added to the solution until all items have their cut position defined, while for the SLOPP, a solution is created considering only one object.

The main concept behind the proposed heuristics is that if a good solution was packed using some base ordering S_{base} , it may be the case that a better solution can be reached introducing few changes to the base ordering by generating a new packing sequence $S_{current}$. Otherwise, if no improvement is obtained, the heuristic incrementally allows more changes to the base ordering to provide diversification through the search. Orderings that generate a new best solution are used as base ordering in the next iterations. The main steps of the proposed heuristics are shown in Algorithm 1.

Algorithm 1. ASH main steps.

```

 $S_{base} \leftarrow$  Items ordered by efficiency with decreasing value as tiebreaker
 $\alpha \leftarrow \alpha_{min}$ 
Generate a new solution with sequence  $S_{base}$ 
While stopping criteria are not met do
     $S_{current} \leftarrow$  Generate a new sequence of boxes based on  $S_{base}$  and  $\alpha$ 
    Generate a new solution with sequence  $S_{current}$ 
    If a new best solution is found then
         $S_{base} \leftarrow S_{current}$ 
         $\alpha \leftarrow \alpha_{min}$ 
    Otherwise
         $\alpha \leftarrow \min\{\alpha + \alpha_{inc}, \alpha_{max}\}$ 

```

ASH iterates until a maximum number of iterations has been performed or the optimality is guaranteed, i.e., for the SSSCSP the solution value is equal to the Continuous Lower Bound ($CLB = \lceil \frac{\sum_{i=1}^m l_i h_i}{LH} \rceil$, see Martello and Vigo [56]) and for the SLOPP all the items are included in the cutting pattern.

We use as starting base sequence S_{base} the items types ordered by efficiency $e_i = \frac{v_i}{(l_i \times h_i)}$ as defined in Alvarez-Valdés *et al.* [42] using decreasing v_i as a tiebreaker. With this ordering, ASH starts with a base sequence of item types ordered by efficiency for weighted problems,

and with a base sequence ordered by decreasing area for unweighted problems since e_i is equal to 1 for all item types.

At each iteration, a new sequence $S_{current}$ is generated, based on S_{base} and using a probability α , which will define the cutting order to be used when creating a new solution. If the newly created solution is the best one found so far, the current sequence becomes the current base and α is reset to its minimum value α_{min} . Otherwise, the base sequence is not changed and α is updated as follows, $\alpha = \min\{\alpha + \alpha_{inc}, \alpha_{max}\}$. The reset of α to its minimum value will generate a new sequence that is very similar to the base sequence intensifying the search to regions of the solution space considered promising. Incrementing α will allow to diversify the search generating sequences that differ incrementally more from the base sequence.

The new sequences are generated based on the algorithm proposed by Lesh *et al.* [57]. This method (see Algorithm 2) creates a new sequence (Out) adding with a probability of α one element at a time from the input sequence (In) into the new sequence until all elements from In are in Out .

Algorithm 2. Sequence generator.

```

Input: sequence  $In$ , probability  $\alpha$ 
 $Out \leftarrow \emptyset$ 
 $n \leftarrow |In|$ 
for  $i \leftarrow 1, \dots, n$  do
     $j \leftarrow 1$ 
     $Out_i \leftarrow \emptyset$ 
    while  $out_i = \emptyset$  do
        if  $\alpha \leq$  generated random value then
             $Out_i \leftarrow In_j$ 
             $In \leftarrow In \setminus \{In_j\}$ 
         $j \leftarrow (j \bmod |In|) + 1$ 
return  $Out$ 

```

The patterns are generated as follows. Iteratively, horizontal strips are created with the first item type, considering the ordering defined by $S_{current}$, that fits into the remaining object height. When created, the first-stage strips are filled solving bounded 1D SKP using the solution method proposed by Pisinger [58]¹. For problems where the maximum number of stages is greater than 2, each of the resulting strips is filled solving the associated SKP. The

¹ Code available online at <http://www.diku.dk/~pisinger/codes.html> for academic and non-commercial purposes.

SKP solved to generate the patterns only consider items that fit at the current strip, have residual demand and can be included due to the cut constraints.

For the SSSCSP, the patterns generated are added to the current solution with the maximum frequency of cut allowed by the current residual demand.

Path Relinking

ASH can be further enhanced searching for even better results with the inclusion of search strategies, such as local search after the cutting plan generation, or with more sophisticated ones such as a Path Relinking (proposed by Glover *et al.* [59]) as a final phase of this heuristic. The Path Relinking creates new solutions incorporating into an (initializer) solution attributes from another (guiding) solution exploiting trajectories that connect them.

To improve the results of the SLOPP, all (distinct) patterns generated by ASH are added to a pool of patterns ordered decreasingly by their objective function value. The best $RefSet_{size}$ solutions from the pool are considered by the Path Relinking procedure either as initializer or guiding solutions.

Considering one initializer and one guiding pattern, one at a time, first stage strips from the guiding pattern are added to the current initializer pattern. At each new strip inclusion, the procedure regains the solution feasibility by means of two methods, both, iteratively removing strips (without considering the newly added strip) from the initializer pattern. The first, iteratively, removes the strips that reduces more the demand infeasibility. Next, if the object remaining height is negative, the second method iteratively removes the taller strip until feasibility is achieved.

The best pattern found, at the end of each Path Relinking between one initializer and one guiding pattern, is filled as previously described considering the sequence of items types ordered by efficiency (e_i) and decreasing value (v_i) as tiebreaker.

Each pattern generated through the Path Relinking is improved by means of a simple local search. This local search swaps items that increase the objective function value between those in the pattern and those who have residual demand greater than 0. The local search performs the first swap possible, while such move exists.

4. Computational Results

The proposed heuristics were implemented in C and the computational experiments were run on a computer with an Intel Core i7-4800MQ at 2.70 GHz with 8 Gb RAM and operating system Linux Ubuntu 18.04.

The results obtained by the Adaptive Sequence-Based Heuristic (ASH) for solving both the Single Stock Size Cutting Stock Problem and the Single Large Object Placement Problem are next presented. Each instance was run only once generating at most 2000 cutting patterns with α_{min} , α_{max} and α_{inc} set to 0.1, 0.9 and 0.005, respectively.

For both problems, the values of α_{min} , α_{max} , and α_{inc} were set through experiments carried out considering different options, and the datasets considered correspond to the ones used by the algorithms that were used for comparison.

4.1. Single Stock Size Cutting Stock Problem

Three datasets were used to test the performance of ASH for the SSSCSP. The first dataset, identified in this paper as SET1, was defined by Fayard *et al.* [25] and contains 30 instances previously used by other authors to evaluate their solution methods. The dataset ATP contains the last 20 instances (APT30 to ATP49) defined in Alvarez-Valdés *et al.* [31]. The dataset CLASS contains 500 instances, grouped in 10 subsets of 50 instances each, and was defined by Berkey and Wang [16] and Martello and Vigo [56]. The main characteristics of the datasets are presented in Table 1, where the columns show in the order of appearance, the name of the dataset, the number of instances, the objects dimensions range, the number of different item types, the item types dimension range and the average demand.

Table 1. Main features of the datasets.

Dataset	Number of instances	Object $w \times h$	Number of item types	Items $w \times h$	Average demand
SET1	30	[20-267]x[20-244]	[5-40]	[1-170]x[2-135]	2.5
ATP	20	[167-960]x[124-983]	[25-58]	[8-363]x[6-390]	5.0
CLASS	500	[10-300]x[10-300]	[18-100]	[1-100]x[1-100]	1.1

The results obtained by ASH are shown in Table 2 to Table 7. The first column, identified in Table 2 to Table 5 with column header Instance, gives the name that each instance is best-known, while in Table 6 and Table 7, identified with column header subset, gives the subset

identification. In Table 2 to Table 5, the lower bounds for each instance (denoting the minimum number of bins required to fulfil the demand reported by Silva *et al.* [16]) are shown in column Z_{lb} . The next columns, present the results obtained by other approaches and by ASH, grouped by the number of stages considered (identified at the first line of those columns). The last row shows, in Table 2 to Table 5, the average execution times for each instance, while, in Table 6 and Table 7, the average time to solve each subset is reported.

In the following tables, we will use the following notation. The symbol $=$ denotes that the approach obtained a result equal to Z_{lb} . The symbol $*$ denotes that the approach failed to obtain a solution within the computation time limit. The symbol $-$ denotes an unavailable value.

Table 2. Computational results on the instances of SET1 for exact problems.

Instance	2-staged exact			3-staged exact		
	Z_{lb}	Silva	ASH	Z_{lb}	Silva	ASH
A1	27	=	=	23	=	=
A2	15	=	=	12	=	=
A3	10	=	=	8	=	=
A4	8	=	=	5	=	=
A5	8	=	=	4	=	5
CHL1	11	=	=	6	=	=
CHL2	4	=	=	3	=	=
CHL5	5	=	=	3	=	4
CHL6	9	=	=	5	6	6
CHL7	9	=	=	6	=	=
CU1	15	=	16	12	=	=
CU2	20	=	21	14	=	15
CW1	13	=	=	10	=	=
CW2	17	=	=	12	=	13
CW3	22	=	23	16	=	17
Hchl2	9	=	10	6	=	=
Hchl3s	4	=	=	3	=	=
Hchl4s	3	=	=	2	=	=
Hchl6s	7	=	=	5	=	=
Hchl7s	11	=	12	7	=	8
Hchl8s	3	=	=	2	=	=
Hchl9	14	=	15	10	=	11
HH	2	=	=	2	=	=
OF1	5	=	=	4	=	=
OF2	6	=	=	4	=	5
STS2	17	=	18	12	=	13
STS4	6	=	=	5	=	=
W	31	=	=	24	=	=
2	3	=	=	2	=	=
3	24	=	=	23	=	=
Avg. time (s)	0.42	0.01		335.28	0.02	

Table 2 presents the results for the instances of SET1 solved as two- and three-staged exact problems. We compare our results with the exact algorithm proposed by Silva *et al.* [16] (column Silva). The results presented in Table 2 show that although achieving better results for this dataset, the computational time required by the exact method grows extremely from two- to three-staged problems, while ASH maintains a low computational time to solve the instances. It can be noted that the maximum gap to the optimal solution is 1.

The results for the instances of ATP solved as two- and three-staged exact problems are presented in Table 3. It must be noted that Silva *et al.* [16] set a computational time limit of 7200 seconds to solve each instance and was not able to attain a feasible solution for two instances, ATP31 and ATP40, for the three-staged exact problem.

Table 3. Computational results on the instances of ATP for exact problems.

Instance	2-staged exact			3-staged exact		
	Z_{lb}	Silva	ASH	Z_{lb}	Silva	ASH
ATP30	12	=	=	8	9	9
ATP31	19	=	=	14	*	=
ATP32	16	=	=	12	14	13
ATP33	18	=	=	12	13	13
ATP34	9	=	=	6	=	=
ATP35	10	=	=	8	=	=
ATP36	11	=	=	8	=	=
ATP37	16	=	=	11	15	12
ATP38	15	=	=	10	12	11
ATP39	16	=	=	11	12	12
ATP40	20	=	=	15	*	16
ATP41	16	=	=	12	=	=
ATP42	21	=	=	15	17	16
ATP43	18	=	=	12	15	13
ATP44	14	=	=	9	=	=
ATP45	11	=	=	8	=	=
ATP46	16	=	17	11	12	12
ATP47	18	=	=	12	14	13
ATP48	11	=	=	8	9	=
ATP49	8	=	=	5	6	=
Avg. time (s)	217.1	0.04		5561.78	0.10	

From Table 3, we can observe that for the two-staged case only for one instance the solution does not match the optimal solution, while for the three-staged case, considering the imposed time limit, better results than those obtained by the exact method were obtained.

Table 4 presents the results obtained solving the instances of SET1 as two- and three-staged non-exact problems, while Table 5, presents the results for the non-exact case for the set ATP.

We compare these results with the approaches of Silva *et al.* [16] and the Column Generation based approach presented by Cui and Zhao [18] (column RCCG) for the two-staged problem.

As can be observed in Table 4 and Table 5, the proposed heuristic is able to attain results very similar to those obtained by the RCCG. It is noteworthy the growth of the execution time in both Silva and RCCG from SET1 to ATP making clear that the extra complexity influence in a great manner the performance of these approaches.

Table 4. Computational results on the instances of SET1 for non-exact problems.

Instance	2-staged non-exact				3-staged non-exact		
	Z_{lb}	Silva	RCCG	ASH	Z_{lb}	Silva	ASH
A1	23	=	=	=	23	=	=
A2	12	=	13	=	12	=	=
A3	8	=	=	=	8	=	=
A4	5	=	=	=	5	=	=
A5	5	=	=	=	4	=	5
CHL1	6	=	=	=	6	=	=
CHL2	3	=	=	=	3	=	=
CHL5	4	=	=	=	3	=	4
CHL6	6	=	=	=	5	=	6
CHL7	6	=	=	=	6	=	=
CU1	12	=	=	=	12	=	=
CU2	15	=	=	=	14	=	15
CW1	10	=	=	=	10	=	=
CW2	12	=	13	13	12	=	13
CW3	16	=	=	18	16	=	17
Hchl2	6	=	=	=	6	=	=
Hchl3s	3	=	=	=	3	=	=
Hchl4s	2	=	3	=	2	=	=
Hchl6s	5	=	=	=	5	=	=
Hchl7s	7	=	=	8	7	=	=
Hchl8s	2	=	=	=	2	=	=
Hchl9	10	=	=	11	10	=	11
HH	2	=	=	=	2	=	=
OF1	4	=	=	=	4	=	=
OF2	5	=	=	=	4	=	5
STS2	12	=	=	13	12	=	13
STS4	5	=	=	=	5	=	=
W	24	=	=	=	24	=	=
2	2	=	=	=	2	=	=
3	23	=	24	=	23	=	=
Avg. time (s)	8.85	0.19	0.02		121.33	0.02	

As can be observed in Table 5, with respect to the three-staged problem our approach obtained, considering the time limit imposed, better results than the exact approach with a computational time extremely low. Again, the exact algorithm was not able to give,

considering the time limit imposed, a feasible solution for two instances, namely ATP30 and ATP40.

Table 5. Computational results on the instances of ATP for non-exact problems.

Instance	2-staged non-exact				3-staged non-exact		
	Z_{lb}	Silva	RCCG	ASH	Z_{lb}	Silva	ASH
ATP30	9	=	=	=	8	*	9
ATP31	14	15	=	=	14	15	=
ATP32	13	=	=	=	12	14	13
ATP33	12	13	13	13	12	13	13
ATP34	6	=	=	=	6	=	=
ATP35	8	=	=	=	8	=	=
ATP36	8	=	=	=	8	=	=
ATP37	12	=	=	=	11	12	12
ATP38	11	=	=	=	10	11	11
ATP39	11	=	=	12	11	12	12
ATP40	15	=	=	16	15	*	16
ATP41	12	=	=	=	12	=	=
ATP42	15	16	=	16	15	16	16
ATP43	13	14	=	=	12	14	13
ATP44	9	=	=	=	9	10	=
ATP45	8	=	=	=	8	=	=
ATP46	11	=	=	12	11	12	12
ATP47	13	=	=	=	12	13	13
ATP48	8	9	9	9	8	9	9
ATP49	5	=	6	6	5	6	6
Avg. time (s)	3642.58	9.5	0.08		5809.76	0.1	

Table 6 and Table 7 present the results obtained using the CLASS dataset. We show in the following two tables the results obtained when solving this dataset as two- and three-staged exact (Table 6) and non-exact problems (Table 7).

Table 6. Computational results on the subsets of CLASS for exact problems.

Subset	2-staged exact			3-staged exact		
	SNS	VND	ASH	SNS	VND	ASH
CLASS 1	1134	1134	1134	1012	1021	1018
CLASS 2	167	167	167	128	128	128
CLASS 3	998	999	1002	728	731	735
CLASS 4	270	270	270	127	128	132
CLASS 5	1386	1386	1390	921	925	930
CLASS 6	383	383	383	116	117	118
CLASS 7	1059	1061	1064	845	846	851
CLASS 8	1556	1556	1556	861	867	859
CLASS 9	2320	2320	2320	2130	2131	2131
CLASS 10	891	892	897	525	523	536
Avg. time (s)	17.35	-	1.35	12.71	-	1.69

We compare our results with solution methods for solving the SSBPP (as the CLASS dataset average demand is close to 1), namely, SNS (Chan *et al.* [17]) and VND (Alvelos *et al.* [12]).

The VND results were obtained from Chan *et al.* [17]. Before interpreting the computation results, it must be referred that Chan *et al.* [17] presented the best results obtained from 30 runs.

Table 7. Computational results on the subsets of CLASS for non-exact problems.

Subset	2-staged non-exact			3-staged non-exact		
	SNS	VND	ASH	SNS	VND	ASH
CLASS 1	1023	1029	1023	1015	1019	1016
CLASS 2	131	131	131	128	128	125
CLASS 3	729	731	736	730	731	734
CLASS 4	130	130	132	126	126	127
CLASS 5	920	925	931	915	925	928
CLASS 6	117	117	118	116	116	116
CLASS 7	848	848	858	841	844	846
CLASS 8	863	867	859	854	865	857
CLASS 9	2130	2131	2131	2130	2130	2131
CLASS 10	528	527	537	521	522	523
Avg. time (s)	10.76	12.32	1.42	10.6	-	1.63

The SNS attains better results in almost all subsets. Although a direct comparison is not possible, our approach seems to require much less time to solve these subsets. ASH, when compared with the SBSBPP specifically tailored heuristics, was able to attain interesting results, especially when compared with the VND heuristic. Considering the good computational results obtained and the implementation simplicity of ASH, one can conclude that the heuristics proposed are a very attractive approach to solve both the Cutting Stock and the Bin Packing Problems.

4.2. Single Large Object Placement Problem

Four datasets from the literature are used to test the performance of ASH and ASH combined with a Path Relinking procedure (ASH+PR) for the SLOPP. The main characteristics of the datasets are presented in Table 8. The columns show, in the order in which they appear, the name of the dataset, the number of instances, the objects dimensions ranges, the number of different item types, the item types dimension ranges, and the average demand. The dataset HR (Hifi and Roucairol [30]) and ATP (Alvarez-Valdés *et al.* [31]) contains both weighted and unweighted instances. Datasets ATP[10-29], GCUT (Beasley [14]) and CMWX (Cintra *et al.* [44]) were created for unconstrained problems, which means the items does not have associated demand, denoted in Table 8 with the symbol -.

Table 8. Main features of the datasets.

Dataset	Instances	Object $w \times h$	m	Items $w \times h$	Average demand
HR	38	[20-267]x[20-244]	[5-40]	[1-170]x[2-135]	2.5
ATP[10-29]	20	[1674-2899]x[1612-2994]	[31-59]	[96-1142]x[81-1192]	-
ATP[30-49]	20	[167-960]x[124-983]	[25-58]	[8-363]x[6-390]	5.0
GCUT	13	[250-3000]x[250-3000]	[10-50]	[62-970]x[63-1890]	-
CMWX	4	3500x3500	[42-82]	[254-970]x[116-1890]	-

The results obtained by ASH and by ASH+PR when solving the four datasets are shown in Table 9 to Table 13. The $RefSet_{size}$ for the ASH+PR was set to 50.

The content of the columns in Table 9 to Table 11 is the following. The first column, identified with column header First cut, gives the first cut direction considered when solving the instances. The column Algorithm denotes the algorithm name. Columns GAP (%), Optimums, and Time (s) give, respectively, the average GAP, the number of optimal solutions obtained, and the average execution time in seconds.

To solve the problems with first cut direction vertical, we exchange the dimensions of both objects and item types, and then solve the problem considering horizontal first cut direction.

Table 9 presents the results obtained by solving the dataset HR as a constrained two-staged SLOPP with exact cuts. We compare our results with the ones presented by Hifi and Roucairol [30] denoted as Exact and Approximate algorithms.

Table 9. Computational results for exact constrained two-staged SLOPP - Set HR.

First cut	Algorithm	GAP (%)	Optimums	Time (s)
Horizontal	Approximate	0.44	31	0.10
	Exact	0.00	38	1.17
	ASH	0.11	34	0.01
	ASH+PR	0.01	36	0.01
Vertical	Approximate	0.74	31	0.10
	Exact	0.00	38	1.17
	ASH	0.40	35	0.01
	ASH+PR	0.04	37	0.01

From the results presented in Table 9, we can observe the high-quality results that the ASH approaches obtained for this problem/set, achieving almost all the optimal solutions in both first cut directions. It must be noted that both ASH approaches obtained better results than the Approximate approach and that the computational times needed is very low.

Table 10 and Table 11 present the results obtained for dataset HR and ATP[30-49], respectively, as a constrained non-exact two-staged SLOPP. We compare our results with the ones presented by Hifi and Roucairol [30] (Exact and Approximate), Lodi and Monaci [35] (M1 and M2), Hifi and M'Hallah [37] (ALGO_ESGA and ALGO_SGA), Belov and Scheithauer [38] (BCP CG cuts, BCP M1, and BCP), Hifi and M'Hallah [40] (SGA, ESGA, and HESGA_a), Alvarez-Valdés *et al.* [41] (GRASP_Piece, GRASP_Strip, and PR), and Hifi *et al.* [45] (LBS^d₃ and GBS^d₂).

Table 10. Computational results for non-exact constrained two-staged SLOPP - Set HR.

First cut	Algorithm	GAP (%)	Optimums	Time (s)
Horizontal	Approximate	4.68	8	0.12
	Exact	0.00	38	249.19
	M1	0.00	38	27.54
	M2	0.00	38	38.80
	ALGO_ESGA	0.00	38	1.03
	ALGO_SGA	0.00	38	2.03
	BCP CG cuts	0.00	38	3.62
	BCP M1	0.00	38	10.28
	SGA	3.08	9	0.10
	ESGA	0.58	22	0.80
Vertical	GRASP_Piece	2.24	10	0.04
	GRASP_Strip	0.22	35	0.18
	PR	0.00	38	0.50
	ASH	0.31	29	0.02
	ASH+PR	0.22	36	0.02
	Approximate	5.08	7	0.12
	Exact	0.00	38	268.88
Vertical	M1	0.00	38	23.96
	M2	0.00	38	57.62
	PR	0.00	37	0.50
	ASH	0.43	28	0.02
	ASH+PR	0.10	35	0.02

The results in Table 10 demonstrate that ASH and the Path Relinking approach, applied to the solution obtained by ASH, can obtain extremely good results with a computational time extremely low.

As can be observed in Table 11, the GBS^d_2 approach outperform all the other heuristic, but ASH and ASH+PR obtain results close to those obtained by GRASP_Strip and PR. For the vertical case, ASH+PR obtained a GAP that is considerably lower than the one obtained by PR.

Noteworthy that our Path Relinking, applied to the subset of the population pattern, does not produce a notable time overhead, when compared with ASH computational time and considerably improves the results obtained.

Table 11. Computational results for non-exact constrained two-staged SLOPP - Set ATP[30-49].

First cut	Algorithm	GAP (%)	Optimums	Time (s)
Horizontal	BCP	0.01	18	142.91
	SGA	2.06	2	-
	ESGA	0.80	6	-
	HESGAa	0.30	9	13.53
	GRASP_Piece	4.06	0	0.17
	GRASP_Strip	0.23	11	0.74
	PR	0.08	14	1.18
	LBSD3	4.05	4	0.05
	GBSd2	0.00	20	0.20
	ASH	0.49	6	0.04
Vertical	ASH+PR	0.21	11	0.05
	HESGAa	0.35	5	14.42
	GRASP_Piece	2.88	0	0.17
	GRASP_Strip	0.31	13	0.89
	PR	0.18	14	1.34
	LBSD3	4.04	5	0.05
	GBSd2	0.00	20	0.20
	ASH	0.50	8	0.04
	ASH+PR	0.08	13	0.05

Table 12 presents the results for the dataset ATP considering both unconstrained and constrained non-staged problems. We compare our results with the ones presented by Alvarez-Valdés *et al.* [31] (CONS, GRASP, GRASP+PR, and TABU500) for unrestricted problems. Table 12 presents the average GAP for both unconstrained (instances 10 to 29) and constrained (instances 30 to 49) problems. The last column gives the average time to solve the complete dataset.

We handle unconstrained problems setting the demand for each item type as the maximum number of units that is possible to include in a pattern considering the object dimensions.

Table 12. Computational results for non-staged SLOPP - Set ATP.

Algorithm	GAP (%)		Time (s)
	Unconstrained ATP[10-29]	Constrained ATP[30-49]	
CONS	4.79	4.09	0.40
GRASP	2.05	1.89	23.70
GRASP+PR	1.73	1.13	63.70
TABU500	0.27	0.45	450.40
ASH	2.11	1.40	0.06
ASH+PR	1.80	1.27	0.07

Although our placement heuristic deals only with restricted patterns, the results observed in Table 12 make clear that ASH and ASH+PR can outperform, in both unconstrained and

constrained problems, other approaches specifically designed for problems that consider unrestricted patterns. The computational times are low, and the results are very close to those obtained by the GRASP and GRASP+PR.

The results obtained for the datasets GCUT and CMWX unconstrained two-, four-, and non-staged SLOPP are presented in Table 13. We compare the results obtained with the ones presented by Cintra *et al.* [44] for unconstrained problems. This table gives the average waste produced, the average execution time in seconds, and the maximum execution time observed. We only present the results obtained by ASH and ASH+PR with non-staged cuts because two-, four- and non-staged produced the same result.

From the observation of Table 13, we emphasize the low waste reduction that can be obtained from considering two-staged patterns and non-staged patterns, thus may not justify the extra computational effort required by this cutting style, as already observed in Farley [60].

Table 13. Computational results for unconstrained non-staged SLOPP - Set GCUT and CMWX.

Algorithm	Stages	Waste (%)	Time (s)	Max. time (s)
CINTRA	two-staged	2.49	37.38	223.13
CINTRA	four-staged	2.10	40.23	456.00
CINTRA	non-staged	2.09	83.87	212.66
ASH	non-staged	2.68	0.04	0.21
ASH+PR	non-staged	2.60	0.05	0.24

ASH and AHS+PR were capable to attain constantly good results with low computational times for all the presented variants of the problem. The results show that the proposed heuristics produce optimal and near-optimal solutions and are also competitive when compared with other heuristics from the literature specially tailored for specific problems. It is noteworthy the implementation simplicity and little to no parametrization needed by both ASH and AHS+PR.

5. Conclusion

We present heuristics for two related cutting problems. The first problem aims the maximum-profit subset of items to extract from an object, while the second problem intent to extract all the items from a given set using the minimum number of objects. The object and items considered are rectangular and must be extracted through guillotine cuts without

overlapping. In the proposed heuristics, a new sequence is generated at each iteration to create a new solution, i.e., cutting plan. The sequences generated try to incorporate some knowledge from previous sequences in order to intensify and diversify the solution space explored. To evaluate the performance of the proposed heuristics several computational experiments have been performed and discussed. The computational results validate the effectiveness of the proposed heuristic since it provides high-quality solutions with very low computational times in all problems considered. As the proposed approach seems to be very promising, a future research direction could be to apply it to other combinatorial optimization problems. The problems to be considered could be the three-dimensional cutting and packing problems, the facility location problems (e.g., the sequences can represent the order in which the location will be opened or the order to assign the clients to facilities) or the Travel Salesman Problem (e.g., the sequences could represent possible paths between the cities).

References

- [1] G. Wäscher, H. Haußner, and H. Schumann, “An improved typology of cutting and packing problems,” *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1109–1130, 2007.
- [2] P. C. Gilmore and R. E. Gomory, “A linear programming approach to the cutting-stock problem,” *Oper. Res.*, vol. 9, no. 6, pp. 849–859, 1961.
- [3] P. C. Gilmore and R. E. Gomory, “A linear programming approach to the cutting stock problem-Part II,” *Oper. Res.*, vol. 11, no. 6, pp. 863–888, 1963.
- [4] M. R. Garey and D. S. Johnson, “A Guide to the Theory of NP-Completeness,” *Mathematical Sciences*. W.H. Freeman and Company, San Francisco, 1979.
- [5] P. C. Gilmore and R. E. Gomory, “Multistage cutting stock problems of two and more dimensions,” *Oper. Res.*, vol. 13, no. 1, pp. 94–120, 1965.
- [6] J. Herz, “Recursive Computational Procedure for Two-dimensional Stock Cutting,” *IBM J. Res. Dev.*, vol. 16, no. 5, pp. 462–469, 1972.
- [7] N. Christofides and C. Whitlock, “An Algorithm for Two-Dimensional Cutting Problems,” *Oper. Res.*, vol. 25, no. 1, pp. 30–44, 1977.
- [8] N. Christofides and E. Hadjiconstantinou, “An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts,” *Eur. J. Oper. Res.*, vol. 83, no. 1, pp. 21–38, 1995.
- [9] M. Hifi and V. Zissimopoulos, “Constrained two-Dimensional cutting: an improvement of Christofides and Whitlock’s exact algorithm,” *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 324–331, 1997.

- [10] F. R. K. Chung, M. R. Garey, and D. S. Johnson, “On Packing Two-Dimensional Bins,” *SIAM J. Algebr. Discret. Methods*, vol. 3, no. 1, pp. 66–76, 1982.
- [11] P. Y. Wang, “Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems,” *Oper. Res.*, vol. 31, no. 3, pp. 573–586, 1983.
- [12] F. J. Vasko, “A computational improvement to Wang’s two-dimensional cutting stock algorithm,” *Comput. Ind. Eng.*, vol. 16, no. 1, pp. 109–115, 1989.
- [13] J. F. Oliveira and J. S. Ferreira, “An improved version of Wang’s algorithm for two-dimensional cutting problems,” *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 256–266, 1990.
- [14] J. E. Beasley, “Algorithms for Unconstrained Two-Dimensional Guillotine Cutting,” *J. Oper. Res. Soc.*, vol. 36, no. 4, pp. 297–306, 1985.
- [15] R. Bellman, “The Theory of Dynamic Programming,” *Bull. Am. Math. Soc.*, vol. 60, no. 6, pp. 503–515, 1954.
- [16] J. O. Berkey and P. Y. Wang, “Two-Dimensional Finite Bin-Packing Algorithms,” *J. Oper. Res. Soc.*, vol. 38, no. 5, p. 423, 1987.
- [17] R. Morabito, M. N. Arenales, and V. F. Arcaro, “An and-or-graph approach for two-dimensional cutting problems,” *Eur. J. Oper. Res.*, vol. 58, no. 2, pp. 263–271, 1992.
- [18] C. L. Chang and J. R. Slagle, “An admissible and optimal algorithm for searching AND/OR graphs,” *Artif. Intell.*, vol. 2, no. 2, pp. 117–128, 1971.
- [19] K. V Viswanathan and A. Bagchi, “Best-First Search Methods for Constrained Two-Dimensional Cutting Stock Problems,” *Oper. Res.*, vol. 41, no. 4, pp. 768–776, 1993.
- [20] A. H. Land and A. G. Doig, “An Automatic Method of Solving Discrete Programming Problems,” *Econometrica*, vol. 28, no. 3, p. 497, 1960.
- [21] N. Agin, “Optimum Seeking with Branch and Bound,” *Manage. Sci.*, vol. 13, no. 4, pp. B176–B185, 1966.
- [22] M. Hifi, “An improvement of Viswanathan and Bagchi’s exact algorithm for constrained two-dimensional cutting stock,” *Comput. Oper. Res.*, vol. 24, no. 8, pp. 727–736, 1997.
- [23] V. Cung, M. Hifi, and B. Cun, “Constrained two-dimensional cutting stock problems a best-first branch-and-bound algorithm,” *Int. Trans. Oper. Res.*, vol. 7, no. 3, pp. 185–210, 2000.
- [24] D. Fayard and V. Zissimopoulos, “An approximation algorithm for solving unconstrained two-dimensional knapsack problems,” *Eur. J. Oper. Res.*, vol. 84, no. 3, pp. 618–632, 1995.
- [25] D. Fayard, M. Hifi, and V. Zissimopoulos, “An efficient approach for large-scale two-dimensional guillotine cutting stock problems,” *J. Oper. Res. Soc.*, vol. 49, no. 12, pp. 1270–1277, 1998.
- [26] R. Morabito and M. N. Arenales, “Performance Of Two Heuristics For Solving Large Scale Two-Dimensional Guillotine Cutting Problems,” *INFOR Inf. Syst. Oper. Res.*, vol. 33, no. 2, pp. 145–155, 1995.

- [27] R. Morabito and M. N. Arenales, "Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach," *Eur. J. Oper. Res.*, 1996.
- [28] A. Lodi, S. Martello, and D. Vigo, "Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems," *INFORMS J. Comput.*, vol. 11, no. 4, pp. 345–357, 1999.
- [29] M. Hifi, "Exact algorithms for large-scale unconstrained two and three staged cutting problems," *Comput. Optim. Appl.*, vol. 18, no. 1, pp. 63–88, 2001.
- [30] M. Hifi and C. Roucairol, "Approximate and Exact Algorithms for Constrained (Un)Weighted Two-dimensional Two-staged Cutting Stock Problems," *J. Comb. Optim.*, vol. 5, no. 4, pp. 465–494, 2001.
- [31] R. Alvarez-Valdés, A. Parajón, and J. M. Tamarit, "A tabu search algorithm for large-scale guillotine (un)constrained two-dimensional cutting problems," *Comput. Oper. Res.*, vol. 29, no. 7, pp. 925–947, 2002.
- [32] T. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *J. Glob. Optim.*, pp. 109–133, 1995.
- [33] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986.
- [34] F. Glover, "A template for scatter search and path relinking," *Artif. Evol.*, vol. 1363, no. February 1998, pp. 3–51, 1998.
- [35] A. Lodi and M. Monaci, "Integer linear programming models for 2-staged two-dimensional Knapsack problems," *Math. Program.*, vol. 94, no. 2–3, pp. 257–278, 2003.
- [36] A. Lodi, S. Martello, and D. Vigo, "Models and bounds for two-dimensional level packing problems," *J. Comb. Optim.*, vol. 8, no. 3, pp. 363–379, 2004.
- [37] M. Hifi and R. M'Hallah, "An exact algorithm for constrained two-dimensional two-staged cutting problems," *Oper. Res.*, vol. 53, no. 1, pp. 140–150, 2005.
- [38] G. Belov and G. Scheithauer, "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting," *Eur. J. Oper. Res.*, vol. 171, no. 1, pp. 85–106, 2006.
- [39] M. Jünger, S. Thienel, and J. Michael, "The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization," *Softw. Pract. Exp.*, vol. 30, no. 11, pp. 1325–1352, 2000.
- [40] M. Hifi and R. M'Hallah, "Strip generation algorithms for constrained two-dimensional two-staged cutting problems," *Eur. J. Oper. Res.*, vol. 172, no. 2, pp. 515–527, 2006.
- [41] R. Alvarez-Valdés *et al.*, "GRASP and path relinking for the two-dimensional two-stage cutting-stock problem," *INFORMS J. Comput.*, vol. 19, no. 2, pp. 261–272, 2007.
- [42] J. Puchinger and G. R. Raidl, "Models and algorithms for three-stage two-dimensional bin packing," *Eur. J. Oper. Res.*, vol. 183, no. 3, pp. 1304–1327, 2007.
- [43] D. Pisinger and M. Sigurd, "Using decomposition techniques and constraint

programming for solving the two-dimensional bin-packing problem," *INFORMS J. Comput.*, vol. 19, no. 1, pp. 36–51, 2007.

- [44] G. F. Cintra, F. K. Miyazawa, Y. Wakabayashi, and E. C. Xavier, "Algorithms for two-dimensional cutting stock and strip packing problems using dynamic programming and column generation," *Eur. J. Oper. Res.*, vol. 191, no. 1, pp. 61–85, 2008.
- [45] M. Hifi, R. M'Hallah, and T. Saadi, "Algorithms for the constrained two-staged two-dimensional cutting problem," *INFORMS J. Comput.*, vol. 20, no. 2, pp. 212–221, 2008.
- [46] P. S. Ow and T. E. Morton, "Filtered beam search in scheduling," *Int. J. Prod. Res.*, vol. 26, no. 1, pp. 35–62, 1988.
- [47] F. Alvelos, T. M. Chan, P. Vilaca, T. Gomes, E. M. da C. Silva, and J. M. V. de Carvalho, "Sequence based heuristics for two-dimensional bin packing problems," *Eng. Optim.*, vol. 41, no. 8, pp. 773–791, 2009.
- [48] P. Hansen and N. Mladenović, "Variable neighborhood search," *Search Methodol. Introd. Tutorials Optim. Decis. Support Tech.*, pp. 211–238, 2005.
- [49] R. Macedo, C. Alves, and J. M. V. de Carvalho, "Arc-flow model for the two-dimensional guillotine cutting stock problem," *Comput. Oper. Res.*, vol. 37, no. 6, pp. 991–1001, 2010.
- [50] J. M. V. de Carvalho, "Exact solution of bin-packing problems using column generation and branch-and-bound," *Ann. Oper. Res.*, vol. 86, pp. 629–659, 1999.
- [51] E. M. da C. Silva, F. Alvelos, and J. M. V. de Carvalho, "An integer programming model for two- and three-stage two-dimensional cutting stock problems," *Eur. J. Oper. Res.*, vol. 205, no. 3, pp. 699–708, 2010.
- [52] T. M. Chan, F. Alvelos, E. M. da C. Silva, and J. M. V. De Carvalho, "Heuristics with stochastic neighborhood structures for two-dimensional bin packing and cutting stock problems," *Asia-Pacific J. Oper. Res.*, vol. 28, no. 2, pp. 255–278, 2011.
- [53] Y. Cui and Z. Zhao, "Heuristic for the rectangular two-dimensional single stock size cutting stock problem with two-staged patterns," *Eur. J. Oper. Res.*, vol. 231, no. 2, pp. 288–298, 2013.
- [54] L. Wei and A. Lim, "A bidirectional building approach for the 2D constrained guillotine knapsack packing problem," *Eur. J. Oper. Res.*, vol. 242, no. 1, pp. 63–71, 2015.
- [55] R. Martí, M. G. C. Resende, and C. C. Ribeiro, "Multi-start methods for combinatorial optimization," *Eur. J. Oper. Res.*, vol. 226, no. 1, pp. 1–8, 2013.
- [56] S. Martello and D. Vigo, "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," *Manage. Sci.*, vol. 44, no. April 2015, pp. 388–399, 1998.
- [57] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher, "New heuristic and interactive approaches to 2D rectangular strip packing," *J. Exp. Algorithmics*, vol. 10, no. 1, p. 1.2, 2005.
- [58] D. Pisinger, "A Minimal Algorithm for the Bounded Knapsack Problem," *INFORMS J. Comput.*, vol. 12, no. 1, pp. 75–82, 2000.

- [59] F. Glover, M. Laguna, and R. Martí, “Fundamentals of scatter search and path relinking,” *Control Cybern.*, vol. 39, no. 3, pp. 653–684, 2000.
- [60] A. A. Farley, “Selection of stockplate characteristics and cutting style for two dimensional cutting stock situations,” *Eur. J. Oper. Res.*, vol. 44, no. 2, pp. 239–246, 1990.

