Alvaro Luiz Neuenfeldt Júnior

The Two-Dimensional Rectangular Strip Packing Problem

A thesis submitted to Faculdade de Engenharia da Universidade do Porto in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Industrial Engineering and Management

Department of Industrial Engineering and Management
Faculdade de Engenharia da Universidade do Porto
2017

This research was supported by the BE MUNDUS project awarded by the European Commission Erasmus Mundus programme, by the Brazilian National Council for Scientific and Technological Development and by the Portuguese Foundation for Science and Technology.

Forecasting is the art of saying what will happen, and then explaining why it didn't!
Anonymous (communicated by Prof. Balaji Rajagopalan)

Acknowledgments

Firstly, I would like to thank Professor José Fernando Oliveira for his continuous support and dedication since my first moment in Porto. For my supervisor Professor Elsa Silva, any formal thanks will not be enough to reflect my gratitude for your dedication to me over the past three years. Elsa, your intelligence, and ability to work, as well as your ability to treat people is, was and will be an inspiration to me.

I would also like to thank Professor Maria Antónia Carravilla for the contributions in the first research conducted in this thesis, and Professor Carlos Soares for the contributions and knowledge about data mining and machine learning concepts, which were fundamental to develop the regression analysis. Finally, a special thanks to Professor António Miguel Gomes for the support, time spent and dedication to the development of the thesis. I am sure that your recommendations, as well as our discussions, concordances, and disagreements during the meetings will not be forgotten by me anytime soon.

I am grateful for all my research colleagues from CEGI for the reception and friendship in these years. I will miss our daily living. To Portuguese and Brazilian friends, know that in some way you are part of this thesis.

My friend and Professor Julio, thank you for inspiring me to follow this journey. His friendship made me a better person, and his support was fundamental for me to opt for science as a career option.

For my parents, Alvaro and Cleci, my parents in law, Aleir and Mary, my sisters, Rafaelli and Manuelli, my brothers in law, Sidnei and Rafael, and my grandparents, Edna, Leda and Ildo, thank you for understanding my choice of living in Portugal in the last three years. I know that it was not easy for you to accept my choice, and probably I have changed in some aspects over these three years, but my essence remains the same. Be assured that in both good and bad times I remembered his teachings and advices. Sabine, les mots écrits ici ne vont pas exprimer ce que je ressens pour vous. Sans vous ce moment n'aurait jamais été possible. Merci. Je t'aime!

Last but not the least, I acknowledge the European Commission Erasmus Mundus programme, the Brazilian National Council for Scientific and Technological Development, and the Portuguese Foundation for Science and Technology for providing the financial support for this thesis.

Abstract

In the rectangular Two-Dimensional Strip Packing Problems (2D-SPP) it is expected to find the best pattern in the arrangement of small rectangles into a strip with fixed width and virtually infinite height. These problems are often encountered in different industrial processes, such as the cutting of a solid material into smaller parts or the packing of items in empty packages. The objective is to minimize the required height to pack all small rectangles completely inside the strip, allowing 90 degrees rotation of the rectangles.

As any combinatorial optimization problem, the 2D-SPP is a complex problem, that can be solved by exact approaches, usually based on mathematical programming models, which can guarantee the optimality of the solution. Another possibility for solving the 2D-SPP are the approximation methods, such as heuristics and metaheuristics.

Over the years, heuristics were extensively explored for solving different variants of the 2D-SPP, given the efficiency finding good solutions avoiding long computational times, especially in problems with a large number of rectangles. Heuristic algorithms do not guarantee optimality and do not provide any information on the quality of the solutions found. The lower bounds are the most traditional alternative of reference value to evaluate the quality of the solutions obtained with heuristics, and are also used as stopping criterion for improvement heuristics. The area lower bound is the only lower bound available in the 2D-SPP literature which considers rectangles' rotation. The main disadvantage is the lack of accuracy of the area lower bound, mainly when much waste space between the rectangles in the strip are allowed.

This fact served as motivation to develop this thesis, which aims to explore the feasibility of developing a methodology based on data mining and machine learning concepts to obtain reference values for cutting and packing problems. In specific, a "pilot test" for this type of approach was conducted with the development of a methodology to predict the strip height for the rectangular 2D-SPP with 90 degrees rotations.

The main focus of this research thesis is to conduct the development of a regression analysis using the data mining process and supervised machine learning techniques. In a general form, three basic input data must be defined: (1) the observations, that are problem instances; (2) one known response variable, which is the objective reference value of the problem, in this research the strip height; and (3) and the explanatory variables, provided by the quantification of relevant characteristics of the problem.

It is important to emphasize that the developed methodology, based on data mining and machine learning concepts, to predict a reference value for the rectangular 2D-SPP, can be generalized for other types of cutting and packing problems.

Resumo

Nos problemas de corte e empacotamento bidimensional retangular em faixas é esperado encontrar o melhor padrão na distribuição de retângulos pequenos numa faixa retangular com largura fixa e altura virtualmente infinita. Estes problemas são frequentemente encontrados em diferentes processos industriais, seja no corte de um material sólido em partes mais pequenas ou no empacotamento de peças em embalagens vazias. O objetivo é minimizar a altura necessária para empacotar todos os retângulos completamente dentro da faixa, permitindo a rotação dos retângulos em 90 graus.

Como em qualquer problema de otimização combinatória, o empacotamento em faixas é um problema complexo, que pode ser resolvido através de métodos exatos, baseados em modelos de programação matemática que garantem a optimalidade da solução encontrada. Outra forma de resolver o empacotamento em faixas é através do uso de métodos de aproximação, como heurísticas e metaheurísticas.

Ao longo dos anos, as abordagens heurísticas foram extensivamente exploradas para resolver diferentes versões do empacotamento em faixas, devido à sua eficiência em encontrar soluções de boa qualidade com baixos tempos computacionais, principalmente em problemas com um elevado número de retângulos. No entanto, as heurísticas não garantem a optimalidade, nem fornecem qualquer informação sobre a qualidade das soluções encontradas. Os lower bounds são a alternativa mais tradicional de valor de referência para avaliar a qualidade das soluções obtidas com a heurística, e também como critério de paragem em heurísticas de melhoria. Porém, o único lower bound disponível para o empacotamento em faixas bidimensional que permite a rotação dos retângulos, o lower bound da área, não apresenta valores com precisão nos casos em que muitos espaços livres no arranjo dos retângulos dentro da faixa são permitidos.

Este fato serviu como motivação para o desenvolvimento desta tese, que tem por objetivo explorar a viabilidade de desenvolver uma metodologia baseada em conceitos da mineração de dados e machine learning para obter valores de referência para os problemas de corte e empacotamento. Foi adotado como teste-piloto para este tipo de abordagem o desenvolvimento de uma metodologia para prever a altura da faixa para o problema de empacotamento em faixas bidimensional que admite rotações de 90 graus.

O escopo da investigação está centrado no desenvolvimento de uma análise de regressão, a partir do processo de mineração de dados, através do uso de técnicas supervisionadas de machine learning. Para tanto, três dados de entrada necessitam ser definidos: (1) as observações, que são instâncias do problema; (2) uma variável de resposta conhecida, que é o valor de referência objetivo do problema, no caso a altura da faixa; (3) e as variáveis explicativas, fornecidas pela quantificação de características relevantes do problema.

É importante ressaltar que a metodologia desenvolvida, com base em conceitos da mineração de dados e machine learning, para prever um valor de referência para o empacotamento em faixas bidimensional retangular pode ser generalizado para outros tipos de problemas de corte e empacotamento.

Contents

Chapter 1

Motivation and Overview

1.1. Problem setting and statement

Currently, the industrial competitiveness requires more financial investments to optimize the production process, including industries with cutting or packing of materials in the productive process. The reduction of raw material waste during cutting or packing operations represent a cost reduction in different industrial sectors (e.g. the warehouse space required to stock raw material).

Regardless the industrial context, the cut of solid materials in smaller parts or packing of items in empty spaces are optimization problems with the same logical structure. The main objective is to find the best assortment of small items (or small pieces) in the large objects (large raw material surfaces), that minimizes the waste satisfying all problem constraints.

The cutting and packing problems are characterized as NP-hard (Garey and Johnson, 2002), therefore solving cutting and packing problems using exact methods is computationally expensive, especially when instances involve a large number of items to be packed in the large objects.

Cutting and packing problems can appear in different applications and can have different objectives and constraints, the literature about the problem is vast. In order to classify the diferent versions of the problem, an important contribution was given in the typology proposed by Wäscher et al. (2007). In the typology for cutting and packing problems different criteria are used to classify the related problems types in categories. These criteria were divided in: (1) dimensionality; (2) kind of assignment; (3) assortment of small items; (4) assortment of large objects; and (5) shape of small items. The dimensionality considers the number of relevant dimensions (1D, 2D, or 3D) of the problem. The kind of assignment is related with objective function orientation (the output value maximization or input minimization value). Variations of the shape and size of the items (identical items, weakly heterogeneous, or strongly heterogeneous) defines the assortment of small items. In a similar way, variations of the size and shape of one objects (with one dimension fixed and one or more variable dimension) or several large objects with fixed dimensions (identical objects, weakly heterogeneous or strongly heterogeneous) are considered for the assortment of large objects. Finally, the items can have regular (e.g. rectangles or circles) or irregular shapes.

In this thesis we dealing with a rectangular Two-Dimensional Open Dimension Problem (2D-ODP), in which the kind of assignment is to minimize the space needed in a single object to pack all items in the object without overlaps. The object has one dimension fixed and another variable, and the assortment of items is defined arbitrarily between

the weakly and strongly heterogeneous options. The rectangular Two-Dimensional Strip Packing Problem (2D-SPP) is a specific type of ODP, in which the object (named as strip) width W is fixed and the height H is unlimited. Therefore, for a set of items (named as rectangles) ($r = 1, 2, ..., n$) with a maximum dimension ($d1_r$) and a minimum dimension ($d2_r$), the main objective is to minimize the strip height H required to pack all rectangles. An example with nine rectangles ($n = 9$) detailing the maximum ($d1_4$) and minimum ($d1_4$) dimensions of the rectangle 4 can be found in Figure 1.1.

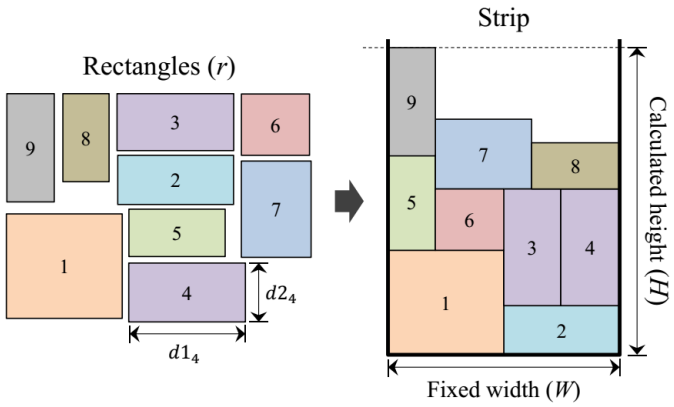


Figure 1.1 – A general view of the rectangular 2D-SPP.

In this case, some constraints must be considered when solving the 2D-SPP:

- The rectangles can not overlap;
- The rectangles have to be fully contained in the strip;
- The rectangles can be rotated by 90 degrees;
- The rectangles must be packed in orthogonal positions;
- The type of cut is non-guilhotinable.

Different versions of the 2D-SPP applications can be found in many industrial contexts, such as in the automation process to cut textile or paper rolls of raw material in small parts. The rolls are replaced by rigid plates to cut metal, glass, wood, marble, or plastics. Non-typical applications are related with data packing computational process for different technological situations. In Very-Large-Scale Integration (VLSI) integrated circuits, one of the most important problems is to pack a set of modules (data) within the rectangular domain (*chip*). The challenge is to minimize the total area used per *chip*, which is crucial to obtain production costs reductions (Pisinger, 2007). The 2D-SPP context is also found in the multidimensional time representation for scheduling problem with limited resources (Castro and Grossmann, 2012).

The 2D-SPP can be solved using (1) exact methods; (2) approximate or heuristics methods; or (3) hybrid methods. To evaluate the performance of these methods, numeric problem instances are a useful source of information to represent particularities and characteris-

tics found mainly in 2D-SPP practical applications, such as the total number of rectangles and the rectangles/strip shape variations.

The exact methods, based on mathematical programming models, provide a guaranteed optimal solution, or at least knowledge about the solution quality. In cases in which the problem instances have a large number of rectangles, a fact routinely found in real-world industry applications, the exact methods are not efficient, requiring long computational process times. The hybrid methods combine exact and heuristic methods taking advantage of the strengths of the heuristics and exact methods, improving the resolution efficiency and the solutions quality.

Heuristics are able of finding good solutions, requiring lower computational processing times when compared with the exact methods. However, the heuristics are not able to guarantee the optimal solution or the distance between the solution found and the optimal solution, being necessary to find the best cost-benefit between the computational process time required to pack the items in the objects and the expected quality of the solutions. Heuristic methods for the 2D-SPP can be divided in constructive heuristics and improvement heuristics procedures. In constructive heuristics the rules for packing the rectangles in the strip are defined. The improvement heuristic procedures start with an initial solution, usually obtained by applying a constructive heuristic, then the solution is improved by applying small consecutive changes over the rectangles' sequence or over the rectangles' arrangement in the strip, until a stopping criterion is met.

Different stopping criteria can be used to determine when to finish the heuristic iterations. Traditionally, the stopping criteria are related with a maximum number of iterations or a maximum computational ime consumed. More sophisticated stopping criterion options are related with specific evaluation measures, defined according to the improvement heuristic adopted (e.g. the halting criterion for simulated annealing, the number of searches for tabu search, or the population homogeneity for genetic algorithms). Finally, another stopping criterion option is to identify the solutions stagnation at each iteration, verified when no significant convergence evolution between the last solutions and the current solution is found.

Independent of the stopping criterion adopted, the main difficulty is to define a consistent and accurate stopping criterion value, which requires a deep knowledge about the characteristics of the problem instances to be tested. Also, probably the value defined will work well only for specific problem instances, and cannot be used to measure the quality of different problem instances.

To avoid the use of arbitrary values, known reference values are useful to compare and measure the quality of the computed solutions, and to provide a more accurate stopping criterion for the improvement procedure. These reference values should be represented by feasible or, also, infeasible solutions for each problem instance. As mentioned before, the guarantee that the solution is optimal can be found only with exact methods, which is not possible for problem instances with a high number of rectangles. For these situations, the alternative is to find approximate solution values.

Lower bounds are widely used as stopping criterion for combinatorial optimization problems, in which a given objective function must be minimized. In heuristics improvement procedures, the lower bounds are derived from adaptations of bounds developed for

exact methods, or based on linear combinations between the rectangles dimensions and the fixed strip dimension. Almost all lower bounds, i.e. Martello et al. (2003), Bekrar et al. (2007), Boschetti and Montaletti (2010), Alvarez-Valdes et al. (2009), were developed or adapted only for 2D-SPP without rectangle rotation. These lower bounds cannot be used as reference values for rectangular 2D-SPP that allows the rectangles' rotation.

The area lower bound proposed by Martello et al. (2003) is the only lower bound available in the 2D-SPP literature which considers rectangles' rotation. The area lower bound is obtained by considering the maximum value between the continuous lower bound, which is calculated using the total area of all rectangles divided by the strip width, and the maximum $d2_r$ between all rectangles of a problem instance is defined as the height lower bound. The main disadvantage is the lack of accuracy of the area lower bound, mainly for problem instances where the optimal solution has high waste space between the rectangles in the strip.

To illustrate this case, Figure 1.2 shows two examples of area lower bound applications compared with optimal solutions. For the first example (on the left), waste spaces are not found in the optimal solution (OS), the area lower bound (L_0) works very well, and the optimal solution value is equal to the area lower bound ($OS = L_0$). For the second example (on the right), a large quantity of waste spaces between the rectangles are available, and it is notorious the difference between the area lower bound and the optimal solution.

Therefore, the motivation for conducting this thesis is to obtain an accurate reference value of the strip height for the 2D-SPP with rectangles' rotation, given the impossibility of using the optimal solution as the reference value for most of the problem instances and the lack of accuracy of the area lower bound.

This chapter is organized in the following way. This first section characterizes the problem settings, illustrating its practical relevance and the motivations to conduct this thesis. Then, Section 1.2 describes the objective and the methodologies applied to answer the research motivation. Finally, Section 1.3 presents the structure of the thesis and describes the contents of each chapter.

1.2. Research objective and methodology

Although cutting and packing problems have been widely studied over the years, the use of data mining and machine learning concepts was not well explored in the literature. Inserted in the knowledge discovery context, data mining and machine learning are multidisciplinary concepts used to analyse a problem. Generally, a data mining process, using machine learning techniques, manipulates huge volumes of information to understand the input data structure, looking for patterns in order to find associations and systemic relations for further analysis, benefited of a high logical data processing capacity provided by computer systems (Fayyad et al., 1996; Kuhn and Johnson, 2013).

Data mining and machine learning concepts in cutting and packing problems are related with the process of converting problem information into measurable factors, in order to reflect the main problem characteristics and compare the algorithms performance with different types of problem instances.

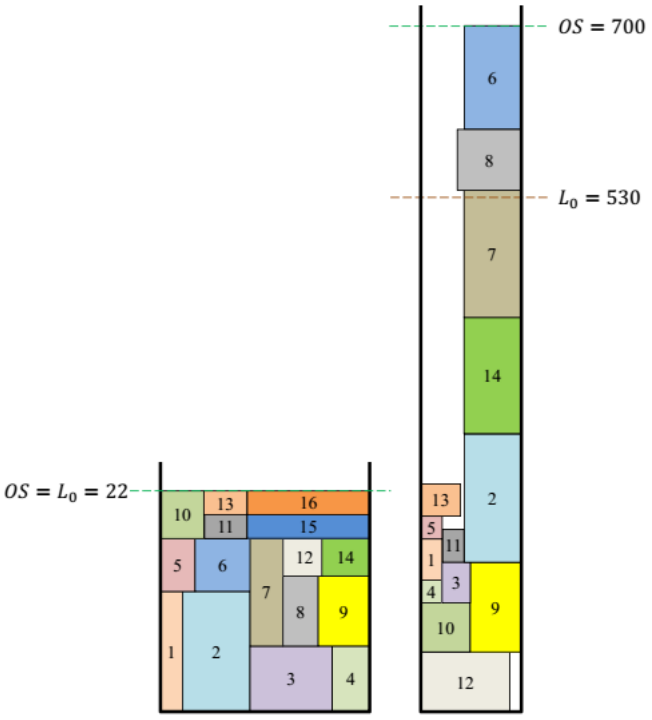


Figure 1.2 – Two examples of area lower bound (L_0) and optimal solution (OS).

In Smith-Miles and Lopes (2012) an extensive study on the most relevant characteristics required to measure algorithm performance was conducted for different combinatorial optimisation problems, including knapsack and bin packing problems. A methodology for heuristic selection for each particular instance of the knapsack problem is proposed in Hall and Posner (2007) and a regression model was also fitted to predict the relative solution procedure performance, considering knapsack problem attributes.

A relevant data mining approach is related with the simplification of the problem structure, aiming a better characterisation of different problem instances (Perez et al., 2004). López-Camacho et al. (2010) developed a method to replace an intuitive process to select the most important features of the irregular two-dimensional bin packing problem. Also for the bin packing problem, a linear correlation analysis was conducted in Santoyo et al. (2015) aiming the problem structure simplification for problem instances difficulty characterisation.

Machine learning techniques were also used in the knowledge discovery methodology, with the aim of better understanding the 1D and the 2D irregular bin packing problems' structure in order to influence the development complex hyper-heuristics algorithms (López-Camacho et al., 2013).

The main objective of this thesis is to explore the feasibility of developing a methodology based on data mining and machine learning concepts to obtain reference values for cutting and packing problems. This alternative reference value can be used to assess the quality of solutions found by heuristics, and also as stopping criterion in improvement heuristics, to avoid long computational times. This thesis is the starting point of this type

of approach, which has as "pilot test" the development of a methodology to predict the strip height for the rectangular 2D-SPP with 90 degrees rotations. However, as mentioned before, this methodology can be generalized for other types of cutting and packing problems.

By going one step further, the scope of this research lies on the development of regression analysis based on a data mining process using supervised machine learning techniques. These techniques are useful to analyze the behavior of an initial group of observations in order to predict with accuracy the strip height for new observations. In a general form, three basic input data must be defined to develop regression analysis: (1) the observations; (2) one known response variable; and (3) explanatory variables. In the following paragraphs it will be detailed the regression analysis input data.

The observations are used to refer a set of single and independent unit of data about the problem. For the 2D-SPP, the observations are problem instances. These predefined problem instances should satisfy the properties and different variants of the 2D-SPP found mainly in real-world applications.

The known response variable is the reference value defined as objective of the problem. For the 2D-SPP, the known response variable is given by the strip height, calculated for each problem instance using a heuristic together with a local search algorithm to improve the heuristics' solutions.

A previous knowledge about the format (e.g. binary or continuous variable) of the known response variable allows identifying which techniques have the greatest potential to predict with accuracy the response variable for new problem instances. Additionally, a measure of error (e.g. mean-squared-error) is used as external supervisor to evaluate the differences between the known response value and the predictions of this response value.

The explanatory variables are provided according to relevant characteristics of the problem, which can be measured quantitatively. The main objective is to provide an adequate way to measure the known response variable (strip height) for a predefined number of observations (problem instances).

An exploratory approach to study the 2D-SPP is useful to identify the main characteristics of the problem and to develop the explanatory variables for the regression analysis, representing the possible combinations between rectangles and strip shapes and the total number of rectangles to be packed.

In the development of explanatory variables, problem generators can be used as one source of information to study the problem characteristics, not only for the 2D-SPP, but for all cutting and packing problems. Additional explanatory variables found in other cutting and packing problems can be tested and adapted if correctly characterize the 2D-SPP. If the number of defined explanatory variables is considered high, a correlation analysis is useful to quantify and validate the similarity between variables, and a dimension reduction or a feature selection method can be used to reduce the complexity of the problem, taking advantage of the similarity relation between the explanatory variables.

Finally, the data mining approach uses the predefined group of observations (problem instances), the explanatory variables, and the known response variable (the strip height) to fit a regression model to predict the response variable for new problem instances. Different supervised machine learning techniques can be used to fit the regression model, in order to ensure its level of generalization and the predictions' accuracy for new problem instances.

1.3. Thesis synopsis

This thesis is paper-oriented, where Chapters 2, 3, and 4 consist of submitted papers to international journals. The thesis chapters are organized as presented in Figure 1.3. The Chapter 2 is related with a survey in heuristics for the 2D-SPP. In Chapter 3 characteristics of the 2D-SPP are studied to develop, in Chapter 4, the data mining based framework to assess solutions' quality. Also, Appendix A is used as support to understand better some concepts about data mining and machine learning techniques. In the following paragraphs a detailed description of each chapter is presented.

A literature review about the seminal and the most recent approaches (from the last decade) of the rectangular 2D-SPP is presented in Chapter 2. A survey on constructive heuristics, improvement heuristics with search over sequences, and improvement heuristics with search over layouts processes for the 2D-SPP is proposed. The relevant literature was reviewed and links between the most frequently used heuristics and the data characteristics were sought. The literature review, not only update the previous reviews, but also provides a classification for both constructive and improvement heuristics, considering common characteristics. All descriptions, concepts and conclusions obtained in Chapter 2 were fundamental to provide essential information about the 2D-SPP to begin the development of both Chapter 3 and Chapter 4.

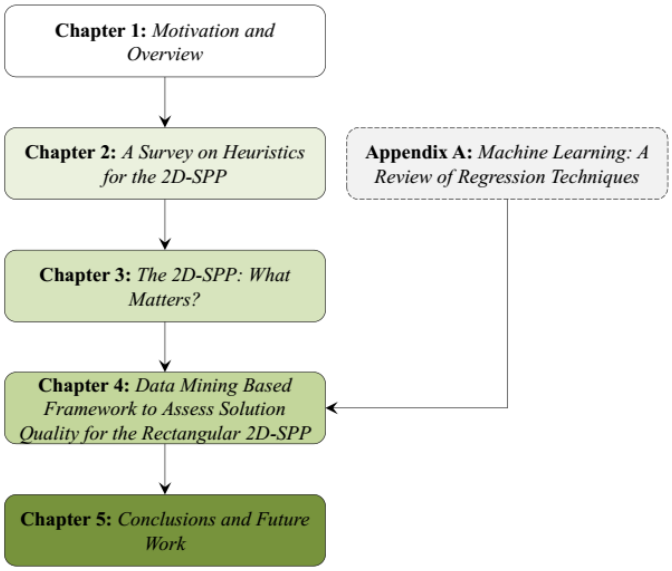


Figure 1.3 – Chapters of the thesis.

Chapter 3 is motivated by three important tasks. Firstly, an exploratory analysis was conducted to identify the main 2D-SPP characteristics, served as a base for the development of descriptive variables. Descriptions of some concepts about parameters of problem generators created for the 2D-SPP, together with studies about descriptive variables used in other cutting and packing problems were explored. Secondly, a methodology focused on reducing the quantity of descriptive variables to a manageable quantity was proposed, in order to define the explanatory variables to be used in the regression analysis. Finally,

the explanatory variables are used to analyse the distribution behavior of the most relevant benchmark problem instances used for the 2D-SPP found in the literature.

Chapter 4 is the core of the thesis and presents a data mining based framework to predict the strip height and to assess the quality of heuristic solutions for the 2D-SPP allowing 90 degrees rotation. Concepts and definitions described in Chapter 3 are incorporated as input data (explanatory variables, normalized gap response variable, and the problem instances) to fit the regression model. The Random forest was the data mining technique that betters adjusts the input data provided. The 5-fold cross-validation process and additional validation procedures were adopted to guarantee the predictions' accuracy and level of generalization of the regression model, using an additional dataset containing new problem instances compared with the problem instances used to fit the regression model.

The data mining framework efficiency, to predict the strip height for the 2D-SPP, was tested after the validation steps, using the predictions as stopping criterion in three different improvement procedures. Also, an analysis was conducted by varying the predictions values, to verify if small perturbations in the strip height predictions are capable to change substantially the local search stopping process behavior.

Chapter 5 summarizes the work contributions and provides the answers to the research questions raised. This chapter also presents new research directions. In Appendix A, a working paper is presented with the aim of complement the literature on data mining and supervised machine learning techniques concepts. Also, a practical guide to compute supervised machine learning techniques using *R* and *RStudio* software is explored in details.

References

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

Bekrar, A., Kacem, I., and Chu, C. (2007). A comparative study of exact algorithms for the two dimensional strip packing problem. *Journal of Industrial and Systems Engineering*, 1(2):151–170.

Boschetti, M. A. and Montaletti, L. (2010). An exact algorithm for the two-dimensional strip-packing problem. *Operations Research*, 58(6):1774–1791.

Castro, P. M. and Grossmann, I. E. (2012). From time representation in scheduling to the solution of strip packing problems. *Computers and Chemical Engineering*, 44:45–57.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park.

Garey, M. R. and Johnson, D. S. (2002). *Computers and intractability*, volume 29. WH Freeman.

Hall, N. G. and Posner, M. E. (2007). Performance prediction and preselection for optimization and heuristic solution procedures. *Operations Research*, 55(4):703–716.

Kuhn, M. and Johnson, K. (2013). *Applied predictive modeling*, volume 1. Springer.

López-Camacho, E., Terashima-Marín, H., Ochoa, G., and Conant-Pablos, S. E. (2013). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics*, 145(2):488–499.

López-Camacho, E., Terashima-Marín, H., and Ross, P. (2010). Defining a problem-state representation with data mining within a hyper-heuristic model which solves 2D irregular bin packing problems. In *Ibero-American Conference on Artificial Intelligence*, pages 204–213. Springer.

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

Perez, J., Frausto, J., Cruz, L., Fraire, H., and Santiago, E. (2004). A machine learning approach for modeling algorithm performance predictors. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 70–80. Springer.

Pisinger, D. (2007). Denser packings obtained in O(n log log n) time. *INFORMS Journal on Computing*, 19(3):395–405.

Santoyo, A. M., Ortega, J. P., Vargas, D. R., and Reyes, L. C. (2015). Towards a characterization of difficult instances of the bin packing problem. *IEEE Latin America Transactions*, 13(7):2454–2462.

Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

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

CHAPTER 2

A Survey on Heuristics for the 2D-SPP

José Fernando Oliveira* · **Alvaro Neuenfeldt Júnior*** · **Elsa Silva*** · **Maria Antónia Carravilla***

Abstract

Two-dimensional rectangular strip packing problems belong to the broader class of Cutting and Packing (C&P) problems, in which small items are required to be cut from or packed on a larger object, so that the waste (unused regions of the large object) is minimized. C&P problems differ from other combinatorial optimization problems by the intrinsic geometric constraints: items may not overlap and have to be fully contained in the large object.

This survey approaches the specific C&P problem in which all items are rectangles, therefore fully characterized by a width and a height, and the large object is a strip, i.e. a rectangle with a fixed width but an infinite height, being the problem's goal to place all rectangles on the strip so that the height is minimized.

These problems have been intensively and extensively tackled in the literature and this paper will focus on heuristic resolution methods. Both the seminal and the most recent approaches (from the last decade) will be reviewed, in a rather tutorial flavor, and classified according to their type: constructive heuristics, improvement heuristics with search over sequences and improvement heuristics with search over layouts.

Building on this review, research gaps are identified and the most interesting research directions pointed out.

Keywords

Strip packing problems, Cutting and packing problems, Heuristics.

2.1. Introduction

The Strip Packing Problem (SPP) aims to pack a set of small items inside a larger object, the container, with all dimensions but one fixed, with the objective of minimizing the free dimension of the large object. The small items cannot overlap each other and must be completely inside the large object. This description fits the definition of a Cutting and Packing (C&P) problem and indeed the SPP can be classified as an Open Dimension Problem

*INESC TEC, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4600-001 Porto, Portugal.

(ODP) according to Wäscher et al. (2007) typology for C&P problems. Other well-known C&P problems are the knapsack problem, the bin-packing problem, the container loading problem, the cutting-stock problem and the pallet loading problem.

The SPP can be found in many practical settings and has many real-world applications in industry and services, such as the cut of wood boards, steel plates or paper rolls and also multidimensional resource scheduling.

One of the most distinctive characteristics of C&P problems is the fundamental role of geometry in their definition and resolution. When dealing with cutting of raw-material or item packing, three dimensions (length, width, and height) are always present. However, for modeling and resolution purposes, C&P problems are usually classified in one-, two- and three-dimensional problems, when taking into account the number of dimensions that are relevant for the problem. One should bear in mind that the dimensions that are common to the small items and the large object are not relevant for the optimization problem. In this review two-dimensional SPP problems will be tackled.

Geometry also plays an important role in the description of small items' and large object's shapes. Two-dimensional problems are usually divided into rectangular, circular and irregular problems. The first two categories are self-explanatory and the last one refers to problems in which the small items have polygonal shapes, at least in the most common form of these problems. In this paper, problems in which both the small items and the large object are rectangles will be approached. Conjugating the rectangular shape of the problem with the open dimension characteristic, the large object will be a rectangle with a given width and an infinite height (hereafter called strip), and the objective of the problem is to minimize the actual height used to pack all small items.

As any combinatorial optimization problem, the SPP can be solved by exact approaches (in the sense that the approach provides a guaranteed optimal solution, being usually based on mathematical programming models), approximation methods (heuristics and metaheuristics) or hybrid methods (e.g. matheuristics), resorting to elements from both worlds. Given the complexity of these problems, heuristic methods, providing good solutions in a fairly small amount of time, have been rather popular in the field, being able to solve problems with many items.

This paper will focus on heuristic methods for the rectangular Two-Dimensional Strip Packing Problem (2D-SPP). The relevant literature will be reviewed and links between the most frequently used methods and the data characteristics will be sought.

Surveys have been published in the past on the SPP. In Hopper and Turton (2001b) a review of metaheuristic approaches, with a special emphasis on genetic algorithms but also looking at simulated annealing, tabu search, and artificial neural networks, can be found. Lodi et al. (2002) discuss exact and approximate methods for both the rectangular bin-packing problem and the rectangular SPP, giving special attention to the level (guillotinable) SPP. With the same scope, Ntene and van Vuuren (2009) discuss heuristics previously published in the literature, in order to provide a comparison basis for their own new approach. Riff et al. (2009) also review both exact and heuristic methods for the 2D-SPP and present some benchmark instances commonly used in the literature in the computational validation of algorithms for specific instances.

This survey will not only update the previous reviews but will also provide a classifica-

tion of both constructive and improvement heuristics, looking at characteristics they may have in common. Metaheuristic approaches will be referred and discussed along the text, but focusing on the underlying heuristics and not on the search methods that characterize and distinguish each one of them. The decomposition of the heuristics in their building blocks, together with the study of the past research effort that will be presented, will help to identify research gaps in the field.

The paper is organized as follows. Section 2.2 fully presents and characterizes the SPP. In Section 2.3 the heuristic methods found in the literature are presented, while in Section 2.4 the most relevant findings are presented and discussed. Finally, Section 2.5 presents some final remarks and future research ideas.

2.2. The strip packing problem

The SPP is a C&P problem that is classified as an Open Dimension Problem (ODP) under Wäscher et al. (2007) typology. However it can be further characterized, as the typology itself anticipates. Not only the dimensionality of the problem and the geometry can be taken into account (in this survey two-dimensional problems where the small items are rectangles will be reviewed), but further characteristics regarding the way how the rectangles are laid out on the strip can be considered for the problem categorization. The actual disposition of the rectangular small items on the strip is referred to as pattern (either cutting or packing pattern) or layout. The characteristics that may be imposed to the patterns arise from the practical constraints that the different real-world applications require. These characteristics are summarized in Figure 2.1.

The first level considers the rectangle list input. In the online version, rectangles are input one by one without any information regarding the following rectangle. Once the rectangle is placed on the strip there is no possibility of repositioning it (Ye et al., 2009).

Not knowing which will be the next rectangle arises for instance in scheduling problems, when a sequence of tasks or jobs have to be processed in a given number of machines but the tasks are not known in advance.

In fact, scheduling problems can be modeled as SPP if the height of the strip (the open dimension) is associated to time and the strip's width to the machine capacity. Each task requires a given amount of capacity and takes a given amount of time to complete, defining therefore a rectangle. The goal is to process all tasks in the minimum amount of time, i.e. to minimize the height of the strip (Hurink and Paulus, 2011).

However, the most frequent in the literature (and in real-world applications) is the offline case. In this situation it is possible to define criteria for the order by which the rectangles will be placed on the strip, as length, width, area or perimeter, once all characteristics of the rectangles are known in advance. Nevertheless, this ordering possibility is not necessarily used by the solution approaches and random orders are also frequent in the literature.

The second level concerns the geometry of the small items. In practical contexts SPP problems arise with small items having irregular shapes (e.g. polygons), circular shapes or rectangular shapes, which is the focus of this survey. Directly related to the geometry is orthogonality. In orthogonal patterns the edges of the rectangles are parallel to the edges

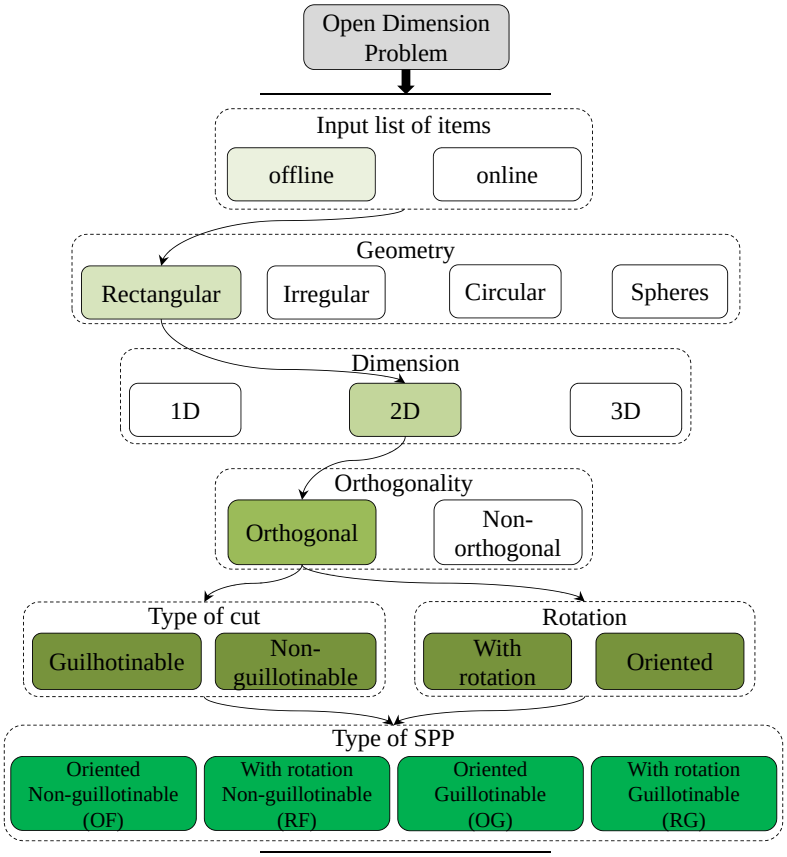


Figure 2.1 – Classification structure for the SPP.

of the strip. When the orthogonality constraint is not imposed, the problem is treated as an irregular SPP, even if the shapes are rectangular, and therefore it is out of the scope of this survey.

The following characteristic is the type of cut. When guillotine cuts are considered the pattern has to be formed so that it is possible to obtain the requested rectangles by cutting from edge to edge, either the strip's edges or the edges of the sub-rectangles generated by previous cuts. In fact the name arises from the applications where the cutting tool is a guillotine. This same constraint has to be imposed when patterns are to be cut by saws, such as in the wood industry. Patterns that do not have this property are designed as non-guillotinable patterns.

A further characterization of guillotinable patterns arises from the number of stages or phases required to fully cut the pattern. If, using only a succession of horizontal (or vertical) cuts, all rectangles are extracted from the strip, the pattern is called "one-stage guillotinable". If it is necessary to orthogonally rotate the parts resulting from the first stage cuts, and apply a second series of guillotine cuts (now vertical or horizontal), the pattern is designated as a "two-stage guillotinable pattern". Following the same line of reasoning a three-stage guillotinable pattern requires a further change in the cutting direction.

Patterns with no limit on the number of stages are called *n*-stage guillotinable patterns, where *n-1* holds for the number of cutting direction changes (Imahori and Yagiura, 2010). Guillotinable patterns may also be classified as level packing patterns if the first cuts are horizontal (across the width of the strip) and the rectangles are then extracted from the generated levels (Coffman Jr. et al., 1980; Mumford-Valenzuela et al., 2004). Each level has the height of the tallest rectangle placed there. Some examples of these patterns are represented in Figure 2.2.

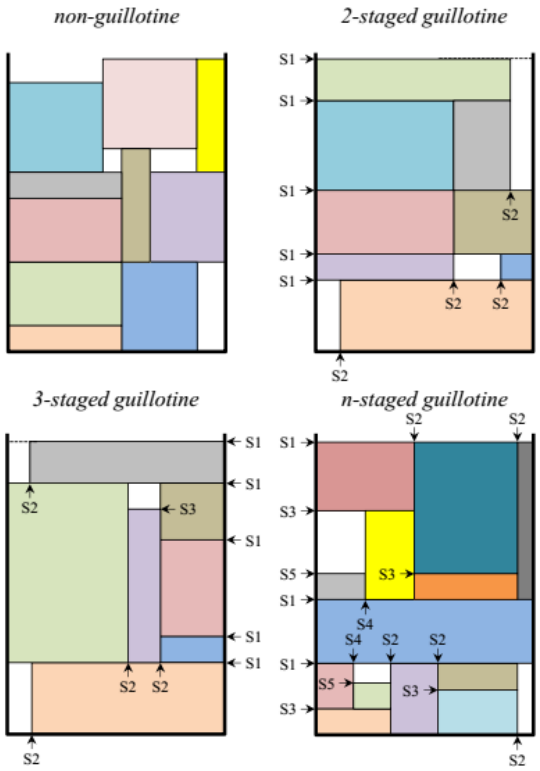


Figure 2.2 – Examples of cutting patterns.

Regarding the orientation of the rectangles, when the materials have vein or other physical properties that make the width and height directions different from each other (e.g. when cutting striped fabrics in the garment industry), the rectangles have to be cut as they are defined, i.e. width along the strip's width and height along the strip's height. However, there are materials that are isomorphic along the two directions and therefore the rectangles can be rotated, i.e. it is indifferent to cut the rectangle's width along the strip's width to the strip's height, and vice-versa. In this case it is said that the rectangles may be 90° rotated or more simply, rotated.

Combining the rectangle orientation constraints and the types of cuts, four sub-problems can be defined (last level of the characterization structure). The less constrained problem is the RF where the rectangles may be rotated and no constraints on the type of cuts are imposed. Fixing the rectangles' orientation defines the sub-problem OF. When imposing the guillotine cut constraint the sub-problems RG and OG are defined, respectively allowing

for the rotation of the rectangles or fixing their orientation, (Lodi et al., 1999). These two sub-problems may be further refined if the number of stages is considered, as previously explained.

Given the real-world origin of these problems, additional constraints and characteristics may appear in the literature, motivated by different cutting technologies and material properties, but the most commonly tackled are the ones described above.

2.3. Heuristics

Heuristic algorithms do not find guaranteed optimal solutions and are not able to recognize an optimal solution if one is found. Additionally, no distance to the optimal solution can be computed or guaranteed. However, given the low computational times, when compared to exact methods, they are considered to be an efficient resolution strategy for large instances of hard combinatorial optimization problems. Heuristic algorithms are usually divided into constructive heuristics and improvement heuristics. The most representative rectangular 2D-SPP heuristics will be described in detail under this division.

2.3.1 Constructive heuristics

Constructive heuristics build solutions by adding elements one by one. Only in the end of the construction process a complete solution is obtained and if the process is interrupted no feasible solution is generated. In the SPP context, the rectangles are successively placed on the strip's free/empty spaces or regions.

In general, the older constructive heuristics resort to simple placement mechanisms, with a low computational complexity, and therefore are less able to check for empty spaces in the strip. The improvement of this capacity became the most important trend in the development of the more recent heuristics.

When analyzing the solution construction methodologies used in the literature, four strategies were found: "positioning', "fitness", "level" and "profile". This classification is based on the conceptual idea behind the actual placement of the rectangles on the strip and further details are presented in the following sections.

2.3.1.1 Positioning-based heuristics

Positioning-based heuristics are the oldest in the SPP literature, and most common in the early works. They are rather flexible, allowing to incorporate the most usual constraints of the problem. The basic mechanism behind positioning-based heuristics is the identification of the free space on the strip that is most suitable for a given piece, according to an also given criterion.

The "Bottom-Left (BL)" heuristic, proposed by Baker et al. (1980), was not only the first positioning-based heuristic for the SPP but is also the most widely and well-known approach to this problem. The goal is to place each rectangle the lowest and to the left as possible, as illustrated in Figure 2.3. An initial, feasible position is assigned to the

rectangle, and then it is alternately moved down and to the left. A direction change only occurs when moving along the current direction becomes unfeasible.

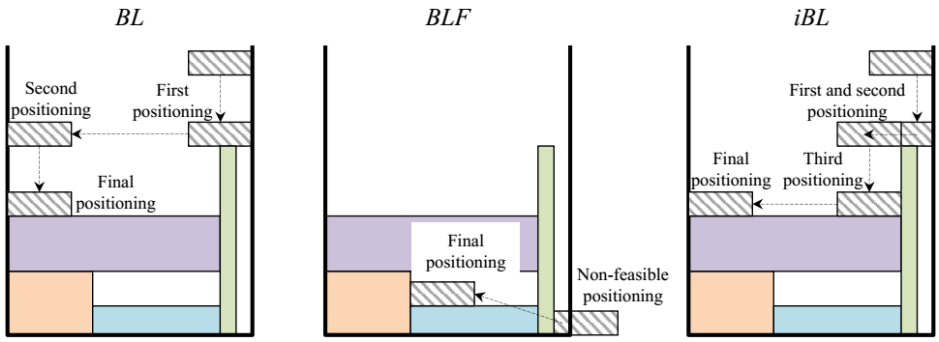


Figure 2.3 – Examples of application of the Bottom-Left (BL), Bottom-Left-Fill (BLF) and improved Bottom-Left (iBL) placement strategies.

With a complexity $O(n^2)$ (where n is the number of rectangles), the main advantage of the bottom-left strategy is being very fast. However, one of its main drawbacks is that it is unable to fill "holes", i.e. empty spaces that are surrounded by rectangles that were previously placed. To overcome this limitation several modified versions of the bottom-left heuristic were proposed along the years. The most relevant one is the "Bottom-Left-Fill (BLF)", developed by Chazelle (1983).

As in the bottom-left heuristic, the bottom-left-fill heuristic aims to place the rectangles the lowest and to the left as possible. However, as illustrated in Figure 2.3, spaces between already placed pieces are considered to be admissible. The search for empty feasible (in the sense that the rectangle fits the space) spaces is much more complex and therefore the bottom-left-fill heuristic has a complexity of $O(n^3)$, but for the same sequence is always equal or better than the bottom-left heuristic. The implementations of the bottom-left-fill heuristic places the rectangle on a position that is not feasible, because it overlaps other rectangles that had been already placed (e.g. at the bottom-left corner of the strip), and then moves the rectangle right and up until a feasible position is found (Figure 2.3).

The "Improved Bottom-Left (iBL)" heuristic, proposed by Liu and Teng (1999), gives always priority to the down movement, i.e. when it is not possible to move the rectangle downwards it moves it to the left, but just the distance necessary to make again a downward movement (Figure 2.3).

Common to these and other positioning-based heuristics is their dependence of the order by which the rectangles are placed, as the focus of these approaches is to find the best placement for a given rectangle.

In order to decrease this dependence from the rectangle input order Zhang et al. (2016b) have very recently further developed a recursive heuristic previously proposed by the same authors (Zhang et al., 2006), the "Priority Heuristic (PH)", and dynamically assign a priority to each rectangle. This recursive algorithm successively divides the strip into rectangles and sub-rectangles and when choosing which rectangle to place in a given empty rectangle, this choice does not depend just on the rectangle input sequence but on how well the rectangle fits the space (the "priority"). In Zhang et al. (2007) the authors have already

resorted to the recursive algorithm proposed in Zhang et al. (2006), but decomposing the strip into layers and filling each layer by a simple positioning-based constructive heuristic.

In the nineties Dowsland et al. (1998) have proposed a rather innovative heuristic for the irregular 2D-SPP, named "Jostle". In this heuristic a bottom-left-fill and a up-right-fill (an artificial top edge is defined for the strip) heuristic are alternately applied, "throwing" the pieces up and down. But the piece sequence is defined by the position of each piece in layout generated in the previous iteration. When applying the bottom-left-fill heuristic, the pieces are ordered by decreasing y-coordinates, regarding the previous layout, while when applying the top-right-fill heuristic, the pieces are ordered by increasing y-coordinates, also regarding the previous layout. The general idea is that the pieces that are farthest away in the previous layout should now be placed first. The possibility of filling holes in the middle of the already placed pieces has revealed to be critical for the efficacy of this approach. Wauters et al. (2013) test this idea on the rectangular 2D-SPP and name the algorithm "Improved Deepest Bottom-Left-Fill (iDBLF)". Alternative ways of generating sequences from the actual layouts are proposed and tested, as well as alternatives to the bottom-left-fill/top-right-fill heuristics.

2.3.1.2 Fitness-based heuristics

Fitness-based heuristics are focused on the free spaces, i.e. the best-fit between the rectangle to place and the empty spaces is sought. It is common to give some priority to the spaces closer to the edges of the strip (left, right, or bottom).

The first constructive heuristic that has explored this mechanism was the "Best-Fit (BF)" heuristic by Burke et al. (2004). The selection of the rectangle to place is dynamic and depends on the space to be filled. The lowest free space is selected and the first choice will be a rectangle that perfectly fits this space. If there is no rectangle with this characteristic, the largest rectangle that fits the space is selected and placed according to one of three different policies: leftmost policy; next-to-the-tallest policy; and next-to-the-shortest policy (Figure 2.4).

In the leftmost policy the rectangle is always placed at the left of the space, while in the other two policies the placement inside the space depends on the previously placed rectangles. In the next-to-the-tallest policy the side of the empty space where the tallest rectangle is, is chosen, while in the next-to-the-shortest policy it is the other way around.

The implementation proposed by Burke et al. (2004) has a time complexity of $O(n^2 + nW)$. However, Imahori and Yagiura (2010) propose an implementation based on balanced binary search trees with a time complexity of $O(n \log n)$. This efficient implementation of the best-fit heuristic decomposes the "skyline" formed by the already placed rectangles into line segments and stores them in a heap structure using their y-coordinate as the key to access the memory structure (Figure 2.5). At the same time these same segments are stored in a doubly linked list ordered by their x-coordinates. The authors not only prove that the time and space complexity achieved are optimal, but also provide a "Worst-Case approximation ratio (W-C appr.)" for the best-fit heuristic. Computational experiments demonstrate the capacity of efficiently solving extra-large instances ($n > 1,000,000$). Verstichel et al. (2013) use these data structures in their implementation of the best-fit algorithm, which is named

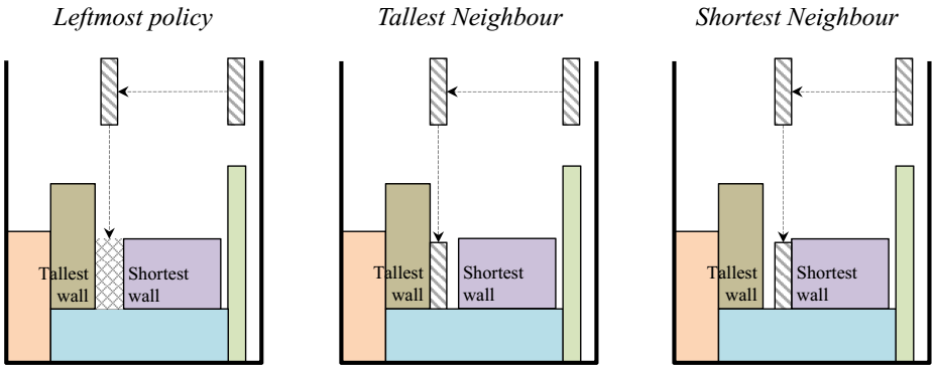


Figure 2.4 – Example of application of the Best-Fit (BF) heuristic.

"Three-way Best-Fit (T-w BF)" heuristic, and add three new placement criteria: place the fitting rectangle at the right-hand side of the space; place the fitting rectangle so that the difference of the top edge with its neighbour is maximal; place the fitting rectangle so that the difference of the top edge with its neighbour is minimal.

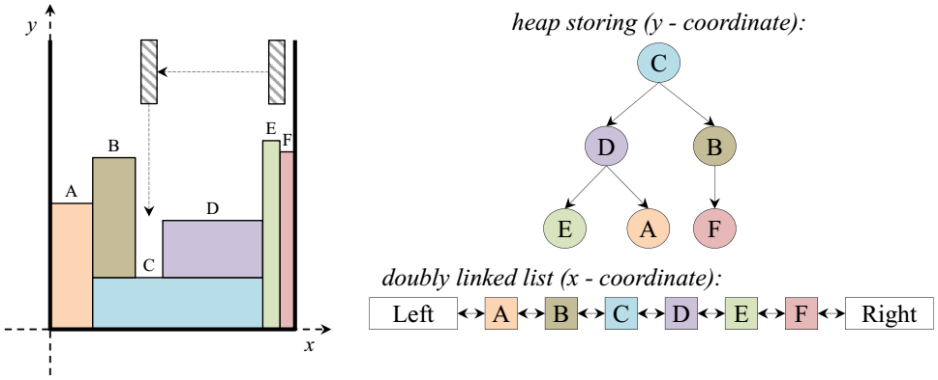


Figure 2.5 – Memory structures to efficiently implement the best-fit heuristic.

Another improvement to the best-fit heuristic was proposed by Aşık and Özcan (2009): the "Bidirectional Best-Fit (BBF)" heuristic. In this approach not only the spaces among already placed rectangles are considered as feasible placement places for the following rectangles, but also what the author calls vertical spaces, i.e. spaces on the top of previously placed rectangles and with no side neighbour rectangles, are considered. The space is closed at the top by what is designated as the "expected best height" (Figure 2.6). The "expected best height" is the lower bound on the height and is obtained by dividing the sum of the areas of all the rectangles by the width of the strip. Vertical niches are also identified giving to the heuristic a spatial two-dimensional awareness that the original proposal did not have, once it is focused on the x-coordinate. In Özcan et al. (2013) the authors extend this work to the "Modified Bidirectional Best-Fit (BBFM)" heuristic, by placing the rectangles on the strip in groups, which were created by looking at the geometric similarities of the rectangles, instead of placing them one by one.

As usually rectangles do not perfectly fit the available empty spaces, the approaches de-

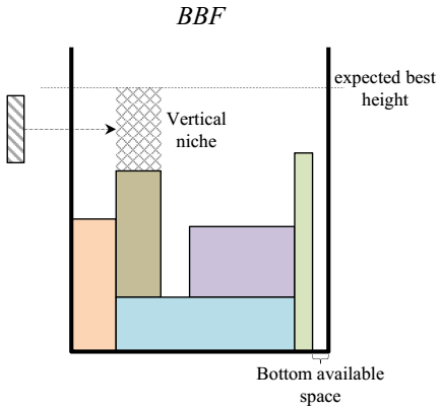


Figure 2.6 – Example of application of the Bidirectional Best-Fit (BBF) heuristic.

scribed until now tend to generate many small spaces that, for large instances, deteriorate significantly the efficiency when seeking for the best placement for a given rectangle. To overcome this drawback Leung and Zhang (2011) developed the "Fast layer-based Heuristic (FH)" that aims to keep the layout skyline as flat as possible. To achieve this, the placement strategy favors placements that eliminate (or least increase) the number of edges of the skyline.

To achieve the flat skyline goal, the quality of the relationship between a free space and a rectangle to be placed is assessed and is designated as the *fitness value*. This measure takes values between a maximum of 4, when there is a perfect match between the space and the rectangle, and a minimum of 0. In Figure 2.7 examples corresponding to the fitness values of 0 to 3 are presented. In each layout construction step the fast layer-based heuristic selects the pair rectangle/space that maximizes the fitness value.

Several authors have worked on improving this scoring methodology. Leung et al. (2011) in the "Intelligent Search Algorithm (ISA)" resort to 5 scoring values, between 0 and 4, as does Chen et al. (2015) ("Hybrid Demon Algorithm (HDA)"), while Yang et al. (2013) in the "Simple Randomized Algorithm (SRA)" considers 8 scoring values, between -1 and 6, as does Wei et al. (2016) ("Efficient Intelligent Search Algorithm (IA)") that also uses a 8 value scoring system. It is worthwhile to notice that these different scoring values correspond to different valuations of different configurations of the skyline.

A similar idea, of flattening the skyline, is followed by Kotov and Cao (2011) that restrict the potential placement points to the concave vertices of the skyline, aiming its flatness.

2.3.1.3 Level-based heuristics

The central idea of these heuristics is the placement of the rectangles on levels, i.e. parallel guillotine cuts across the width of the strip. The height of each level i is defined by the tallest rectangle h_i placed on that level. The layouts generated by these heuristics are not comparable with the ones generated by the heuristics described in the previous sections. They address specific real-world problems in which the level orientation of the layouts is a

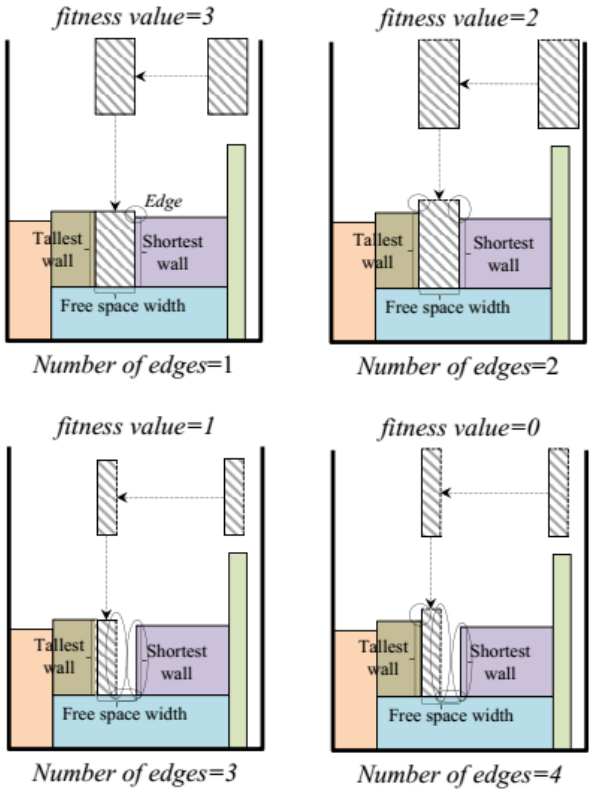


Figure 2.7 – Examples of fitness values computed by the Fast layer-based Heuristic (FH).

constraint, as for instance when planning the display of goods on supermarket shelves.

Proposed by Coffman Jr. et al. (1980) the "Next-Fit Decreasing Height (NFDH)" heuristic sorts the rectangles by non-increasing height and places them one by one on the currently open level and at the leftmost admissible position. The currently open level is closed when it does not have enough space to accommodate the candidate rectangle, a new level is opened and the candidate rectangle is placed on this level, next to the left edge of the strip (Figure 2.8).

A clear limitation of this heuristic, and a source of space waste, is that free spaces of closed levels cannot be later on used by smaller rectangles. The "First-Fit Decreasing Height (FFDH)" heuristic, proposed by the same authors (Coffman Jr. et al., 1980), addresses this drawback by not closing levels, i.e. each rectangle is placed at the lowest possible level as long as it fits. As depicted in Figure 2.8, a new level is created only if the rectangle does not fit any previous level. A variation of this heuristic was proposed by Berkey and Wang (1987): the "Best-Fit Decreasing Height (BFDH)" heuristic. In this approach the rectangle is not placed at the lowest level where it fits, but at the level, among those where it fits, for which the unused horizontal space is minimum.

The previous heuristics sort the rectangles by non-increasing height, as their names suggest. Therefore, instances where consecutive rectangles, which most probably will be placed on the same level, have very different heights lead to a very poor performance of these approaches as the taller rectangles will define a height for the level that can not be

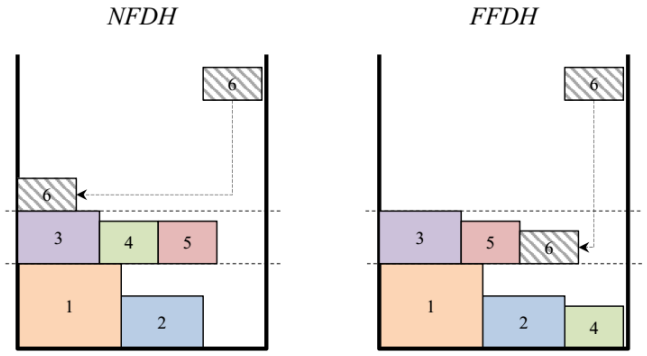


Figure 2.8 – An example of application of the Next-Fit Decreasing Height (NFDH) and the First-Fit Decreasing Height (FFDH) heuristics (the numbers represent the order by which the rectangles were placed).

afterwards met by the shorter ones. It should be kept in mind that these heuristics are intrinsically uni-dimensional, trying to make good choices along the width, and only look at the height to determine if the rectangles fit or not in the level.

To address this poor behavior Ntene (2007) have more recently developed the "Size Alternating Stack (SAS)" heuristic. The rectangles are divided in two lists (L_1 and L_2). List L_1 consists of all rectangles that are strictly taller than wider (*narrow* rectangles), and list L_2 has the rectangles that have a width equal or greater than the height (*wide* rectangles). The first list is ordered by decreasing height while the second list is ordered by decreasing width. The main idea behind the size alternating stack heuristic is to alternate between *narrow* and *wide* rectangles. Each level is initiated by comparing the heights of the first rectangles of each list and the tallest one is selected for placement, hence defining the level height. The following rectangle is taken from the list that did not open the level and additionally rectangles from this list are placed on the top of each other, forming a column, until the level height is reached. Only afterwards a new rectangle is taken from the alternate list. The resulting layout is a set of stacks of rectangles organized by levels. In Ortmann et al. (2010) this work is further extended and combined with ideas from Ntene and van Vuuren (2009), originating the "modified Size-Alternating Stack algorithm (SASm)", and with the floor ceiling heuristic from Lodi et al. (1999), resulting in the "Stack Ceiling with Re-sorting algorithm (SCR)".

A similar extension of the idea of level is used by Cui et al. (2008). The strip is divided into levels (sections, according to the terminology of these authors), the height of each level is equal to either the height or the width (rectangles may be rotated) of the rectangle placed at the bottom-left corner of the level, but the remaining region of the level may be freely occupied by any set of rectangles, without any additional constraint. A recursive function is proposed to find the "best" occupancy of this region. This approach is further developed in Cui et al. (2013) in the "Sequential Grouping and Value Correction Procedure (SGVCP)" by assigning to each rectangle a value not proportional to its area, trying to guide the recursive function towards the use of arrangements that led to good partial solutions in previous levels.

In Bortfeldt and Jungmann (2012) the authors propose the "Strip Packing by Tree Search (SPTRS)", an adaptation of the tree search algorithm developed by Fanslau and Bortfeldt (2010) for the three-dimensional container loading problem, also based on the idea of sectioning the strip in layers.

Concluding, although rooted in the eighties, research on level-oriented heuristics is still active, as shows the recent paper Buchwald and Scheithauer (2016) where an improved version of the FFDH heuristic is proposed, the "modified First-Fit Decreasing Height (FFDH*)".

2.3.1.4 Profile-based heuristics

Scheithauer (1997) introduced for the first time the concept of profile (the "contour" in these authors' terminology) to describe a partial solution. The contour is a polygonal line starting on the left side of the strip and ending on the bottom or on the right side of the strip, composed by vertical and horizontal edges. These edges are either edges of rectangles already placed or an extension of those edges. Globally the contour looks like a staircase. The orthogonal polygon defined by this line and the sides and bottom of the strip fully contains all placed rectangles (Figure 2.9).

This representation is used by Scheithauer to build exact methods, which rely on the implicit enumeration of the placement sequences and of the contour's corner points (the feasible placement points). It is revisited by Martello et al. (2003), in the same context of the exact resolution of the SPP, but later on its potential for the development of heuristics for this same problem was recognized by other authors and used to develop new solution approaches.

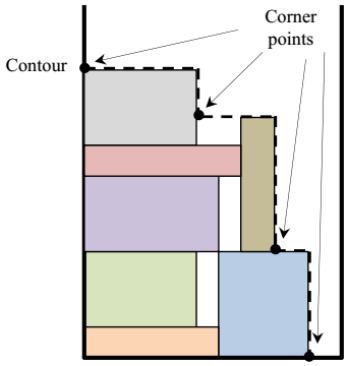


Figure 2.9 – Example of a contour line, used to describe partial solutions in profile-based heuristics.

Pisinger (2007) uses sequence pairs to represent the solutions (the placed rectangles), and the partial solution profile and the corner points to decide where to place the following rectangle. The use of sequence pairs to represent two-dimensional packing solutions was proposed by Murata et al. (1996), and further used by other researchers. As the name suggests sequence pairs are two sequences of rectangles, each of it representing the order by which the rectangles appear to a fictitious observer located at two different points of

the space, namely at the upper-left corner of the layout and at the bottom-left corner of the layout. These sequences are named positive and negative sequences and each of them defines a step-line (layout profiles), named as positive and negative step-lines. In Figure 2.10 the positive and negative sequence that describe a layout are represented, together with the correspondent sets of positive and negative step-lines, the former starting at the bottom-left corner and ending at the upper-right corner, and the latter starting at the upper-left corner and ending at the bottom-right corner. In the figure lines are artificially spaced from each other to improve drawing legibility. Pisinger (2007) propose an innovative and computationally efficient transformation of sequences pairs into layouts. Two different algorithms are described, a "Sequence Pair (SP)" algorithm and a "Semi-normalized Sequence Pair (Seminorm-SP)" algorithm. The first one transforms sequence pairs into normalized layouts, i.e. layouts in which all pieces are bottom-left placed, and the second one applies the same bottom-left property but to the corners of the partial solution profile.

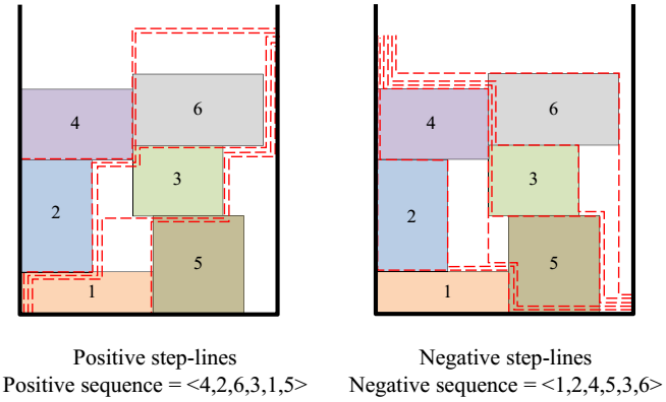


Figure 2.10 – Negative and positive step-lines and their relationship with sequence pairs.

Wei et al. (2011) also uses a profile to describe the partial solution in the "Iterative Doubling Binary Search (IDBS)" heuristic, but in a conceptually different way from the previous authors. Wei's profile does not hold the staircase property of Scheithauer's contour. However, as in the contours, the edges of the profile (called skyline) are either edges or extensions of edges of already placed rectangles (Figure 2.11). The feasible placement points correspond to place the bottom-left corner of the rectangles at the left endpoint of the horizontal lines that belong to the skyline, and to place the bottom right corner of the rectangles at the right endpoint of these same lines. When choosing the placement point for a concrete rectangle, Wei et al. (2011) resort to several criteria based on the waste that is induced by the new rectangle in the position under evaluation and on a scoring method, close to the strategies used in fitness-based heuristics.

Finally, the "Binary Search Heuristic Algorithm (BSHA)" was proposed by Zhang et al. (2013) introducing some improvements in the representation of Scheithauer's contour, by discarding regions that, when considered the shape of the rectangles that remain to be placed, can not be used anymore. The selection of the placement point is based on the waste generated by each tentative placement.

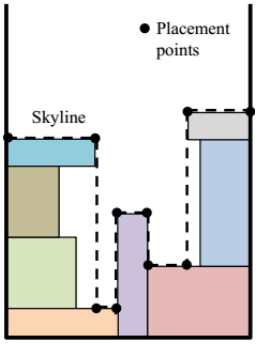


Figure 2.11 – Example of a skyline line, used to describe partial solutions, and its feasible placement points.

2.3.2 Improvement heuristics

Improvement heuristics start with an initial solution, i.e. a complete solution obtained either by a constructive heuristic or randomly generated. This initial solution is then improved by applying small consecutive changes until a stopping criterion is met. The goal is to keep the computational times fairly low while improving the quality of the solutions, when compared to constructive heuristics (Hopper and Turton, 2001b; Burke et al., 2004).

Depending on how solutions are modified, two different types of improvement heuristics may be defined. When modifications are directly applied to the layouts, by moving rectangles around, the search is performed over the actual layout. These heuristics are usually slower because the geometric feasibility has to be enforced. However in these heuristics there is a very direct link between the modification applied to the solution and its effects on the layout. Another option is to apply the modifications on the order (or sequence) by which the rectangles are inserted on the strip, relying on an additional constructive heuristic to transform this input order in an actual layout. These heuristics search over sequences. The generation of modified solutions is faster but there is a weaker link between the modification applied and its effects on the layout.

2.3.2.1 Search over sequences

Heuristics for the SPP that search over sequences resort to modification operators that are common to other problems that rely on sequences to codify their solutions (e.g. traveling salesperson problem, scheduling problems, vehicle routing problems, etc.): the insert and the exchange or swap operators. The insert operator takes a rectangle out of the sequence and re-inserts it in a different position in the sequence. The exchange operator takes two rectangles and exchanges their position in the sequence. Notice that, as in the SPP all rectangles have to be placed on the strip, the also common delete operator cannot be applied, and as only one strip is being considered the exchange between partial sequences (e.g. inter-route swap in vehicle routing problems) does not also make sense.

Figure 2.12 illustrates the overall structure of an improvement heuristic that searches over sequences. An initial input sequence 1 is decoded by a constructive heuristic to pro-

duce an initial layout that has a height h_1, a swap operator is applied to rectangles 1 and 3, resulting in sequence 2. This sequence is decoded into a layout with height h_2, which is better than h_1. Then an insert operator is applied to rectangle 2 and sequence 3 is generated. The correspondent layout has a height h_3, which is equal to h_2. The several heuristics of this type will mainly defer on:

- the type of modification operators used, i.e on how an incumbent sequence is generated from a previous one (many times designated as neighbourhood structure);
- the decoding (constructive) heuristic, that transforms a sequence in a layout;
- when the incumbent sequence is accepted to be the base for further modifications, i.e. the search strategy.

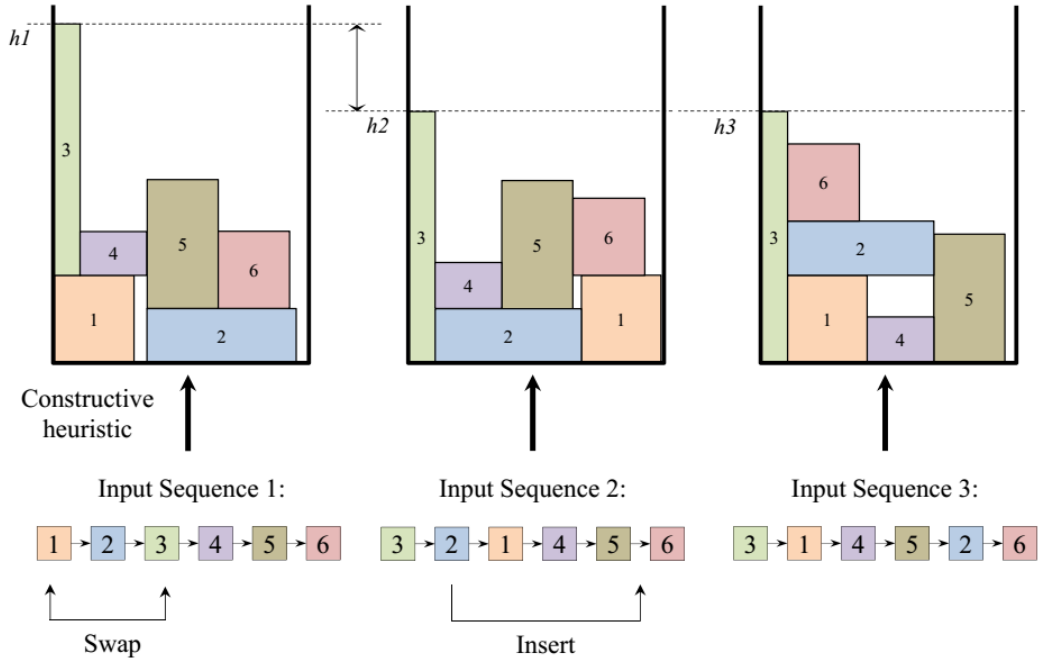


Figure 2.12 – Overall structure of an improvement heuristic that searches over sequences.

The most basic search strategy is Local Search (LS). An incumbent solution is accepted if is better than the previous solution. If not, a new modification is applied to the sequence. The search stops after a previously defined number of sequences have been generated (iterations of the search procedure) without an improvement of the solution. This strategy is used in Hopper and Turton (2001a) but slightly different versions can be found in the literature. In Leung and Zhang (2011) the Fast Layer-Based Heuristic is transformed in a local search heuristic, using the swap operator between two random rectangles to generate different sequences.

Pure local search methods, as the ones just described, are known for getting stuck in local minima, i.e. solutions that are better than any other solution that can be generated

from it, with the chosen modification operator, but with no guarantee of global optimality. To overcome this drawback it is common in combinatorial optimization problems to resort to metaheuristics instead of pure local search. What differentiates metaheuristics is the controlled acceptance of worse incumbent solutions, hoping that the temporary degradation of the solution may lead the search to better regions of the search space, and eventually to the global optimal solution.

One of the seminal works using metaheuristics in the SPP resolution is Jakobs (1996), with genetic algorithms being used as a search strategy and each sequence being transformed into a layout by the bottom-left heuristic. The initial population is composed by random sequences and the crossover operator uses q consecutive rectangles of one parent, starting at position p, and completes the sequence with the missing rectangles by the order given by the other parent. Three mutation operators are used: the sequence of a random block is inverted; some elements of the sequence are exchanged; rectangles are rotated. A similar approach was developed by Hopper and Turton (1999). More recent applications of genetic algorithms to search over sequences can be found in Hadjiconstantinou and Iori (2007); Salto et al. (2008); Burke et al. (2010); Thomas and Chaudhari (2014), resorting to different constructive heuristics to decode the sequences into layouts.

In a hybrid constructive/improvement heuristic, Burke et al. (2009) used simulated annealing to improve a partial initial solution generated by the best-fit heuristic. A first set of rectangles is placed by the best-fit heuristic and these rectangles are not moved during the search procedure. The remaining rectangles are placed by the bottom-left-fill heuristic, according to several sequences that are generated during the search, trying to use the holes between the initially placed rectangles. Resorting to the semi-normalized profile-based constructive heuristic, Pisinger (2007) also developed a simulated annealing approach. Leung et al. (2011) implemented also a simulated annealing algorithm over a constructive heuristic, obtaining improved results over the pure local search algorithm.

Adapting algorithms developed for three-dimensional packing problems to the 2D-SPP is a natural path explored by several authors (e.g. Bortfeldt (2006)). A different approach was followed by Belov et al. (2008) that have adapted the uni-dimensional "Sequential Value Correction" (Belov and Scheithauer, 2007) to the 2D-SPP. Behind this proposal is the observation that solving a two-dimensional rectangular problem is equivalent to solve a set of one-dimensional bin-packing problem, in which the size of the bins is the width of the strip and the items to pack are horizontal slices of the rectangles. In other words, the rectangles to place are horizontally sliced according to a given grid and minimizing the height of the strip becomes minimizing the number of bins used to pack all slices. Of course that contiguity constraints (slices of the same rectangle have to be assigned to contiguous bins) and location constraints (slices of the same rectangle have to be paced at the same x-coordinate) have to be additionally imposed. As the problem is heuristically solved as a succession of knapsack problems (it always selects the left-most free space in the last slice and fills it by a one-dimensional greedy heuristic, considering only item widths) it is possible to impose these additional constraints. In the one-dimensional knapsack problems pseudo-profits are assigned to the slices. These pseudo-profits are related to the area of the rectangle from were each slice comes, and they aim to give priority to the largest rectangles, harder to efficiently place. The overall approach is based on a local search strategy where

different sequences are generated by an insert operator.

Wei et al. (2009) use a mix of profile-based heuristic and fitness-based heuristic, as each rectangle is placed on one of the corner points of the profile but the latter is chosen based on how well the rectangle fits the space. To guide the search a local search strategy is used. Later on Wei et al. (2011) developed a taboo search algorithm over a profile-based constructive heuristic. In Zhang et al. (2013) the profile-based heuristic is also randomized in a pure local search algorithm. Chen et al. (2012) resort to the algorithm by Wei et al. (2009) but apply it by regions, in which the strip is previously divided.

An original and different approach to the concept of searching over sequences was proposed by Burke et al. (2011): the "Squeaky Wheel Heuristic". This improvement heuristic, as all the others, changes the current sequence to generate the incumbent sequence, but incorporates what could be called a learning factor. It uses the best-fit heuristic as a decoder of the rectangle sequence, but the sequence is generated taking into account penalties assigned to each rectangle. A target for the height is defined (in this work it is the so called continuous lower bound – the sum of the areas of the rectangles divided by the width of the strip), and when a rectangle protrudes the target on the height a penalty is assigned to it. The more penalized rectangles are the first ones to place in the next iteration, i.e. the rectangles are ordered by non-increasing value of penalty. This methodology is illustrated in Figure 2.13.

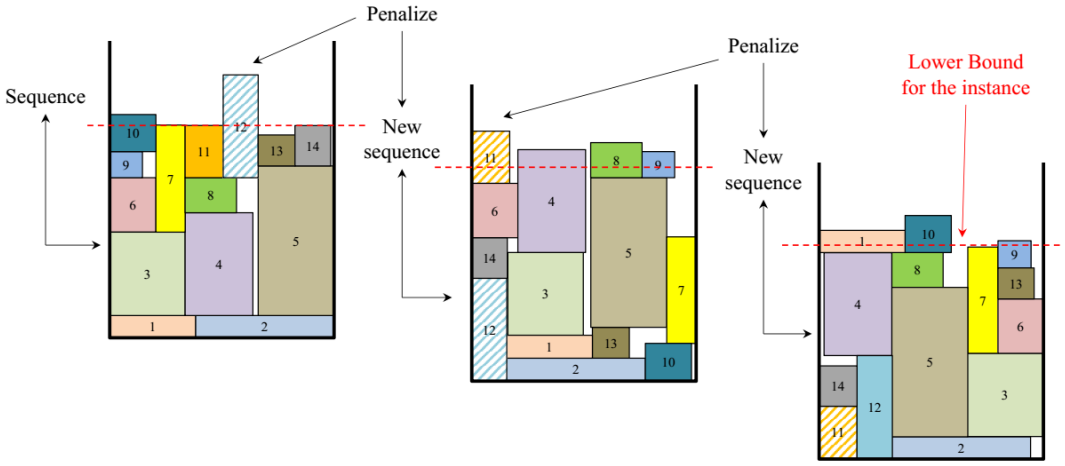


Figure 2.13 – An example of the application of the squeaky wheel heuristic.

Worthwhile of reference are also the works by Borgulya (2014) and Zhang et al. (2016a). If from the point of view of the constructive decoding heuristics they build on previous work, in what concerns the search process these are the first applications of hyper-heuristics and variable neighbourhood search.

2.3.2.2 Search over the layout

When searching over layouts the modifications are operated directly over the layout. An example is presented in Figure 2.14, where the placement of rectangle 1 is exchanged with the placement of rectangles 2, 3 and 4, originating a better solution ($h2 < h1$).

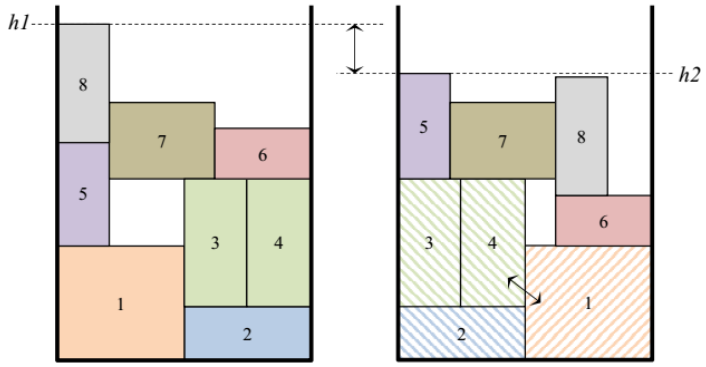


Figure 2.14 – Search over the layout.

One of the first improvement heuristics searching over the layout is proposed by Alvarez-Valdes et al. (2005). Although developed for the two-dimensional cutting problem (a "Placement Problem", according to Wäscher et al. (2007) typology, in which a sub-set of rectangles is to be chosen to be cut from a limited sized stock rectangle), the ideas presented in this paper were at the basis of a later approach to the SPP resolution. The algorithm is based on the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic and therefore has two phases: a constructive phase and an improvement phase. The constructive heuristic is a fitness-based heuristic that gives priority to the use of the smallest empty spaces. A fitness function, which assigns to each piece a fitness value based on how well the piece occupies the space, is defined and, according to the GRASP paradigm, one of the pieces in the restricted candidate list is chosen to be the following one to be placed. The improvement phase is run over the actual layouts, and two different operators are defined:

1. To transform several small unused spaces into larger empty spaces, one or more pieces adjacent to an empty space are removed, all pieces are pushed to the corners of the stock rectangle and the remaining pieces are tentatively placed by the constructive heuristic. This is a pretty good example why the SPP requires specific resolution algorithms: in the SPP all pieces are placed on the strip, there are never remaining pieces except the ones we may explicitly remove from the layout; and there are no corners, or at least there are only two corners, which hinders the idea of separating the already placed rectangles to create room for the incoming ones.

2. To improve the arrangement of the placed rectangles, the last $k\%$ rectangles of the solution are removed (they were placed using the randomized version of the constructive algorithm) and the empty space filled again with (some of) the remaining rectangles by the deterministic version of the constructive heuristic.

In Alvarez-Valdés et al. (2008) this GRASP algorithm is evolved to deal with the SPP. For the improvement phase, four different operators are proposed:

1. From a solution with a height H the last $k\%$ rectangles of the solution are removed, an artificial height of $H\text{-}1$ is imposed to the strip and the solution completed with the

deterministic constructive heuristic. If all rectangles are placed in this $W \times (H-1)$ stock rectangle, then the SPP solution has improved.

2. All rectangles that define the value H of the current solution (i.e. the rectangles whose top edge is at height H – the "guilty" rectangles) are removed from the solution and placed on some of the empty spaces, lower on the strip. If the rectangle exceeds the dimensions of the empty space, the overlapping rectangles are deleted and placed again using the deterministic constructive heuristic (Figure 2.15).

3. The last k% rectangles that have been placed in the constructive phase are removed and placed again with the deterministic constructive heuristic (Figure 2.16). There is no guarantee that the chosen percentage will lead to a layout, with the remaining rectangles, that has a height strictly lower than the original one. If this is the case, additional rectangles are removed until this condition is satisfied.

4. Similar to the previous operator, but in this case all pieces with their top edge exceeding a height λH are removed, with $0 < \lambda < 1$ (Figure 2.17).

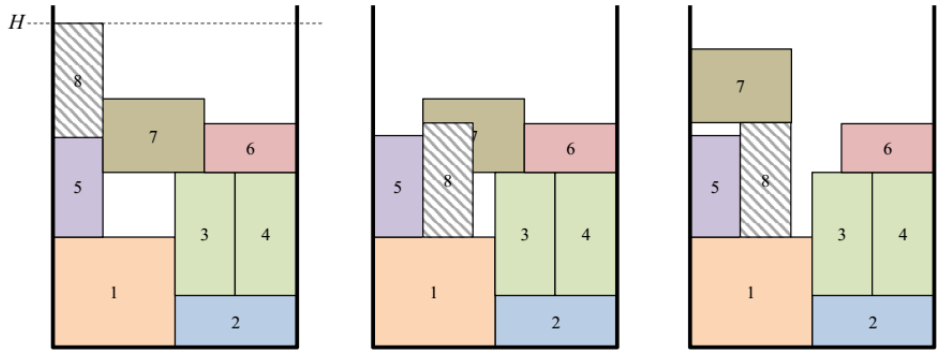


Figure 2.15 – Example of removal and reinsertion of a "Guilty" rectangle.

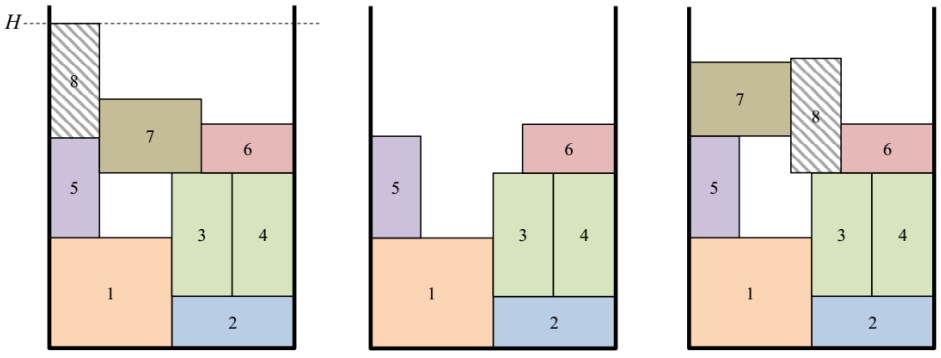


Figure 2.16 – Example of removal and reinsertion of the last 25% rectangles.

Bortfeldt (2006) adapts a genetic algorithm developed for the three-dimensional container loading problem (Bortfeldt and Gehring, 2001) to the 2D-SPP. The algorithm builds

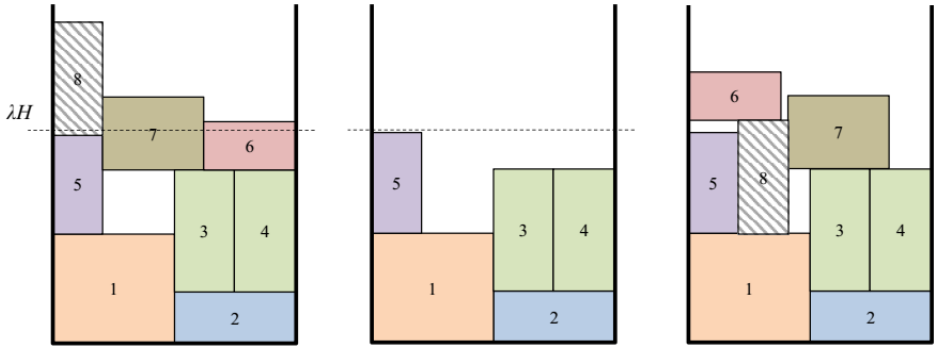


Figure 2.17 – Example of the removal and reinsertion of all rectangles above λH.

on the concept of layer, rather similar to the levels of the level-oriented heuristics, but within each layer rectangles may be placed on the top of other rectangles, as in the more sophisticated level-oriented heuristics. Therefore, a solution is a set of layers and the operators of the genetic algorithm exchange layers between solutions (crossover operator) or simply transfer a certain number of layers (mutation). A repair heuristic is proposed to enforce solutions' feasibility and improve their quality. As partial solutions (the layers) are moved around among different solutions, this approach is rather innovative once usually genetic algorithms for the SPP resort to searching over sequences. Also Zhang et al. (2007) developed a genetic algorithm based on layers, which are generated by the recursive algorithm previously proposed by the same authors (Zhang et al., 2006), the "Improved Heuristic Recursive algorithm (IHR)". Wei et al. (2014) also used layers during the search. The layers are initially created in a level-oriented approach, resorting to the best-fit heuristic to fill each level. However, during the search blocks of rectangles are formed and then placed in each layer instead of the individual rectangles, i.e. the layers' height is dynamically adjusted aiming a tighter placement of the rectangles and, indirectly, a lower layout's height.

Neveu et al. (2008) use the same modification operator as Alvarez-Valdés et al. (2008), picking the rectangle that defines the height of the layout, placing it on an empty space and removing eventually overlapping rectangles, and finally placing them again with a constructive heuristic. The maximal-space representation of the empty regions of the strip is used (here under the name of maximal holes) and a local search strategy, similar to the one proposed by Wei et al. (2009), guides the search. Several constructive heuristics are used to generate different initial solutions.

2.4. Discussion

This work aims to present a structured view over heuristics for the rectangular 2D-SPP. To achieve this, a systematic literature survey was run and 36 papers published in the last decade (from 2007 onward), in international journals with peer review, were reviewed and categorized. Also some previous papers that can be considered seminal in the use of a relevant method, concept or idea, were discussed. This led to the analysis of a total of 48

papers.

In Table 2.1 a list of the references discussed in the previous sections, classified by type of heuristic, is presented. Quite often, in a single paper more than one method or algorithm is proposed and in these papers it is frequent to find both constructive and improvement heuristics, mainly when using metaheuristics. Therefore, their classification is based on the authors' opinion about where the main contribution was made, but it is naturally arguable. Research gaps and future research directions will now be discussed.

Constructive	Improvement			
	Search over sequences	Search strategy[a]	Search over layouts	Search strategy[a]
Baker et al. (1980)	Jakobs (1996)	GA	Alvarez-Valdes et al. (2005)	GRASP
Coffman Jr. et al. (1980)	Hopper and Turton (1999)	GA	Bortfeldt (2006)	GA
Chazelle (1983)	Liu and Teng (1999)	GA	Zhang et al. (2007)	GA
Berkey and Wang (1987)	Hopper and Turton (2001a)	GA; SA; NE; LS	Alvarez-Valdés et al. (2008)	GRASP
Burke et al. (2004)	Hadjiconstantinou and Iori (2007)	GA	Neveu et al. (2008)	LS
Zhang et al. (2006)	Pisinger (2007)	SA		
Ntene (2007)	Belov et al. (2008)	SVC		
Cui et al. (2008)	Salto et al. (2008)	GA		
Aşık and Özcan (2009)	Burke et al. (2009)	SA		
Imahori and Yagiura (2010)	Wei et al. (2009)	LS		
Ortmann et al. (2010)	Burke et al. (2010)	HH; GA		
Kotov and Cao (2011)	Burke et al. (2011)	LS		
Bortfeldt and Jungmann (2012)	Leung and Zhang (2011)	SA; LS		
Cui et al. (2013)	Leung et al. (2011)	LS		
Özcan et al. (2013)	Wei et al. (2011)	TS		
Verstichel et al. (2013)	Chen et al. (2012)	LS		
Buchwald and Scheithauer (2016)	Yang et al. (2013)	SA		
Wei et al. (2014)	Wauters et al. (2013)	ShP		
Zhang et al. (2016b)	Zhang et al. (2013)	LS		
	Borgulya (2014)	HH; GA; LS		
	Thomas and Chaudhari (2014)	GA		
	Wei et al. (2016)	LS		
	Chen et al. (2015)	LS		
	Zhang et al. (2016a)	VNS; LS		

[a] GA: Genetic Algorithm; GRASP: Greedy Randomized Adaptive Search Procedure; HH: Hyper-Heuristic; LS: Local Search; NE: Naive evolution; SA: Simulated Annealing; ShP: Shaking procedure; SVC: Sequential Value Correction; TS: Tabu Search; VNS: Variable Neighbourhood Search.

Table 2.1 – Heuristics for the SPP.

The chart from Figure 2.18 shows that the last decade has been rich in terms of contributions for the SPP problem. All types of strategies have been used, from the more simple constructive heuristics to the more complex improvement heuristics with search over the layout. However, most probably due to the complexity of dealing with the geometric feasibility of the layouts, the latter strategy has been less frequently used. It should be kept in mind that, in improvement heuristics that search over sequences, all modifications to a solution originate a feasible layout, as it is the decoder responsibility to produce the actual layouts, given the sequences of rectangles. Additionally, these heuristics can use as

decoders the many and efficient constructive heuristics available. This may be the main reason for the ratio of 1 to 4 that we find when comparing the number of improvement heuristics that search over sequences versus improvement heuristics that search over layouts.

From the search control algorithm point of view, pure local search, simulated annealing and genetic algorithms are the most frequent algorithms, with the most recent metaheuristics being absent. With just one approach based on variable neighbourhood search, and another one based on hyper heuristics, there is plenty of room for research in the use of more sophisticated search control mechanisms, in particular in improvement heuristics with search over layouts.

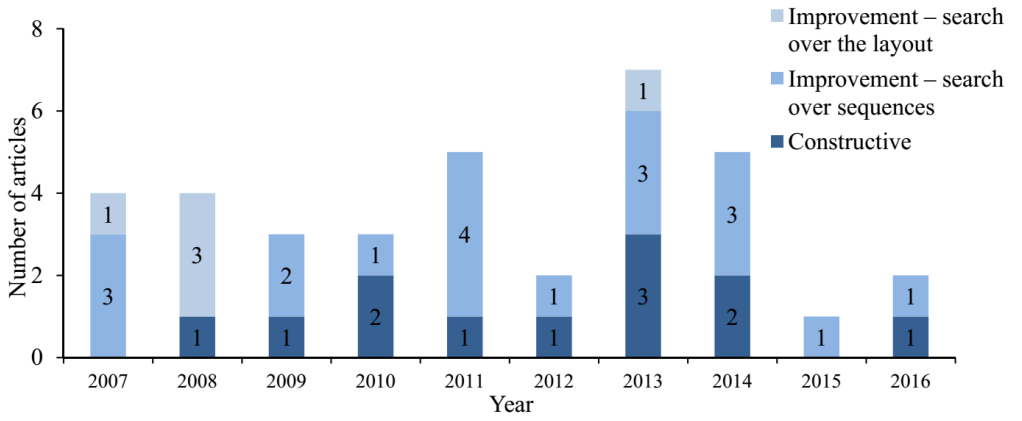


Figure 2.18 – Number of publications distributed by type of improvement heuristic.

Another relevant factor for the heuristics' design choices is the size of the instances. Since 2007, very large instances became the main challenge for new heuristics, and by very large instances it is meant instances with more than 5,000 rectangles. These instances require very fast solution generation and evaluation and special care has to be put on the computational implementation of the methods, with important and relevant contributions from the Computer Science field. Some of the methods proposed in the past are specialized in solving zero-waste instances, but these are not only rather artificial instances, that do not arise in practice, but these instances can also significantly simplify the SPP resolution, when this characteristic is explored by the resolution method.

When looking in more detail at the constructive heuristics (Figure 2.19), it can be seen that more than 40% of them are fitness-based heuristics. It is surely due to their good performance, when compared to positioning-based heuristics. In fact, fitness-based heuristics have been a trend in the research in this field. In particular scoring heuristics, which are able to measure the quality of the placement of a rectangle in a space, as initially proposed by Leung and Zhang (2011), seems to deliver good results and several other scoring heuristics were proposed (IA, ISA, SRA, and HDA).

Level-based heuristics are basically extensions of the FFDH constructive heuristic. They received a lot of attention in the eighties but are currently less studied. Probably this is because only very specific applications require this type of layouts, not being the extra waste that levels impose worthwhile, when levels are not a concrete constraint of the

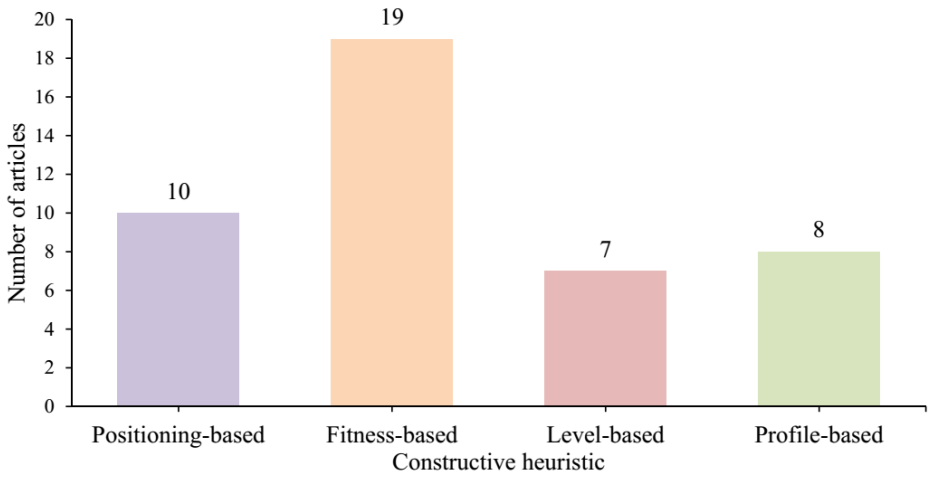


Figure 2.19 – Number of papers by type of constructive heuristics.

application.

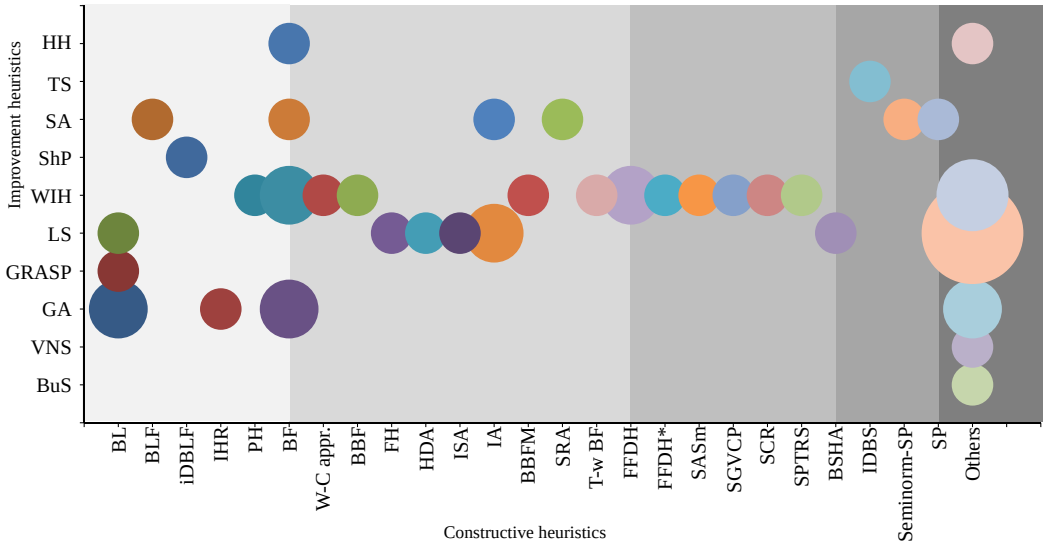
Profile-based heuristics have been less explored and represent a clear research opportunity in this field. It is well-known since the eighties (Baker et al., 1980) that there are optimal solutions for the SPP that can not be achieved by the bottom-left heuristic, whatever rectangle sequence is considered. However, this is not true when profile-based approaches are used. Martello et al. (2003) proposed an exact tree-search algorithm in which, in each node of the branch-decision tree, child nodes are generated combining all rectangles remaining to place with all corner points of the profile describing the partial solution. Therefore, for each optimal layout there is a sequence of rectangles that, properly choosing the placement points on the profile, generates it. Profile-based solution representations are, therefore, rather promising for the development of effective heuristics for the SPP, as they seem to gather the advantages of positioning-based and fitness-based methods, while keeping under control time complexity.

The chart of Figure 2.20 relates the improvement heuristics based on search over sequences with the constructive heuristics used to decode the sequences (the list of acronyms used in the x-axis is presented in Table 2.2). The acronym *WIH*, in the y-axis, stand for "without improvement heuristic" and refers to the algorithms that do not have an improvement phase, i.e. are just constructive. The first four shades of gray stands for the four types of constructive heuristics. From the lightest gray to the darkest gray: positioning-based, fitness-based, level-based and profile-based. This chart confirms the findings already mentioned. Clearly, combining genetic algorithms with the bottom-left heuristic, and its variants, is the most frequent approach and the most recent metaheuristics have not yet been explored.

A final word about benchmark instances. There is a well-established set of instances that is consistently used by the research community. However, many of these instances have been proposed decades ago and it is reasonable to question if they are still challenging for now-a-days algorithms, given the hardware where they are currently run. Moreover, some of these instances have characteristics that are not typical from real-world applications and

Acronym	Constructive Heuristic	Authors
BF	Best-Fit	Burke et al. (2004)
W-C appr.	Best-Fit Worst-Case Approximation ratio	Imahori and Yagiura (2010)
BBF	Bidirectional Best-Fit	Aşık and Özcan (2009)
BSHA	Binary Search Heuristic Algorithm	Zhang et al. (2013)
BL	Bottom-Left	Baker et al. (1980)
BLF	Bottom-Left-Fill	Chazelle (1983)
IA	Efficient Improved Algorithm (IA)	Wei et al. (2016)
FH	Fast layer-based Heuristic	Leung and Zhang (2011)
FFDH	First-Fit Decreasing Height	Coffman Jr. et al. (1980)
HDA	Hybrid Demon Algorithm	Chen et al. (2015)
iDBLF	Improved Deepest Bottom Left Fill	Wauters et al. (2013)
IHR	Improved Heuristic Recursive algorithm	Zhang et al. (2007)
ISA	Intelligent Search Algorithm	Leung et al. (2011)
IDBS	Iterative Doubling Binary Search	Wei et al. (2011)
BBFM	Modified Bidirectional Best-Fit	Özcan et al. (2013)
FFDH*	Modified First-Fit Decreasing Height	Buchwald and Scheithauer (2016)
SASm	Modified Size-Alternating Stack algorithm	Ortmann et al. (2010)
PH	Priority Heuristic	Zhang et al. (2016b)
Seminorm-SP	Seminormalized-Sequence Pair	Pisinger (2007)
SP	Sequence Pair	Pisinger (2007)
SGVCP	Sequential Grouping and Value Correction Proc.	Cui et al. (2013)
SRA	Simple Randomized Algorithm	Yang et al. (2013)
SCR	Stack Ceiling with Re-sorting algorithm	Ortmann et al. (2010)
T-w BF	Three-way Best-Fit	Verstichel et al. (2013)
SPTRS	Strip Packing by Tree Search	Bortfeldt and Jungmann (2012)

Table 2.2 – Constructive heuristics for the SPP: list of acronyms and descriptions.



Improvement heuristics: BuS: Bubble Search; GA: Genetic Algorithm; GRASP: Greedy Randomized Adaptive Search Procedure; HH: Hyper Heuristic; LS – Local Search; SA: Simulated Annealing; ShP: Shaking Procedure; TS: Tabu Search; VNS: Variable Neighbourhood Search; WIH: Without Improvement Heuristic.

Figure 2.20 – Relation between the improvement heuristics based on search over sequences and the constructive heuristics used to decode the sequences.

allow that less general approaches outperform more solid and robust algorithms, by taking advantage of the special characteristics of these instances. Computational experiments are not league tables. Computational experiments are meant to understand the proposed algorithm, in which situations it goes well, in which it does not so well, which are its strengths and its limitations.

The limitations of the benchmark instances can only be overcome by the use of problem generators that generate instances to systematically explore the space of input data configurations: rectangles area, aspect ratio, heterogeneity, number, etc. Silva et al. (2014) proposed a problem generator for two-dimensional (and three-dimensional) rectangular cutting and packing problems, in all the variants, that could also be used for strip packing problems, to overcome the limitations of the existing benchmark instances.

2.5. Conclusion

In this review, the last decade of publications related to heuristics for rectangular 2D-SPP has been surveyed. Each contribution was classified according to a well-known conceptual framework (constructive heuristics and improvement heuristics, both with search over sequences and search over layouts), with an additional division of constructive heuristics in positioning-based, fitness-based, level-based and profile-based heuristics.

From the literature review emerges the big dynamism of the field. Despite having its roots in the very early eighties, strip packing problems are still intensively researched and more complex approaches are being proposed by researchers from all over the world. Research gaps are identified, mainly associated to the use of more sophisticated search algorithms, specially metaheuristics, to the use of profile representations of solutions and to the development of improvement heuristics with search over the layout.

A necessary step in the field is related to the computational validation of the new algorithms, that have to make progresses in the direction of having as goal the understanding of why and when the methods work better and worse.

References

Alvarez-Valdes, R., Parreño, F., and Tamarit, J. M. (2005). A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems. *Journal of the Operational Research Society*, 56(4):414–425.

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2008). Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, 35(4):1065–1083.

Aşık, Ö. B. and Özcan, E. (2009). Bidirectional best-fit heuristic for orthogonal rectangular strip packing. *Annals of Operations Research*, 172(1):405–427.

Baker, B. S., Coffman, Jr, E. G., and Rivest, R. L. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855.

Belov, G. and Scheithauer, G. (2007). Setup and open-stacks minimization in one-dimensional stock cutting. *INFORMS Journal on Computing*, 19(1):27–35.

Belov, G., Scheithauer, G., and Mukhacheva, E. a. (2008). One-dimensional heuristics adapted for two-dimensional rectangular strip packing. *Journal of the Operational Research Society*, 59(6):823–832.

Berkey, J. O. and Wang, P. Y. (1987). Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38(5):423–429.

Borgulya, I. (2014). A parallel hyper-heuristic approach for the two-dimensional rectangular strip-packing problem. *Journal of Computing and Information Technology*, 22(4):251–265.

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3):814–837.

Bortfeldt, A. and Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161.

Bortfeldt, A. and Jungmann, S. (2012). A tree search algorithm for solving the multi-dimensional strip packing problem with guillotine cutting constraint. *Annals of Operations Research*, 196(1):53–71.

Buchwald, T. and Scheithauer, G. (2016). Upper bounds for heuristic approaches to the strip packing problem. *International Transactions in Operational Research*, 23(1-2):93–119.

Burke, E. K., Hyde, M., Kendall, G., and Woodward, J. (2010). A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. *IEEE Transactions on Evolutionary Computation*, 14(6):942–958.

Burke, E. K., Hyde, M. R., and Kendall, G. (2011). A squeaky wheel optimisation methodology for two-dimensional strip packing. *Computers and Operations Research*, 38(7):1035–1044.

Burke, E. K., Kendall, G., and Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671.

Burke, E. K., Kendall, G., and Whitwell, G. (2009). A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem. *INFORMS Journal on Computing*, 21(3):505–516.

Chazelle, B. (1983). The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, 100(8):697–707.

Chen, B., Wang, Y., and Yang, S. (2015). A hybrid demon algorithm for the two-dimensional orthogonal strip packing problem. *Mathematical Problems in Engineering*, 2015(1):1–14.

Chen, J., Zhu, W., and Peng, Z. (2012). A heuristic algorithm for the strip packing problem. *Journal of Heuristics*, 18(4):677–697.

Coffman Jr., E. G., Garey, M. R., Johnson, D. S., and Tarjan, R. E. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826.

Cui, Y., Yang, L., and Chen, Q. (2013). Heuristic for the rectangular strip packing problem with rotation of items. *Computers and Operations Research*, 40(4):1094–1099.

Cui, Y., Yang, Y., Cheng, X., and Song, P. (2008). A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem. *Computers and Operations Research*, 35(4):1281–1291.

Dowsland, K. A., Dowsland, W. B., and Bennell, J. A. (1998). Jostling for position: Local improvement for irregular cutting patterns. *Journal of the Operational Research Society*, 49(6):647–658.

Fanslau, T. and Bortfeldt, A. (2010). A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing*, 22(2):222–235.

Hadjiconstantinou, E. and Iori, M. (2007). A hybrid genetic algorithm for the two-dimensional single large object placement problem. *European Journal of Operational Research*, 183(3):1150–1166.

Hopper, E. and Turton, B. (1999). A genetic algorithm for a 2d industrial packing problem. *Computers and Industrial Engineering*, 37(1985):375–378.

Hopper, E. and Turton, B. C. H. (2001a). An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57.

Hopper, E. and Turton, B. C. H. (2001b). A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4):257–300.

Hurink, J. L. and Paulus, J. J. (2011). Improved online algorithms for parallel job scheduling and strip packing. *Theoretical Computer Science*, 412(7):583–593.

Imahori, S. and Yagiura, M. (2010). The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. *Computers and Operations Research*, 37(2):325–333.

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1):165–181.

Kotov, V. M. and Cao, D. (2011). A heuristic algorithm for the non-oriented 2D rectangular strip packing problem. *Buletinul Academiei de Ştiinţe a Republicii Moldova. Matematica*, (2):81–88.

Leung, S. C. H. and Zhang, D. (2011). A fast layer-based heuristic for non-guillotine strip packing. *Expert Systems with Applications*, 38(10):13032–13042.

Leung, S. C. H., Zhang, D., and Sim, K. M. (2011). A two-stage intelligent search algorithm for the two-dimensional strip packing problem. *European Journal of Operational Research*, 215(1):57–69.

Liu, D. and Teng, H. (1999). An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. *European Journal of Operational Research*, 112(2):413–420.

Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252.

Lodi, A., Martello, S., and Vigo, D. (1999). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357.

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

Mumford-Valenzuela, C. L., Vick, J., and Wang, P. Y. (2004). Heuristics for large strip packing problems with guillotine patterns: An empirical study. *Metaheuristics: Computer Decision-Making*, 2004(1):501–522.

Murata, H., Fujiyoshi, K., Nakatake, S., and Kajitani, Y. (1996). VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1518–1524.

Neveu, B., Trombettoni, G., Araya, I., and Riff, M. C. (2008). A strip packing solving method using an incremental move based on maximal holes. *International Journal on Artificial Intelligence Tools*, 17(5):881–901.

Ntene, N. (2007). *An algorithmic approach to the 2D oriented strip packing problem*. PhD thesis, Department of Logistics, University of Stellenbosch.

Ntene, N. and van Vuuren, J. H. (2009). A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem. *Discrete Optimization*, 6(2):174–188.

Ortmann, F. G., Ntene, N., and van Vuuren, J. H. (2010). New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. *European Journal of Operational Research*, 203(2):306–315.

Özcan, E., Kai, Z., and Drake, J. H. (2013). Bidirectional best-fit heuristic considering compound placement for two dimensional orthogonal rectangular strip packing. *Expert Systems with Applications*, 40(10):4035–4043.

Pisinger, D. (2007). Denser packings obtained in O(n log log n) time. *INFORMS Journal on Computing*, 19(3):395–405.

Riff, M. C., Bonnaire, X., and Neveu, B. (2009). A revision of recent approaches for two-dimensional strip-packing problems. *Engineering Applications of Artificial Intelligence*, 22(4-5):833–837.

Salto, C., Alba, E., Molina, J. M., and Leguizamón, G. (2008). Greedy seeding procedure for GAs solving a strip packing problem. *Inteligencia Artificial*, 12(40):73–85.

Scheithauer, G. (1997). Equivalence and dominance for problems of optimal packing of rectangles. *Ricerca Operativa*, 83(1):3–34.

Silva, E., Oliveira, J. F., and Wäscher, G. (2014). 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, 237(3):846–856.

Thomas, J. and Chaudhari, N. S. (2014). A new metaheuristic genetic-based placement algorithm for 2D strip packing. *Journal of Industrial Engineering International*, 10(1):1–16.

Verstichel, J., De Causmaecker, P., and Berghe, G. V. (2013). An improved best-fit heuristic for the orthogonal strip packing problem. *International Transactions in Operational Research*, 20(5):711–730.

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

Wauters, T., Verstichel, J., and Vanden Berghe, G. (2013). An effective shaking procedure for 2D and 3D strip packing problems. *Computers and Operations Research*, 40(11):2662–2669.

Wei, L., Oon, W. C., Zhu, W., and Lim, A. (2011). A skyline heuristic for the 2D rectangular packing and strip packing problems. *European Journal of Operational Research*, 215(2):337–346.

Wei, L., Qin, H., Cheang, B., and Xu, X. (2016). An efficient intelligent search algorithm for the two-dimensional rectangular strip packing problem. *International Transactions in Operational Research*, 23(1-2):65–92.

Wei, L., Tian, T., Zhu, W., and Lim, A. (2014). A block-based layer building approach for the 2d guillotine strip packing problem. *European Journal of Operational Research*, 239(1):58–69.

Wei, L., Zhang, D., and Chen, Q. (2009). A least wasted first heuristic algorithm for the rectangular packing problem. *Computers and Operations Research*, 36(5):1608–1614.

Yang, S., Han, S., and Ye, W. (2013). A simple randomized algorithm for two-dimensional strip packing. *Computers and Operations Research*, 40(1):1–8.

Ye, D., Han, X., and Zhang, G. (2009). On-line multiple-strip packing. *Combinatorial Optimization and Applications*, 2009:155–165.

Zhang, D., Che, Y., Ye, F., Si, Y.-W., and Leung, S. C. (2016a). A hybrid algorithm based on variable neighbourhood for the strip packing problem. *Journal of Combinatorial Optimization*, 32(2):513–530.

Zhang, D., Kang, Y., and Deng, A. (2006). A new heuristic recursive algorithm for the strip rectangular packing problem. *Computers and Operations Research*, 33(8):2209–2217.

Zhang, D., Shi, L., Leung, S. C., and Wu, T. (2016b). A priority heuristic for the guillotine rectangular packing problem. *Information Processing Letters*, 116(1):15–21.

Zhang, D., Wei, L., Chen, Q., and Leung, S. C. H. (2013). A binary search heuristic algorithm based on randomized local search for the rectangular strip packing problem. *INFORMS Journal on Computing*, 25(2):332–345.

Zhang, D.-F., Chen, S.-D., and Liu, Y.-J. (2007). An improved heuristic recursive strategy based on genetic algorithm for the strip rectangular packing problem. *Acta Automatica Sinica*, 33(9):911–916.

CHAPTER 3

The 2D-SPP: What Matters?

Alvaro Neuenfeldt Júnior* · **Elsa Silva*** · **A. Miguel Gomes*** · **José Fernando Oliveira***

Abstract

This paper presents an exploratory approach to study and identify the main characteristics of the Two-Dimensional Strip Packing Problem (2D-SPP). A large number of descriptive variables was defined to represent the main problem characteristics, aggregated in six groups, established through qualitative knowledge about the context of the problem. Coefficient correlation is used as a quantitative measure to validate the assignment of descriptive variables to groups. A Principal Component Analysis (PCA) is used to reduce the dimensions of each group, taking advantage of the relations between descriptive variables from the same group. Our analysis indicates that the problem can be reduced to 19 characteristics, retaining most part of the total variance. These characteristics can be used to fit regression models to estimate the strip height necessary to position all rectangles inside the strip.

Keywords

Strip packing problems; Cutting and packing problems; Principal component analysis; Knowledge discovery.

3.1. Introduction

In the Two-Dimensional Strip Packing Problem (2D-SPP) the aim is to pack a set of rectangles inside a rectangular strip with a fixed width, minimizing the height dimension of the strip that is infinite. The small rectangles can be rotated, orthogonally positioned without overlapping and completely inside the strip. This description fits in the definition of cutting and packing problems and indeed the 2D-SPP can be classified as an open dimension problem (Wäscher et al., 2007). An example can be found in Figure 3.1.

Over the years a considerable number of problem instances appeared in the literature to test the different heuristics that have been developed to solve the 2D-SPP. However, none of the developed heuristics were able to solve efficiently all the existing problem instances and 2D-SPP variants.

The problem instances are generally created with the use of some problem generators which were developed considering specific characteristics, methodologies and input parameters. As a consequence, it is possible to find problem instances in the literature

*INESC TEC, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4600-001 Porto, Portugal.

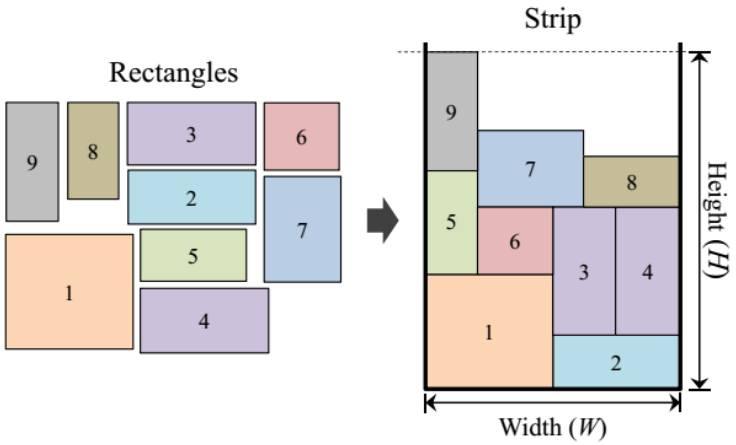


Figure 3.1 – A general view of the rectangular 2D-SPP.

with different characteristics and combinations between rectangles and strip shape variation (Oliveira et al., 2016).

Data mining techniques can be used to facilitate a better understanding of the problem instances characteristics, ensuring that the main details of the problem are known (Smith-Miles and Lopes, 2012).

In this paper, we conduct an exploratory research to find the most relevant problem characteristics for the rectangular 2D-SPP. An initial set of descriptive variables (more than fifty) are used to represent these characteristics, and the Principal Component Analysis (PCA) was chosen as a technique to explore the relations among descriptive variables and to convert them in a smaller number of components. A sample of 1,217 problem instances extracted from the literature is explored.

A similar approach was developed by López-Camacho et al. (2013), where a PCA was considered to develop two-dimensional analysis with the aim of better understand the structure of the two-dimensional (and three-dimensional) bin packing problems. This information was used to compare the performance of heuristics with a wide set of characteristics provided by problem instances found in the literature. López-Camacho et al. (2014) also developed a hyper-heuristic approach and compared the computational results with other approaches using the components developed through the PCA.

The paper is organized as follows. Section 3.2 fully presents the 2D-SPP characteristics. In Section 3.3 the most important problem instances found in the literature are presented. Section 3.4 the exploratory analysis using the PCA is proposed, while in Section 3.5 the most relevant findings are presented, together with some future work ideas and proposals.

3.2. The strip packing problem characteristics

Problem generators are one of the most efficient ways to replicate real-world applications of different problems over controlled scenarios, ensuring the reproducibility of the prob-

lem instances. Problem generators can be used as one source of information to study the characteristics of packing problems. A literature review about problem generators for the 2D-SPP is presented below and will serve as a base for the development of descriptive variables.

Wang and Valenzela (2001) developed two important factors to generate problem instances for the rectangular placement problems: the maximum area ratio between all pairs of rectangles; and the aspect ratio, to identify the variation level between the largest and the smallest dimension of each rectangle. As consequence, the maximum aspect ratio identifies the rectangle that has a greater difference in its dimensions.

Regardless the total number of rectangles, larger values for aspect ratio and area ratio indicates a more significant variability in rectangles shapes, which allows the generation of more heterogeneous process instances. For example, "nice" problem instances have rectangles of smaller size and variability, which indicate a homogeneous behavior. In contrast, "path" problem instances have rectangles with larger size and variability, which characterizes problem instances with higher heterogeneity.

Bortfeldt and Gehring (2001) used the degree of heterogeneity among the rectangles as one of the primordial factors to generate problem instances. The degree of heterogeneity measures the ratio between the total number of different rectangles in comparison to the total number of rectangles of a test problem instance. This characteristic is reaffirmed by Bortfeldt (2006) as one of the most important aspects to be considered in cutting and packing problems.

In Berkey and Wang (1987) the width ratio is considered, that is calculated using the strip width and rectangles dimensions. This measure is one of the most important to develop problem instances for both two-dimensional bin packing problems and 2D-SPP. The influence of width ratio on the quality of solutions for the strip packing was verified mainly in problem instances with a smaller number of rectangles ($n < 100$). Smaller width ratios indicated a greater probability of obtaining lower strip heights.

The 2DCPackGen problem generator proposed by Silva et al. (2014) is able to generate problem instances for all two-dimensional (and three-dimensional) rectangular cutting and packing problems. In specific, for the 2D-SPP, the number of different rectangle types, the rectangle type demand and size and shape of the rectangles are some of the measures that influences the assortment of rectangles.

Leung et al. (2011) combined in a dataset with 16 problem instances some characteristics of *gcut13* by Beasley (1985b) and *cx* by Ferreira and Oliveira (2005) to obtain rectangles with different maximum and minimum areas using the generator proposed by Wang and Valenzela (2001). The main objective was to evaluate the capacity of some heuristics to solve problems with a strongly heterogeneous rectangles shape variation, where the position of the larger rectangles into the strip is a determinant factor to obtain good solutions.

To help the generation of experimental data to cover more practical applications for the knapsack problem, Hall and Posner (2001) used a wide range of factors to identify the problem instances difficulty. Three characteristics can be explored in the 2D-SPP context: the total number of rectangles (problem instances size); the coefficients of all rectangles values; and the heterogeneity (relation between the size and the number of different problem instances).

All the concepts and parameters about the problem generators previously described are used to develop the descriptive variables to study the characteristics of the rectangular 2D-SPP. These descriptive variables were created considering both rectangles and strip shape variation, as well as some intrinsic factors of the problem instances.

Finally, a total of 56 descriptive variables defined, divided into six groups, accordingly to their origin and level of similarity: *Area* (Table 3.1), *Perimeter* (Table 3.2), *Dimensions* (Table 3.3), *Widthdimensions* (Table 3.4), *Proportions* (Table 3.5), and *Other* (Table 3.6).

To simplify the descriptive variables calculation five reference parameters for each rectangle r were defined:

- $area_r = A/a_r$: Ratio between the strip area (A) and rectangle r area (a_r);
- $perimeter_r = P/p_r$: Ratio between the strip perimeter (P) and rectangle r perimeter (p_r);
- $dimension_r = W/[(d1_r + d2_r)/2]$: Average dimension of rectangle r compared to the strip width (W). $d1_r$ is the largest rectangle dimension and $d2_r$ is the smallest rectangle dimension;
- $proportion_r = (D1/D2)/(d1_r/d2_r)$: Level of proportion between the strip and rectangle dimensions. $D1$ is the largest strip dimension and $D2$ is the smallest strip dimension;
- $widthdimension_r = W/d1_r$: Size of the largest rectangle dimensions ($d1_r$) compared to the strip width (W).

The notation of variables and parameters used in this research were standardized, in order to facilitate the understanding of the text. Small letters (e.g. a_r) represent rectangles dimensions, capital letters (e.g. A) are used to strip dimensions. Extended words are reserved for the definition of the reference parameters (e.g. $area_r$), for descriptive variables (e.g. *arearatio*), and for components (e.g *areacomp*).

Variable	Definition
arearatioextr	$arearatioextr = area_{max}/area_{min}$.
arearatioperc	Ratio between percentile 90% and percentile 10% of $area_r$.
arearatioquart	Ratio between third quartile and first quartile of $area_r$.
areacompnormal	Ratio between the sum of 50% larger $area_r$ and the sum of 50% smaller $area_r$.
areacompquart	Ratio between the sum of 25% larger $area_r$ and the sum of 25% smaller $area_r$.
areacompextr	Ratio between the sum of 10% larger $area_r$ and the sum of 10% smaller $area_r$.
areamean	Mean value of $area_r$.
areamed	Median value of $area_r$.
areastdev	Standard deviation of $area_r$.

Table 3.1 – Descriptive variables summary for group *Area*.

Variable	Definition
perimratioextr	$perimratioextr = perimeter_{max}/perimeter_{min}$.
perimratioperc	Ratio between percentile 90% and percentile 10% of $perimeter_r$.
perimratioquart	Ratio between third quartile and first quartile of $perimeter_r$.
perimcompnormal	Ratio between the sum of 50% larger $perimeter_r$ and the sum of 50% smaller $perimeter_r$.
perimcompquart	Ratio between the sum of 25% larger $perimeter_r$ and the sum of 25% smaller $perimeter_r$.
perimcompextr	Ratio between the sum of 10% larger $perimeter_r$ and the 10% smaller $perimeter_r$.
perimmean	Mean value of $perimeter_r$.
perimmed	Median value of $perimeter_r$.
perimstdev	Standard deviation of $perimeter_r$.

Table 3.2 – Descriptive variables summary for group *Perimeter*.

Variable	Definition
vardim	$vardim = W/[\sum(d1_r + d2_r)/n]$.
dimratioextr	$dimratioextr = dimension_{max}/dimension_{min}$.
dimratioperc	Ratio between percentile 90% and percentile 10% of $dimension_r$.
dimratioquart	Ratio between third quartile and first quartile of $dimension_r$.
dimcompnormal	Ratio between the sum of 50% larger $dimension_r$ and the sum of 50% smaller $dimension_r$.
dimcompquart	Ratio between the sum of 25% higher $dimension_r$ and the sum of 25% smaller $dimension_r$.
dimcompextr	Ratio between the sum of 10% higher $dimension_r$ and the sum of 10% smaller $dimension_r$.
dimmean	Mean value of $dimension_r$.
dimmed	Median value of $dimension_r$.
dimstdev	Standard deviation of $dimension_r$.

Table 3.3 – Descriptive variables summary for group *Dimensions*.

Variable	Definition
widthdimratioextr	$widthdimratioextr = widthdimension_{max}/widthdimension_{min}$.
widthdimratioperc	Ratio between percentile 90% and percentile 10% of $widthdimension_r$.
widthdimratioquart	Ratio between third quartile and first quartile of $widthdimension_r$.
widthdimcompnormal	Ratio between the sum of 50% larger $widthdimension_r$ and the sum of 50% smaller $widthdimension_r$.
widthdimcompquart	Ratio between the sum of 25% larger $widthdimension_r$ and the sum of 25% smaller $widthdimension_r$.
widthdimcompextr	Ratio between the sum of 10% larger $widthdimension_r$ and the sum of 10% smaller $widthdimension_r$.
widthdimmean	Mean value of $widthdimension_r$.
widthdimmed	Median value of $widthdimension_r$.
widthdimstdev	Standard deviation of $widthdimension_r$.

Table 3.4 – Descriptive variables summary for group *Widthdimensions*.

Variable	Definition
aspectratio	$aspectratio = (D1/D2)/[\sum(d1_r/d2_r)/n]$.
propratioextr	$propratioextr = proportion_{max}/proportion_{min}$.
propratioperc	Ratio between percentile 90% and percentile 10% of $proportion_r$.
propratioquart	Ratio between third quartile and first quartile of $proportion_r$.
propcompnormal	Ratio between the sum of 50% larger $proportion_r$ measures and the sum of 50% smaller $proportion_r$.
propcompquart	Ratio between the sum of 25% larger $proportion_r$ measures and the sum of 25% smaller $proportion_r$.
propcompextr	Ratio between the sum of 10% larger $proportion_r$ measures and the sum of 10% smaller $proportion_r$.
propmean	Mean value of $proportion_r$.
propmed	Median value of $proportion_r$.
propstdev	Standard deviation of $proportion_r$.

Table 3.5 – Descriptive variables summary for group *Proportions*.

Variable	Definition
n	Total number of rectangles in the test problem instance.
coefficient	$coefficient = [(\Sigma d1_r/n) + (\Sigma d2_r/n)]/2$. Average rectangles dimensions values.
heterogeneity	$heterogeneity = nt/n$. Proportion between the quantity of different rectangles (nt) for all n.
heterognt	Measure of heterogeneity considering only types of rectangles with more than one rectangle.
difcoefficient	For all n, the total number of different rectangles dimensions.
objdimratio	Number of times that the maximum rectangles is bigger than the strip width.
itdimratio	Number of times that the rectangles the maximum rectangles dimension is bigger than the minimum rectangles dimensions.
maxcoefficient	10% larger rectangles dimensions values.
mincoefficient	10% smaller rectangles dimensions values.

Table 3.6 – Descriptive variables summary for group *Other*.

3.3. Problem instances

In this section, the most frequently benchmark datasets used over the years for the rectangular 2D-SPP allowing (or not) 90 degrees rotations are identified. Table 3.7 describes the main characteristics of these problem instances, organized by name, number of problem instances, minimum and maximum number of rectangles, organization and source.

In Hopper (2000), the total number of rectangles and the strips width and height similarities were used to generate twenty-one problem instances divided in seven classes. Different rectangles shape were randomly generated with a maximum ratio between the rectangles dimensions equal to seven. The strip shape varies for dimensions ratio between one and three. Hopper and Turton (2001) generated all problem instances with the same characteristics, but the first 35 problem instances (*NTn*) correspond to guillotine patterns, while the next 35 (*NTt*) corresponds to non-guillotine patterns. The problem instances are classified into seven classes, according to the total number of rectangles.

The *N* problem instances proposed by Burke et al. (2004) were generated with constant values for the dimensions of the strip. In a second moment, these strips were randomly divided in small rectangles. In Ferreira and Oliveira (2005) the main focus was to create very heterogeneous rectangles, with extreme differences between the maximum and minimum rectangles dimensions. Imahori and Yagiura (2010) used the generator proposed by Wang and Valenzela (2001) to develop the *iy* problem instances. The main characteristic is the exponential variation of the total number of rectangles per problem instance, varying from 16 to 32,768 rectangles.

Christofides and Whitlock (1977) prefixed a value for the strip area of each *cgcut* test problem instance, and the rectangle's dimensions were generated according to a uniform distribution. As a consequence, rectangles dimensions are proportional to the strip. The ten classes of the *bwmv* developed by Berkey and Wang (1987) (*C01-C06*) and later updated by Martello and Vigo (1998) (*C07-C10*) were proposed using some bin packing problem parameters. Each class has a total of 50 problem instances, with rectangle's dimensions uniformly generated.

In Beasley (1985a) and Beasley (1985b), the *ngcut* and *gcut* were generated based on the strip width parameter. Guillotinable cuts were also considered for both x and y coordinates. Leung et al. (2011) divided the *zdf* in two different groups. The first one

Dataset	Instances[a]	Rectangles[b]	Organization	Source
C	21	17-197	7 classes (*C1-C7*)	Hopper and Turton (2001)
NTn	35	17-199	7 classes (*NTn1-NTn7*)	Hopper (2000)
NTt	35	17-199	7 classes (*NTt1-NTt7*)	Hopper (2000)
N	13	10-3,152	1 class (*N1-N13*)	Burke et al. (2004)
cx	7	50-15,000	1 class (*cx*)	Ferreira and Oliveira (2005)
iy	170	16-32,768	11 classes (*i4-i15*)	Imahori and Yagiura (2010)
cgcut	3	23-623	1 class (*cgcut*)	Christofides and Whitlock (1977)
bwmv	300	20-100	6 classes (*C01-C06*)	Berkey and Wang (1987)
bwmv	200	20-100	4 classes (*C07-C10*)	Martello and Vigo (1998)
ngcut	12	7-22	1 class (*ngcut*)	Beasley (1985a)
gcut	13	10-50	1 class (*gcut*	Beasley (1985b)
zdf	16	580-75,032	1 class (*zdf*)	Leung and Zhang (2011)
AH	360	1,000	6 classes (*AH1-AH6*)	Bortfeldt (2006)
beng	10	20-200	1 class (*beng*)	Bengtsson (1982)
nice	36	25-5,000	8 classes (*nice1-nice5t*)	Wang and Valenzela (2001)
path	36	25-5,000	8 classes (*path1-path5t*)	Wang and Valenzela (2001)

[a] Total number of problem instances.
[b] Minimum and maximum number of rectangles.

Table 3.7 – Benchmark problem instances.

(*zdf1-zdf8*) is composed of medium and large number of rectangles, varying from 580 to 2,532 rectangles. The remaining (*zdf9-zdf16*) are considered extra-large problem instances, varying from 5,032 to 75,032 rectangles.

In Bortfeldt (2006), the *AH* test problems set was developed using two parameters, varying in uniform distributions: rectangles heterogeneity; and the ratio between the strip width and rectangles average width. Bengtsson (1982) developed ten *beng* problem instances, based on the industrial cutting processes found in the manufacturers of excavators and mechanical shovels.

As mentioned before, Wang and Valenzela (2001) developed one of the most used problem generators for the 2D-SPP. The process is recursive based on the variation of rectangles area, according to some parameters defined by the user, and constant strip dimensions (*W=1,000* and *H=1,000*). The *nice* problem instances have a high rectangles shape similarity. In contrast, *path* problem instances have extreme shape variations.

The problem instances for the 2D-SPP presented have some differences, related to rectangles and strip shape variations and intrinsic problem instances characteristics. The main reason for these effects is the use of different types of problem generators. As a consequence, the total number of problem instances representing each of the descriptive variables listed in the previous session is not uniform.

3.4. Principal component analysis

For the exploratory analysis the use of PCA is proposed, with the aiming of decrease the 56 descriptive variables presented. The objective is to reduce the problem dimension to a more manageable size, retaining as much as possible the original information, by aggregating similar descriptive variables into principal components.

The work was developed in two steps. In a first moment, the consistency of each of the six groups of descriptive variables was checked, by analyzing the correlation coefficients

between pairs of descriptive variables of each group. A linear correlation is used as a quantitative measure to validate the assignment of predictors to groups. A value of 0.75 for the correlation coefficient was used. As a reference, the remaining part of this section specifies the procedures performed in all groups. Due to space constraints, we have chosen to show in detail only the results obtained for groups *Area* and *Proportions*. However, the conclusions proposed at the end of this study considers the results of all groups.

To exemplify this first step, Table 3.8 and Table 3.9 summarizes the correlation coefficients between descriptive variables in groups *Area* and *Proportions*, respectively. A total of 12 and 20 coefficient correlations higher than the reference value are found for these groups, and all descriptive variables have at least one high correlation coefficient that justifies the group coherence. Descriptive variables with low positive or negative correlation coefficients do not represent the same characteristic of the problem. To facilitate the interpretation of the problem and maintain the information provided by the problem instances in the original format, the input data was not previously normalized. In some situations, the correlation may have been suffered small effects of any outliers or obvious unusual cases. Additionally, in Appendix 3.A we introduce the correlation coefficients between descriptive variables in groups *Perimeter*, *Dimensions*, and *Widthdimensions*.

	arearatioperc	***arearatioquart***	***areacompnormal***	***areacompquart***	***areacompextr***	***areamean***	***areamed***	***areasdev***
arearatioextr	0.44	0.45	0.38	0.45	0.44	0.89	0.73	0.81
arearatioperc		0.73	0.79	0.89	0.91	0.36	0.14	0.51
arearatioquart			0.82	0.80	0.67	0.38	0.17	0.51
areacompnormal				0.95	0.82	0.32	0.09	0.53
areacompquart					0.94	0.38	0.12	0.61
areacompextr						0.36	0.11	0.57
areamean							0.88	0.88
areamed								0.57

Table 3.8 – Correlation coefficient matrix for group *Area*.

In a second moment, PCA is used individually for each group to reduce the dimensions of the problem. All the PCA are conducted with orthogonal rotation (varimax), and all requirements were reached: the ratio between the sampling size and the number of descriptive variables is greater than five to one; a minimum of 0.5 for overall Kaiser-Meyer-Olkin measure of sampling adequacy; and the Bartlett test of sphericity is statistically significant (< 0.001).

As a result, two components with eingenvalues greater or equal to one were extracted for each of the first five groups: *areacomp* and *areastats* for group *Area*; *perimcomp* and *perimstats* for *Perimeter*; *dimcomp* and *dimstats* for *Dimensions*; *propcomp* and *propstats* for *Proportions*; and *widthcomp* and *widthstats* for group *Widthdimensions*. For the remaining group, *Other*, it was not possible to extract a small number of components, since the descriptive variables in this group are not related with each other (small correlation coefficients). As a result, a total of 19 characteristics was obtained for the 2D-SPP, 10

	propratioextr	*propratioperc*	*propratioquart*	*propcompnormal*	*propcompquart*	*propcompextr*	*propmean*	*propmed*	*propstdev*
aspectratio	−0.21	−0.25	−0.28	−0.31	−0.25	−0.25	0.97	0.97	0.80
propratioextr		0.81	0.72	0.78	0.78	0.80	−0.13	−0.13	−0.01
propratioperc			0.92	0.95	0.97	0.97	−0.21	−0.22	−0.11
propratioquart				0.97	0.96	0.95	−0.25	−0.25	−0.15
propcompnormal					0.97	0.96	−0.27	−0.28	−0.15
propcompquart						1.00	−0.22	−0.22	−0.12
propcompextr							−0.21	−0.21	−0.12
propmean								1.00	0.91
propmed									0.90

Table 3.9 – Correlation coefficient matrix for group *Proportions*.

components and the original nine descriptive variables from group *Other*.

Table 3.10 presents the percentage of variance explained for the components individually in the first five groups. An average of 93% of the data variation is explained by the components extracted, higher than the variation of 91% obtained if the PCA was used with all the descriptive variables simultaneously.

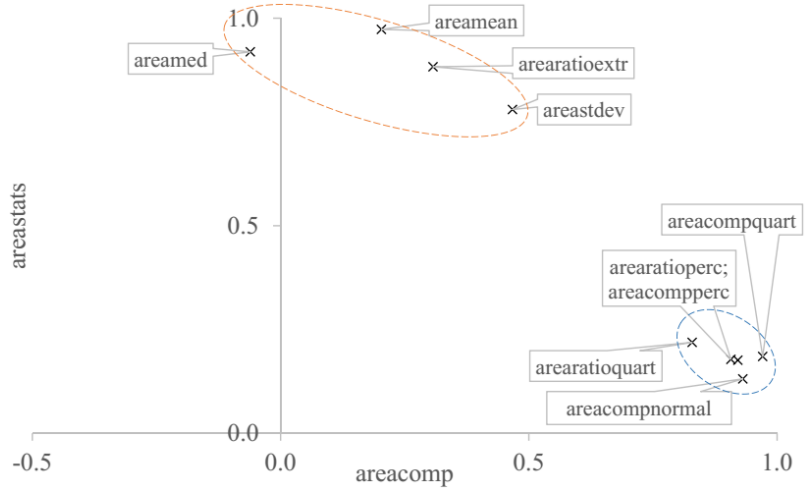
Group	Component	Variance (%)	Cumulative (%)
Area	*areacomp*	62.48	
	areastats	24.92	87.40
Perimeter	*perimcomp*	75.36	
	perimstats	20.16	95.52
Dimensions	*dimcomp*	77.75	
	dimstats	18.00	95.75
Width dimensions	*widthdimcomp*	79.13	
	widthdimstats	16.39	95.52
Proportions	*propcomp*	59.76	
	propstats	33.39	93.15

Table 3.10 – Variance explained by each component for all groups.

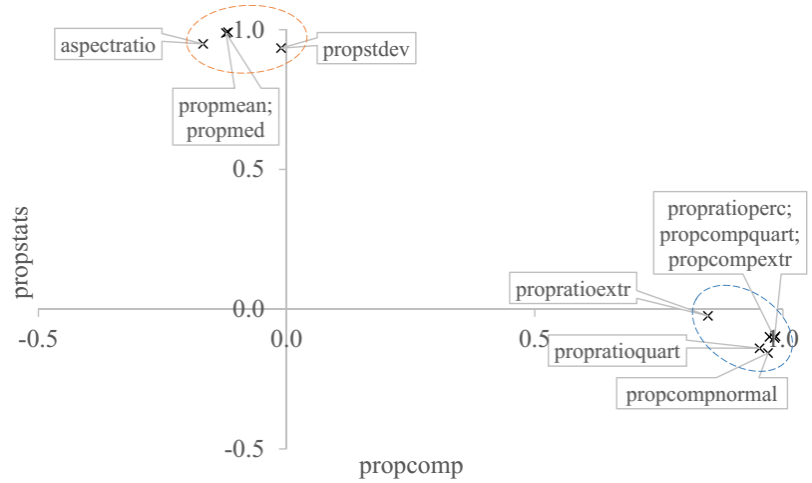
Figure 3.2 represents the descriptive variables' projections along the extracted components for groups *Area* and *Proportions*. In *Area* (Figure 3.2a) there is a clear difference between the descriptive variables with high positive factor loadings for each component. High values for descriptive variables based on ratios (e.g. *areratioperc*) and compositions (e.g. *areacompextr*) establish the component *areacomp*. In contrast, *arestats* is based on classical statistical measures, such as mean, median and standard deviation descriptive variables. One exceptions is *areratioextr*, which is a ratio variable but influences more significantly the *arestats* component.

Group *Proportions* (Figure 3.2b) shows a similar behavior, with component *propcomp* influenced by all ratio and composition descriptive variables, and the *propstats* with all classical statistical measures and the correlated variable *aspectratio*. Additionally, in Appendix 3.B we introduce the representation of the descriptive variables' projections along

the extracted components for groups *Perimeter*, *Dimensions*, *Widthdimensions*, and *Other*.



(a) Group *Area*.

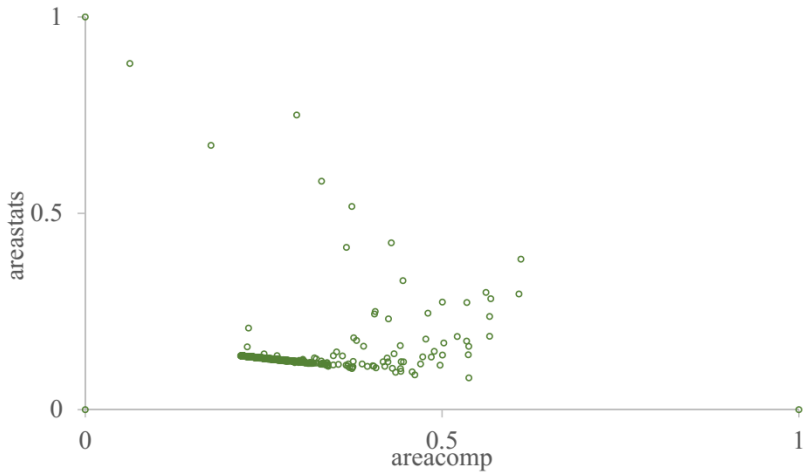


(b) Group *Proportions*.

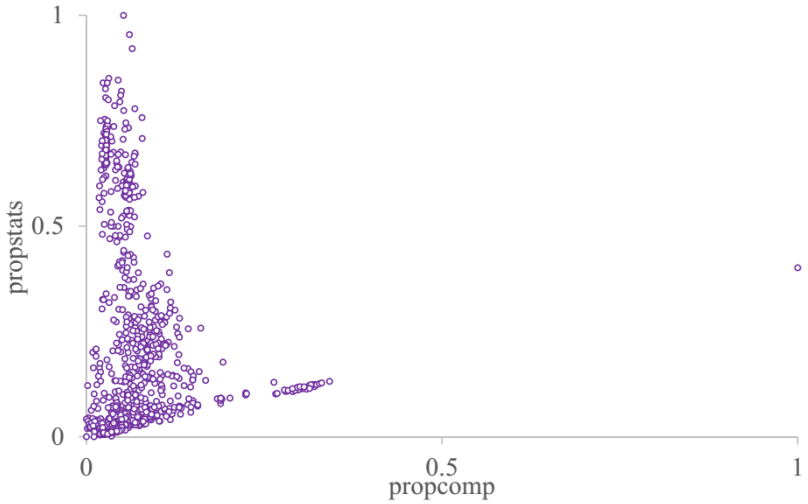
Figure 3.2 – Relations between descriptive variables and components.

Figure 3.3 presents the distribution of the 1,217 problem instances according to the components of *Area* and *Proportions*. To a better visualization of the data, the scores for each problem instance are normalized to a scale between zero and one, according to the maximum and minimum values found for each component. In Appendix 3.C, we present the distribution of the 1,217 problem instances according to the components of *Perimeter*, *Dimensions*, and *Widthdimensions*.

In Figure 3.3a, the highest values for *areastats* are found for seven problem instances *zdf* (*zdf10-zdf16*), a consequence of the high standard deviation and mean value between the largest and smallest rectangles of each problem instance, which can be verified through descriptive variables *areastdev* and *areamed*, respectively. Figure 3.2a shows the strong influence between these descriptive variables and *areastats*.



(a) Group *Area*.



(b) Group *Proportions*.

Figure 3.3 – Distribution of the 1,217 problem instances among components.

Problem instance *cx50* has rectangles with different shape variations, which means that the difference between the largest and smallest rectangles dimensions is very high, and the strip has a square shape, a fact that reflects the high value of the test problem instance for *areacomp* and *propcomp*. Some similar effects can also be found, with less amplitude, for *cx100* and some *path* and *iy* problem instances. In *propstats* (Figure 3.3b) a total of four *AH* problem instances (*AH14*, *AH36*, *AH50* and *AH55*) have a high dispersion between the average value of rectangles proportion compared to the strip. As a consequence, these are problem instances that have a high degree of heterogeneity.

For both graphs, almost all problem instances have similar values. In *Proportions* the problem instances are located near the center of coordinates graph. In *Area* almost all problem instances are between 0.2 and 0.4 for *areacomp* and 0.1 and 0.2 for *areastats*. Therefore, these problem instances have few differences between them, leading to few

variations in the descriptive variables and affecting the analysis of the characteristics of the problem.

3.5. Conclusions

In this paper, we conduct an exploratory research to identify the most significant characteristics for the two-dimensional rectangular 2D-SPP. Initially, a total of 56 descriptive variables were considered, based on parameters and characteristics found in the most used problem generators.

To reduce the complexity the PCA was used to reduce the problem dimensionality. Our analysis suggests that 19 components can explain the problem consistently. A relevant result is the similarity showed by the test problem instance from the literature.

In a second moment, it was verified that the number of problem instances that represent the possible rectangles and strip shape variation must be improved. As a consequence, during the research the need of generation of new problem instances was evident, to overcome the drawback of some missing characteristics in the existing problem instances in the literature.

This study helps in the development of more efficient heuristics for solving the 2D-SPP, by providing a more accurate information on the characteristics of the problem. Future work will describe the relation between the components developed (named as features) with a dependent variable, using regression models in order to allow the prediction of the strip height to be used according to the problem instances characteristics. Also, new descriptive variables will be studied in order to complement the characterization of the problem.

References

Beasley, J. (1985a). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36(4):297–306.

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

Bengtsson, B.-E. (1982). Packing rectangular pieces: A heuristic approach. *The Computer Journal*, 25(3):353–357.

Berkey, J. O. and Wang, P. Y. (1987). Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38(5):423–429.

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3):814–837.

Bortfeldt, A. and Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161.

Burke, E. K., Kendall, G., and Whitwell, G. (2004). A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671.

Christofides, N. and Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44.

Ferreira, E. P. and Oliveira, J. F. (2005). A note on Fekete and Schepers' algorithm for the non-guillotinable two-dimensional packing problem. Technical report, FEUP.

Hall, N. G. and Posner, M. E. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

Hopper, E. (2000). *Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods*. PhD thesis, University of Wales. Cardiff.

Hopper, E. and Turton, B. C. H. (2001). An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57.

Imahori, S. and Yagiura, M. (2010). The best-fit heuristic for the rectangular strip packing problem: An efficient implementation and the worst-case approximation ratio. *Computers and Operations Research*, 37(2):325–333.

Leung, S. C. H. and Zhang, D. (2011). A fast layer-based heuristic for non-guillotine strip packing. *Expert Systems with Applications*, 38(10):13032–13042.

Leung, S. C. H., Zhang, D., and Sim, K. M. (2011). A two-stage intelligent search algorithm for the two-dimensional strip packing problem. *European Journal of Operational Research*, 215(1):57–69.

López-Camacho, E., Terashima-Marín, H., Ochoa, G., and Conant-Pablos, S. E. (2013). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics*, 145(2):488–499.

López-Camacho, E., Terashima-Marin, H., Ross, P., and Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889.

Martello, S. and Vigo, D. (1998). Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44(3):388–399.

Oliveira, J. F., Neuenfeldt Júnior, A., Silva, E., and Carravilla, M. A. (2016). A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, 36(2):197–226.

Silva, E., Oliveira, J. F., and Wäscher, G. (2014). 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, 237(3):846–856.

Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

Wang, P. Y. and Valenzela, C. L. (2001). Data set generation for rectangular placement problems. *European Journal of Operational Research*, 134(2):378–391.

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

Appendix 3.A Correlation analysis

	perimratioperc	*perimratioquart*	*perimcompnormal*	*perimcompquart*	*perimcompextr*	*perimmean*	*perimmed*	*perimstdev*
perimratioextr	0.72	0.63	0.72	0.75	0.82	0.83	0.70	0.90
perimratioperc		0.91	0.96	0.97	0.95	0.51	0.38	0.64
perimratioquart			0.93	0.92	0.87	0.42	0.31	0.54
perimcompnormal				0.99	0.97	0.49	0.34	0.63
perimcompquart					0.99	0.53	0.39	0.66
perimcompextr						0.60	0.46	0.74
perimmean							0.96	0.94
perimmed								0.82

Table 3.11 – Correlation coefficient matrix for group *Perimeter*.

	dimratioextr	*dimratioperc*	*dimratioquart*	*dimcompnormal*	*dimcompquart*	*dimcompextr*	*dimmean*	*dimmed*	*dimstdev*
vardim	0.83	0.58	0.51	0.56	0.60	0.67	0.97	0.99	0.93
dimratioextr		0.72	0.63	0.72	0.75	0.82	0.90	0.87	0.90
dimratioperc			0.91	0.96	0.97	0.95	0.64	0.60	0.65
dimratioquart				0.93	0.92	0.87	0.55	0.53	0.56
dimcompnormal					0.99	0.97	0.62	0.58	0.64
dimcompquart						0.99	0.66	0.62	0.68
dimcompextr							0.73	0.69	0.75
dimmean								0.99	0.99
dimmed									0.95

Table 3.12 – Correlation coefficient matrix for group *Dimensions*.

	widthdimratioperc	*widthdimratioquart*	*widthdimcompnormal*	*widthdimcompquart*	*widthdimcompextr*	*widthdimmean*	*widthdimmed*	*widthdimstdev*
widthdimratioextr	0.67	0.59	0.68	0.70	0.78	0.90	0.86	0.90
widthdimratioperc		0.90	0.95	0.96	0.95	0.60	0.57	0.60
widthdimratioquart			0.92	0.90	0.85	0.53	0.51	0.52
widthdimcompnormal				0.99	0.96	0.60	0.57	0.61
widthdimcompquart					0.99	0.62	0.59	0.63
widthdimcompextr						0.69	0.66	0.71
widthdimmean							0.99	0.99
widthdimmed								0.95

Table 3.13 – Correlation coefficient matrix for group *Widthdimensions*.

Appendix 3.B Descriptive variables' projections

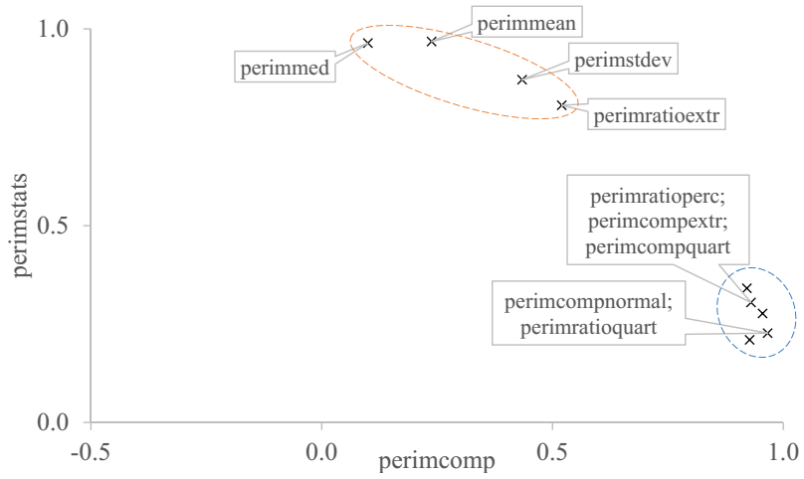


Figure 3.4 – Relations between variables and components for group *Perimeter*.

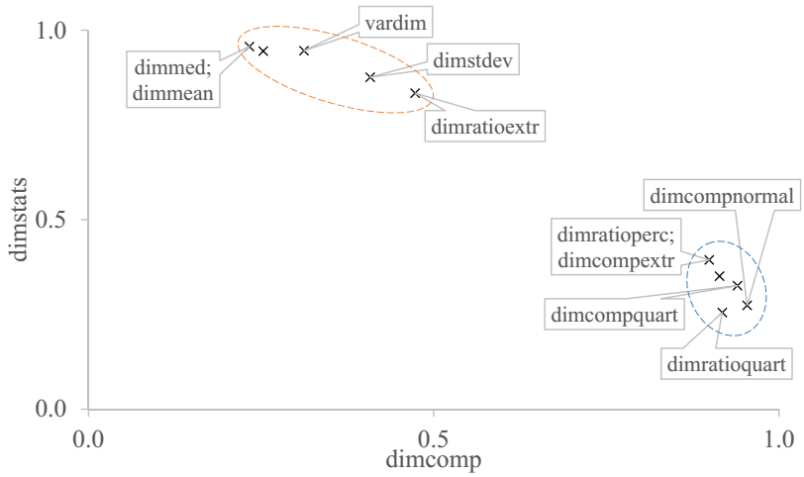


Figure 3.5 – Relations between variables and components for group *Dimensions*.

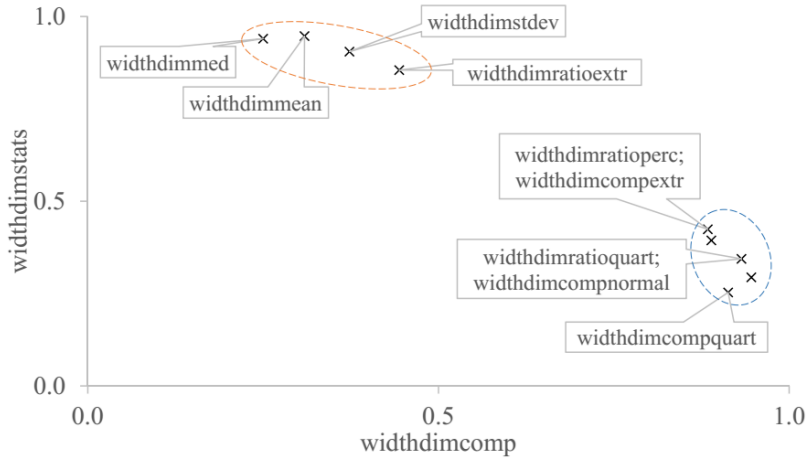


Figure 3.6 – Relations between variables and components for group *Widthdimensions*.

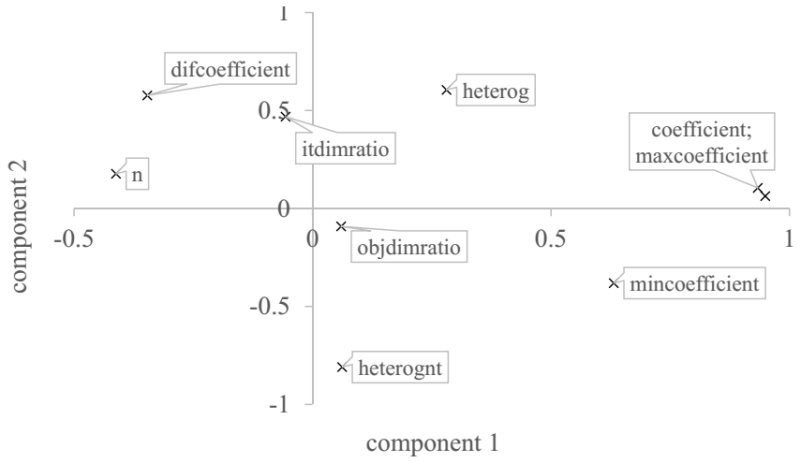


Figure 3.7 – Relations between variables and components 1 and 2 for group *Other*.

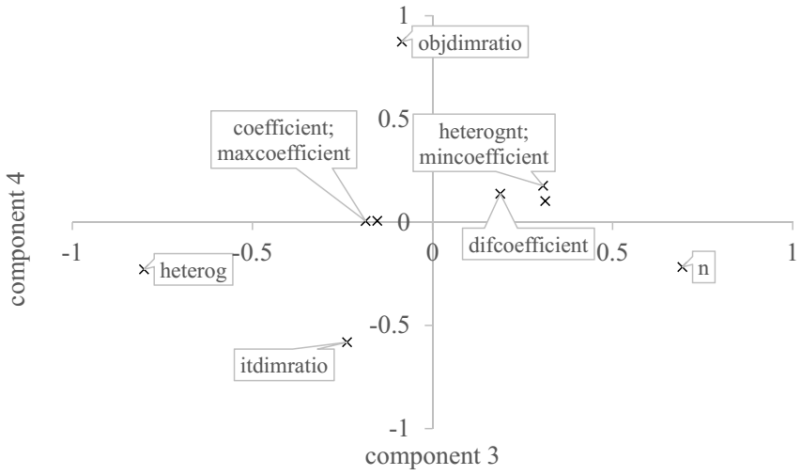


Figure 3.8 – Relations between variables and components 3 and 4 for group *Other*.

Appendix 3.C Distribution of the problem instances

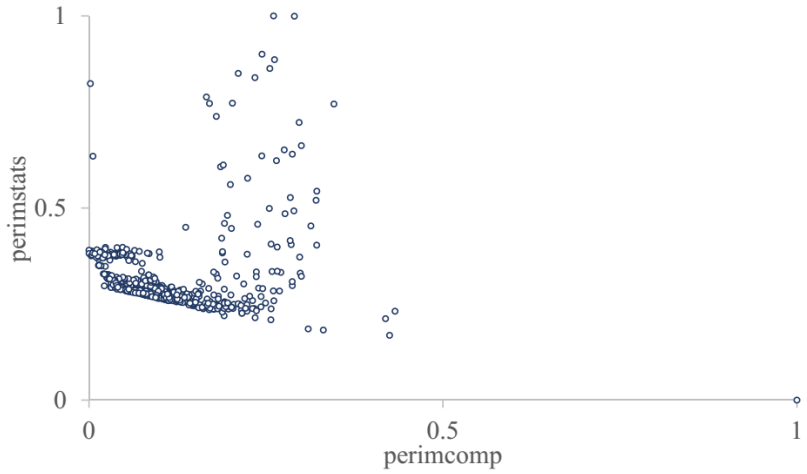


Figure 3.9 – Distribution of the 1,217 problem instances among components for group *Perimeter*.

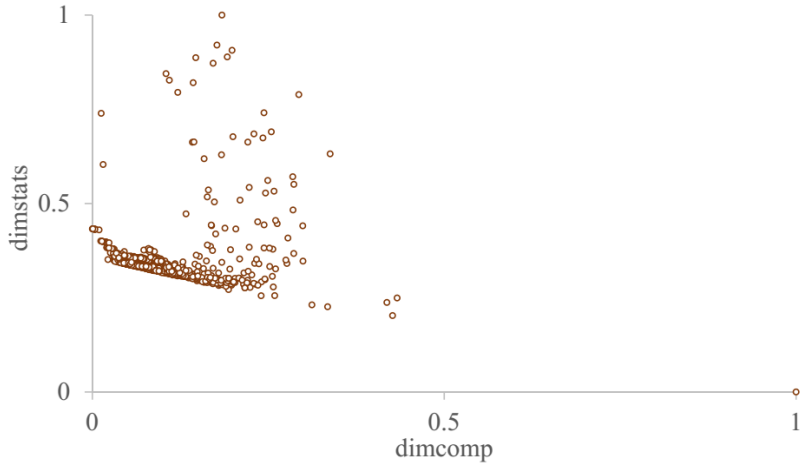


Figure 3.10 – Distribution of the 1,217 problem instances among components for group *Dimensions*.

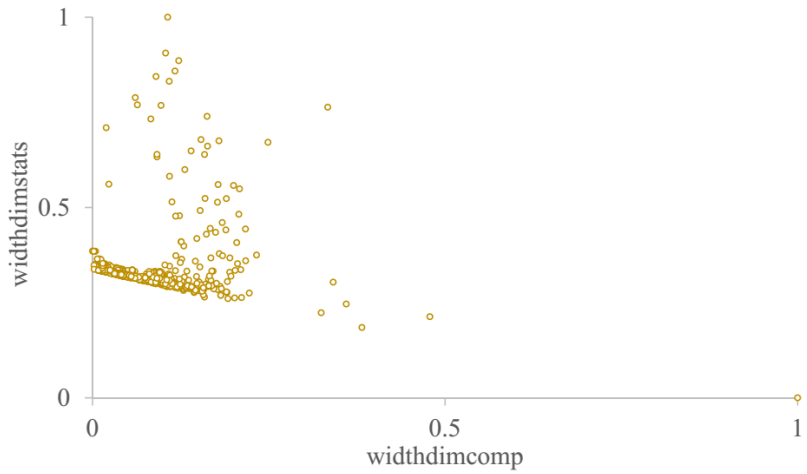


Figure 3.11 – Distribution of the 1,217 problem instances among components for group *Widthdimensions*.

CHAPTER 4

Data Mining Based Framework to Assess Solution Quality for the Rectangular 2D-SPP

Alvaro Neuenfeldt Júnior[1] · **Elsa Silva**[1] · **A. Miguel Gomes**[1] · **Carlos Soares**[1] · **José Fernando Oliveira**[1]

Abstract

Different versions of the rectangular two-dimensional strip packing problem (2D-SPP) can be found in many industrial contexts. Furthermore, one of the most important steps of solving this problem is to use heuristic approaches, which provide good solutions. However, heuristic approaches do not guarantee optimality, and lower bounds are generally used to provide information about solution quality. Almost none of the lower bounds available in the 2D-SPP literature allow rectangles to rotate 90 degrees. The only exception is the area lower bound, which is based on the area of all rectangles and does not depend on the rectangles' rotation. The main problem is the lack of accuracy of the area lower bound, mainly in problem instances where the optimal solution has waste space between the rectangles. The objective of this research is to propose a data mining-based framework capable of assessing the quality of heuristic solutions for the 2D-SPP with 90 degree rotations. A regression model was fitted by comparing the strip height solutions obtained with the bottom-left-fill heuristic and the 19 predictors provided by problem characteristics. The Random forest was selected as the data mining technique with the best level of generalisation for the problem, and 30,000 problem instances were generated to represent different 2D-SPP variations found in real-world applications. Height predictions for new problem instances can be found in the regression model fitted. In this study, we demonstrate that the data mining-based framework proposed is consistent and that it can be applied to find predictions in other instances of combinatorial optimisation problems, specifically the cutting and packing problems.

Keywords

Strip packing problems; Cutting and packing problems; Data mining; Heuristics; Regression analysis.

[1]INESC TEC, Faculdade de Engenharia, Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4600-001 Porto, Portugal.

4.1. Introduction

The two-dimensional strip packing problem (2D-SPP) consists of packing a set of small rectangles into the strip of fixed width and virtually infinite height, minimising the required height. The small rectangles must be positioned without any overlap between them and completely inside the strip. According to the typology of Wäscher et al. (2007), this description fits in the definition of cutting and packing problems as a 2D rectangular Open Dimension Problem (ODP). In the 2D-SPP tackled in this work, the rectangles should be orthogonally packed inside the strip and are allowed to rotate 90 degrees. Typical 2D-SPP applications are found in manufacturing, including the cutting of metals, textiles or paper rolls.

The 2D-SPP is a NP-hard problem and can be solved either by exact methods or heuristic approaches (Martello et al., 2003; Alvarez-Valdés et al., 2008). Exact methods are based on mathematical programming models and are able to obtain optimal solutions or, at least, determine the solution quality. However, these methods are not able to tackle the complexity and size of real-world applications. Instead, heuristic approaches have been used in these situations, given the low computational times required when compared to exact methods. However, heuristic approaches do not guarantee optimality and do not provide any information about solution quality (Hopper and Turton, 2001b; Ntene and van Vuuren, 2009).

Lower bounds have been used to overcome this limitation in various cutting and packing problems (Fekete and Schepers, 2001; Lodi et al., 2002; Martello et al., 2003). However, almost none of the lower bounds available in the 2D-SPP literature allows rectangles to rotate 90 degrees, meaning that they cannot be used for solving the problem tackled in this work, as the lower bound can be greater than the optimal solution. The only exception is the area lower bound proposed by Martello et al. (2003), which is based only on the area of all rectangles and the strip width. The main problem is the lack of accuracy of the area lower bound, mainly in problem instances where the optimal solution has waste space between the rectangles.

The main objective of this research is to propose a data mining-based framework capable of assessing the quality of heuristic solutions for the 2D-SPP with 90 degree rotations, comparing the strip height of a given solution with a prediction of the height required to pack all rectangles into the strip. The prediction is obtained by fitting a regression model with data mining techniques. Besides providing a quality measure to heuristic solutions, the framework can also be used to develop more precise stopping criteria in local search algorithms with the goal of avoiding long computational times. Traditional stopping criteria rely on a high number of iterations to ensure solution quality.

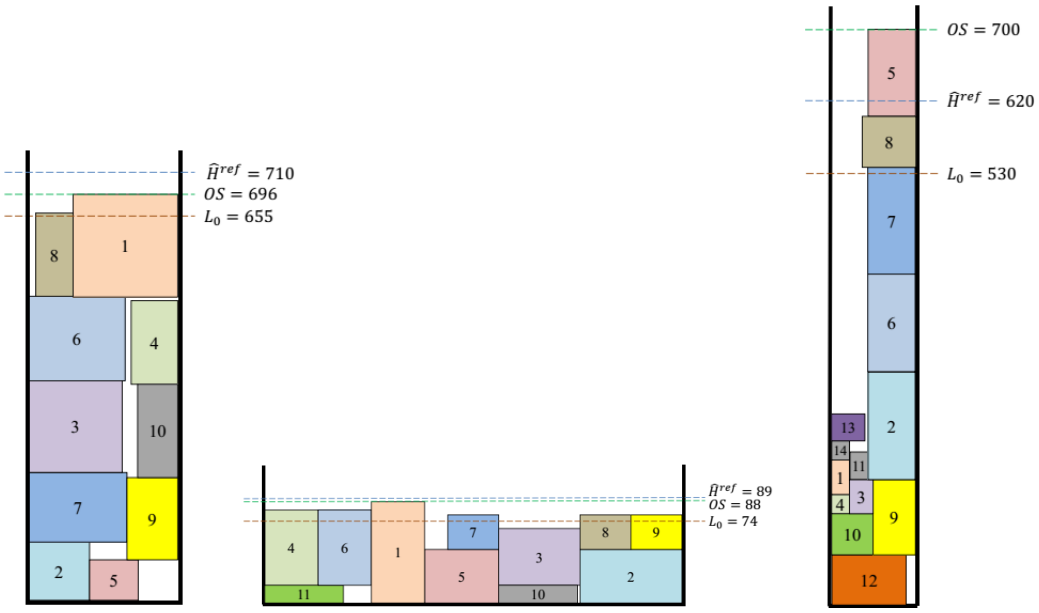
The proposed data mining-based framework can be extended to other cutting and packing problems, such as the bin packing problem (minimise number of used bins), the knapsack problem (maximise the total value in the knapsack), and the cutting stock problem (minimise the amount of scrap).

The framework was developed around a data mining approach, in which the predictors (explanatory variables) were developed according to relevant characteristics of the problem in order to provide an adequate way of measuring a known response variable for predefined

observations (or problem instances). In the 2D-SPP, this known response variable is related to the strip height, which is calculated using constructive heuristics and local search algorithms. The data mining approach uses the predictors and the known response variable to fit a regression model with the goal of predicting the response variable for new problem instances. Different data mining techniques were tested to fit the regression model and to ensure an adequate generalisation level and the predictions' accuracy for new problem instances.

To illustrate the problem and the potential use of the predictions obtained by regression models fitted with the data mining-based framework proposed in this research, three examples comparing the predictions with optimal solutions and area lower bound values are presented.

Firstly, the problem instance *gcut1* proposed by Beasley (1985b) is presented (Figure 4.1a). The wasted space found in the strip for the optimal solution (6%) is one of the largest for benchmark problem instances. The distance from the prediction ($\hat{H}^{ref} = 710$) to the optimal solution ($OS = 696$) is lower when compared with the distance between the area lower bound ($L_0 = 655$) and the optimal solution. Although $\hat{H}^{ref}$ is higher than OS, $\hat{H}^{ref}$ is more realistic and less optimistic when compared with the L_0, which cannot feasibly be reached by any local search.



(a) *gcut*1. (b) *pt*10_23_25. (c) Toy example.

Figure 4.1 – Examples of height prediction ($\hat{H}^{ref}$), area lower bound (L_0), and optimal solution (OS).

The second example is the problem instance *pt10_23_25* (Figure 4.1b), generated using the 2DCPackGen (Silva et al., 2014). The prediction ($\hat{H}^{ref} = 89$) is near the optimal solution ($OS = 88$). The area lower bound ($L_0 = 74$) is too optimistic, 16% below the optimal solution. The total wasted space available in the optimal solution is 16%, which is

higher than the optimal waste of problem instance *gcut1*. Finally, Figure 4.1c shows an extreme toy example for a narrow strip where the total wasted space available in the optimal solution is 33% of the total area. The area lower bound ($L_0 = 530$) is very different when compared with the optimal solution ($OS = 700$). When compared with the area lower bound, the use of the prediction ($\hat{H}^{ref} = 620$) as stopping criterion is recommended.

The paper is organised as follows. Section 4.2 presents the literature review about the 2D-SPP and similar data mining approaches for cutting and packing problems. Section 4.3 presents the data mining-based framework. Section 4.4 focuses on the problem knowledge. Section 4.5 presents the data mining approach. Section 4.6 presents an assessment of the framework performance. Finally, Section 4.7 presents some final remarks and future research ideas. Additionally, in Appendix 4.A we introduce basic notation used in this research for quick reference.

4.2. Literature review

The 2D-SPP is one of the most explored cutting and packing problems in real-word contexts.

In Oliveira et al. (2016), an overview on heuristics for the 2D-SPP was extensively explored. The relevant literature was reviewed and links between the most frequently used heuristics and the 2D-SPP's characteristics were investigated. Most of the works published in recent years present successful heuristic approaches to the 2D-SPP with 90 degree rotations, while some also present heuristic approaches developed to tackle specific real-world problems. Validation of these approaches is usually done by comparison with several "standard" benchmark problem instances. This means that there are no types of absolute measure regarding the quality of solutions achieved by the proposed heuristic approaches.

A typical measure to assess solution quality in NP-hard combinatorial optimisation problems is the usage of lower bounds as a reference value. However, that it is not a viable option for the 2D-SPP with 90 degree rotations due to the lack of strong lower bounds. The only viable option is the area lower bound from Martello et al. (2003).

To access the quality of the solutions, two heuristics (Neveu et al., 2008; Wei et al., 2009) adopted non-zero-waste benchmark problem instances from datasets *gcut* (Beasley, 1985b), *cgcut* (Christofides and Whitlock, 1977), *ngcut* (Beasley, 1985a) and *bwmv* (Berkey and Wang, 1987). An interesting fact observed in both studies is the peculiar usage of lower bounds. Wei et al. (2009) presents a least wasted first heuristic, which uses the lower bounds proposed by Martello et al. (2003) in the non-rotated 2D-SPP as a reference value to measure the quality of the solutions. Neveu et al. (2008) proposed a method that adjusts the rectangles' positioning in the strip based on the location of the maximum holes. As in Wei et al. (2009), the solutions are evaluated using the lower bounds in the non-rotated 2D-SPP of Martello et al. (2003). Lower bounds that do not allow rectangles to rotate should not be used as reference values in the rotated case, since the optimal solution could be better.

Data mining approaches in cutting and packing problems are related with the process of converting problem information into measurable factors in order to reflect the main

problem characteristics and compare the algorithms' performance with different types of problem instances. However, the literature combining knowledge discovery with data mining techniques within the context of cutting and packing problems is not extensive. A literature review about the main contributions in this research field is presented in the next paragraphs. These contributions were fundamental to develop the data mining framework proposed in this research to predict the strip height.

The work by Smith-Miles and Lopes (2012) was one of the first to provide a comprehensive understanding about the methodologies used to measure the difficulty of problem instances. An extensive study on the most relevant characteristics to measure algorithm performance was conducted for six different combinatorial optimisation problems (assignment, travelling salesman, knapsack, bin-packing, graph colouring, and timetabling).

A framework to compare the strengths and weaknesses of optimisation algorithms was described by Smith-Miles et al. (2014), starting with the problem instance selection (or generation), feature selection, dimension reduction using principal component analysis and, finally, the development of algorithm performance metrics. A case study involving the graph coloring problem was presented. This framework is an improvement of the methodology proposed by Rice (1976) for the algorithm selection problem in order to predict the performance level based on characteristics found in the problem instances.

To replace an intuitive process in order to select the most important features of the irregular two-dimensional bin packing problem, López-Camacho et al. (2010) defined a problem-state representation to improve the performance of a hyper-heuristic algorithm. In López-Camacho et al. (2013), a principal component analysis was used as a knowledge discovery method to understand the 1D and the 2D irregular bin packing problems' structure and its relation with the heuristics' performance. All data obtained was used as input to develop a unified hyper-heuristic (López-Camacho et al., 2014) capable of adapting its behaviour to each problem instance.

Simplifying the problem structure using linear correlation analysis was proposed in Santoyo et al. (2015) in order to characterise the difficulty of the problem instances for the bin packing problem. A linear correlation analysis was conducted in order to reduce a total of 27 features selected from the literature to only five metrics and used to compare the quality of six algorithm solutions. Perez et al. (2004) incorporated machine learning techniques in order to predict the performance of seven heuristic algorithms for new problem instances in the bin packing problem. A set of critical characteristics from solved problem instances was used as input data. A comparison between the augmented neural network and minimum bin slack heuristic for the one-dimensional bin packing problem was proposed in Almeida and Steiner (2016), incorporating characteristics from problem instances found in the literature.

For the 0-1 knapsack problem, Hall and Posner (2007) used a set of computed characteristics in order to develop a methodology for predicting the best procedure (branch-and-search or dynamic programming algorithm) that should be used for each particular problem instance. A regression model was fitted to predict the relative performance, considering six typical attributes found in the knapsack problem (problem size, the characteristics of rectangle value and size, the relationship between rectangle value and rectangle size, knapsack capacity and characteristics of the linear relaxation solution). These attributes were ex-

tracted to ten measures, incorporated into the regression model.

4.3. Data mining-based framework

This section focuses on introducing and describing the proposed framework (Figure 4.2). The remainder of this section presents the concepts on each framework step.

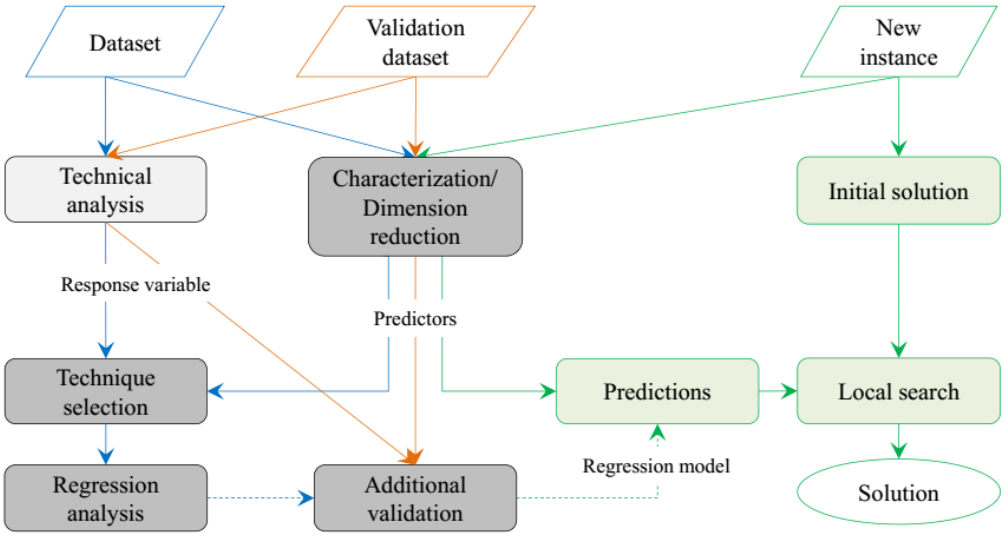


Figure 4.2 – Data Mining-based Methodological framework. Each step is presented in the following sections: Dataset (section 4.4.1), Technical analysis (section 4.4.2), Characterization and Dimension reduction (section 4.5.1), Technique selection (section 4.5.2), Regression analysis (section 4.5.3), and Validation dataset and Additional validation (section 4.5.4).

The data mining-based framework was developed around the processing of two datasets used to fit and validate the regression model, which can be applied to predict the strip height in new problem instances. The Dataset (in blue) was used to develop the regression analysis and to fit the regression model, which is capable of predicting the strip height. The Validation dataset (in orange) was generated to perform adjustment evaluation tests on the regression model fitted with the Dataset, verifying the level of generalisation and predictions' accuracy in different problem instances. Finally, new problem instances (in green) can be solved by different local search algorithms, using as stopping criterion the strip height predictions provided by the regression model deployed after the additional validation step. Dashed arrows represent the transfer of a fitted regression model from one stage of the framework to another, and solid arrows are related with the transfer of input data (variables and datasets) to fit into or to use in the regression model. The knowledge discovery step is identified in light grey, and the data mining approach steps are identified in dark grey.

A framework's cornerstone is to have a deep knowledge about the problem. This can be achieved through a 2D-SPP literature review (Oliveira et al., 2016; Neuenfeldt Júnior et al.,

2017). The Problem knowledge (Section 4.4) was divided into two steps: the generation of problem instances to be used as Dataset, and the technical analysis to describe the heuristics applied in order to obtain solutions for each problem instance. These solutions and the area lower bound were used as reference to define the known response variable for the regression analysis.

The problem instances of Dataset (Section 4.4.1) must be able to describe the wide quantity of characteristics and attributes of the strip packing space, exploring the problem in different perspectives (Smith-Miles et al., 2014). The key is to avoid the problem representation with imprecise or biased information, and at the same time to include different characteristics and perspectives of the problem.

The Technical analysis (Section 4.4.2) requires the calculation of feasible solutions and a lower bound in each problem instance. As mentioned before, the heuristics have been used in real size problems, and are generally divided into constructive heuristics and improvement procedures (Oliveira et al., 2016). In constructive heuristics, the rectangles are successively positioned inside the strip until the last rectangle, and a complete solution is generated. In the improvement procedure (specifically for the local search), a solution, which can be obtained using some constructive heuristic, is improved by applying consecutive changes to the initial input sequence or to the arrangement of the rectangles in the strip. The local search runs until a stopping criterion is reached (Hopper and Turton, 2001a).

In the best-case scenario, the reference value should be the value of the optimal solution. However, with the exception of zero-waste problem instances, the value of the optimal solution is not known. Lower bounds have been used to bridge this gap.

Generally, the lower bounds are developed around mathematical integer formulations or problem relaxations. In the 2D-SPP, the most common relaxation is the one-contiguous bin packing problem (Alvarez-Valdes et al., 2009). Another possibility is to consider the relationship between the geometrical characteristics of the rectangles and the strip, transforming the rectangle dimensions and the strip's fixed dimension using the dual feasible function or calculating the total area that the rectangles can occupy inside the strip (Martello et al., 2003; Boschetti and Montaletti, 2010).

After the problem knowledge, in the Data mining approach (Section 4.5), the development of predictors that fully characterise the problem can be achieved by using concepts found in different cutting and packing problems, not only in the 2D-SPP. In addition, the study of how the problem instances were created and of its generation parameters is an important source of information. In general, the characterization considered in the development process is related to an exploratory context, considered in Smith-Miles and Lopes (2012) and Smith-Miles et al. (2014) as something similar to an art, combining multidisciplinary knowledge about the problem. Each selected feature must be relevant to the problem, avoiding spurious situations or measures that require long or non-feasible processing times to extract information from the problem instances (Hall and Posner, 2007).

The set of predictors selected is incorporated into a high-dimensional space, which can be reduced in order to facilitate the characterisation of the problem. Combinatorial optimisation approaches are useful in reducing the problem dimensions, until a feasible and desirable subset of important predictors is reached, from a set of original descriptive variables in the problem dimensions, given different approximation criterion (Cadima et al.,

2004). Another possibility is the use of mathematical dimension reduction techniques as principal component analysis or factorial analysis to produce uncorrelated components, retaining most of the information available in the predictors (López-Camacho et al., 2013). For example, the principal component analysis is capable of reducing the problem's predictors combinations to only two components, allowing the projection in a two-dimensional visualisation of similarities and differences between a set of observations, as well as the level of correlation between all predictors.

In this study, the Characterisation of the problem and Dimension reduction (Section 4.5.1) was developed around the methodology proposed by Neuenfeldt Júnior et al. (2017), maintaining all the characteristics observed in the original descriptive variables for the interpretation of how the problem characteristics affect the performance of the prediction.

Regression analysis based on data mining techniques provides an useful process for extracting information about the problem from a huge volume of data (Perez et al., 2004; Bastos et al., 2014). Modelling techniques can be used to describe the relationship between the characteristics and the predictions. The most common approaches are related with linear, logistic and polynomial regression models (Kumar and Vijayalakshmi, 2011). Stepwise, ridge, least absolute shrinkage and selection operator are robust techniques used mainly to solve situations with outliers, non-parametric data, or multicollinearity between variables. In contrast, they require more time to process all information (Brazdil et al., 2003). To conduct the data mining Technique selection (Section 4.5.2), the comparison between the performance of different techniques can be evaluated using a statistical test of hypothesis (Demšar, 2006), comparing the coefficient of determination, R^2, obtained with the use of different problem instances.

To develop the Regression analysis (Section 4.5.3), the definition of predictors and one known response variable are required. In this research, components obtained during the dimension reduction process are used as predictors. The response variable is obtained by the gap between the best random weight local search solution (using the bottom-left-fill as constructive heuristic) and the area lower bound. Finally, the problem instances in Dataset generated by the 2DCPackGen problem generator (Silva et al., 2014) are proposed to represent the problem. After the regression analysis, an Additional validation (Section 4.5.4) test must be conducted to validate the level of generalisation of the regression model and an absolute predictions' accuracy, using additional random problem instances and benchmark problem instance datasets (Validation dataset) to predict the strip height in different conditions.

To Assess framework performance (section 4.6), the Framework usage (Section 4.6.1) to predict the strip height for new problem instances is explored. Firstly, the characteristics of a new problem instance must be quantified according to the descriptive variables used to represent the 2D-SPP. Secondly, these quantified descriptive variables are converted into predictors during the dimension reduction process. Finally, the prediction's performance as stopping criterion to avoid long computational times is evaluated.

The Prediction performance analysis (Section 4.6.2) was developed to verify if small perturbations in the strip height predictions are capable of substantially change the local search stopping process behaviour, considering the relation between the quality of the solutions, the number of iterations and variations in the predictions.

4.4. Problem knowledge

This section focuses on the introduction of what's required in order to use the knowledge discovery about the problem as input data for the regression analysis. Firstly, Section 4.4.1 describes the parameters and configurations to generate the problem instances of Dataset. Section 4.4.2 describes the bottom-left-fill heuristic concepts and the area lower bound, used in this research as a reference value for the problem instances. Both heuristic and area lower bound were used to define the response variable of the regression model.

4.4.1 Dataset

The use of data mining techniques to predict the strip height requires a large problem instance dataset, mainly to accurately represent the limits and behaviour of the problem, and also to consider the influence of aspects and characteristics that affect the quality measure of local search solution values (Bortfeldt, 2006).

Problem instances in the 2D-SPP are divided into two types: zero-waste and non-zero-waste. The zero-waste problem instances are generated by successively cutting a strip in smaller rectangles, and consequently the optimal solution does not allow waste space after positioning all the rectangles inside the strip. In non-zero-waste problem instances, the optimal solution is generally not known. The area lower bound value is almost equal to the optimal value when the optimal solution allows less waste space. An optimal solution with more wasted space has more uncertainty in the optimal arrangement of the rectangles in the strip, resulting in a less accurate area lower bound.

The problem instance generation is directly related with the problem generator's ability to combine rectangles and strip geometrical possibilities (Wang and Valenzela, 2001). Based on a fixed input information, a greater variation of the characteristics demands the use of the problem generator's capability to correctly interpret the information available, returning as output a proportional number of problem instances.

In this study, the Dataset is composed by 30,000 problem instances created using the 2DCPackGen problem generator (Silva et al., 2014). The 2DCPackGen allows the generation of a large number of problem instances, using different parameters under controllable aspects, and ensures the reproducibility of the data. For the generation, information on a minimum and maximum value for a set of parameters is required.

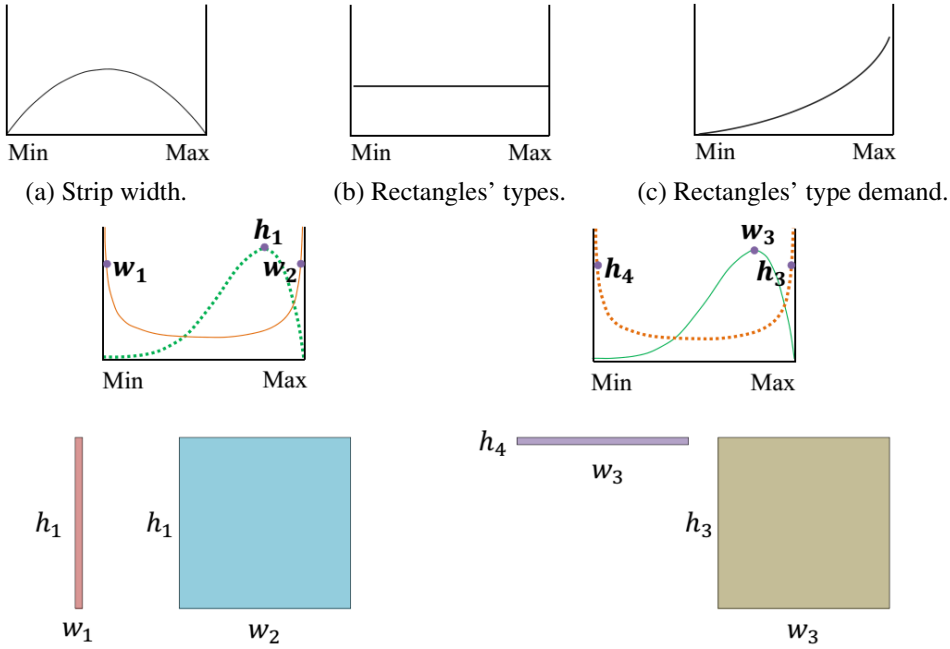
Specifically in the 2D-SPP, the parameters that must be defined are: the rectangle minimum and maximum size dimension (1-100); the strip minimum and maximum width (10-1,000); the minimum and maximum number of different rectangle types (5-500); and the minimum and maximum number for the rectangle type demand (1-10). The problem instance variation is fitted using a beta probabilistic distribution (Gupta and Nadarajah, 2004). Different curve behaviours represent different geometric rectangles and strip shapes. The minimum and maximum values of each parameter were defined based on the most common values found in the benchmark problem instances of the literature.

To avoid the generation of similar problem instances, 10 different classes were developed, based on different characteristics of the size and shape of the rectangles.

In each class, the strip width, the number of different rectangle types and the rectan-

gle type demand vary, respectively, according to two, five and three distribution curves, representing 30 different combinations, named as subtypes. For each subtype, 100 similar problem instances were generated, resulting in a total of 3,000 problem instances in each class.

For example, the *pt7_28_9* is a class 7 problem instance (Figure 4.3), where the probability of the rectangles being narrow (w_1) or long (w_2) and tall (h_1) is higher. In this research, we deal with the rotation condition, so the rectangle dimension is not fixed, which also allows obtaining in class 7 long (w_3) and short (h_4) or tall (h_3) rectangles. The subtype of this problem instance is 28, where the probability of the strip width being of an intermediate size is higher, different rectangle types are defined by an uniform distribution, and a large number of rectangles in each type is highly probable.



(a) Strip width. (b) Rectangles' types. (c) Rectangles' type demand.

(d) Rectangles' shape.

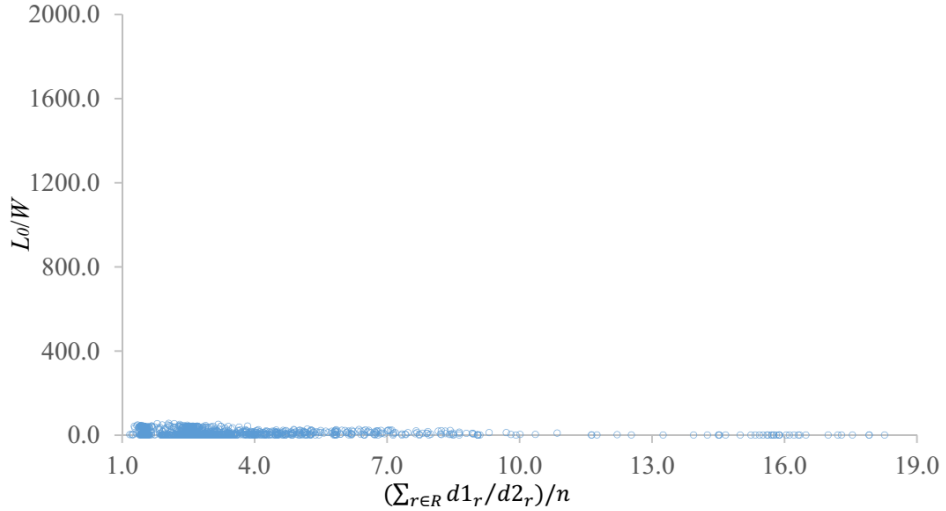
Figure 4.3 – Distribution curves used to generate the problem instance *pt*7_28_9.

Most of the benchmark problem instances of the literature are very similar, which does not provide sufficient diversity to cover some important aspects of the strip packing characteristics. Figure 4.4 illustrates the behaviour of the 1,270 most used benchmark problem instances from the literature and the 30,000 generated problem instances for Dataset, based on two basic metrics: for all rectangles, $r \in R$, the mean aspect ratio shape[1] between the maximum, $d1_r$, and minimum, $d2_r$, rectangle dimensions and the strip aspect ratio[2] between the area lower bound, L_0, and the strip width W.

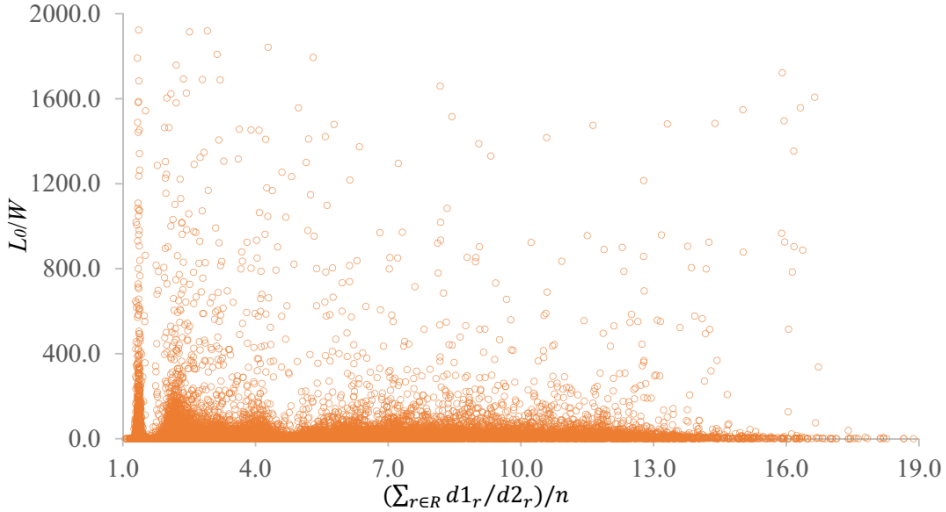
A considerable number of practical applications of the 2D-SPP involve the cutting of strip and rectangle geometry closest to the square shape. This trend is reflected in the

[1] $(\sum_{r \in R} d1_r/d2_r)/n$.
[2] L_0/W.

way the benchmark problem instances were generated over the years, a fact verified by the density of points in the lowest part of Figure 4.4a, where strips and rectangles are geometrically square, rather than narrow. Most of the generated problem instances also have this type of geometry. However, the objective of the regression analysis is to represent most of the possible behaviours of the problem through the generation of different problem instances. For example, a series of problem instances are formed by narrow strip and square rectangles (points at the top of Figure 4.4b), by square strip and narrow rectangles (points at the right), or by narrow strip and rectangles (points at the upper right corner).



(a) 1,270 benchmark problem instances.



(b) 30,000 Dataset problem instances.

Figure 4.4 – Distribution of the problem instances ratio between strip aspect ratio shape and all rectangles mean aspect ratio shape.

4.4.2 Technical analysis

In this study, the Bottom-Left-Fill (BLF) heuristic (Chazelle, 1983) was selected as constructive heuristic. The BLF is a positioning-based constructive heuristic that fits individual rectangles into the lowest left position free space identified in the strip. A complete solution is obtained when the last rectangle is packed into the strip.

The BLF is one of the most used constructive heuristics in the 2D-SPP literature, being efficient to solve both small and large problem instances. The BLF considers the free spaces between already packed rectangles as feasible positions, which is impossible in the Bottom-Left (BL) heuristic proposed by Baker et al. (1980). Thus, the BLF is much more complex ($O(n^3)$, where n is the number of rectangles) when compared with the BL ($O(n^2)$).

Originally, the rectangles in the BLF were sorted by non-increasing area (Riff et al., 2009). With the aim of increasing the search solution space, Hopper and Turton (2001a) proposed the use of four different sequences (by rectangle height, width, perimeter, and area), and the solution with the smaller height is selected. Based on this idea, we developed a BLF heuristic with a local search containing 100 independent sequences, using different types of input sequences, named as random weight local search. In the first four sequences, the rectangles are ordered by decreasing area, perimeter, width, and height. The following 76 sequences relate to a random weighted order procedure, divided into four parts containing 19 sequences, based on area, perimeter, width, and height. Finally, the last 20 sequences are fully randomly generated.

In the random weight procedure, each rectangle has a probability of being chosen to occupy the first position of the sequence. Thus, the rectangles' ordering is not completely random. This probability relates to the rectangles' geometric characteristics (area, perimeter, width or height dimensions). At each time, a rectangle is added to the sequence, and the probabilities of the remaining rectangles to be positioned in the sequence are updated.

The value used as a reference to define which rectangle will be positioned is defined randomly, from 0 to 0.999. The rectangles' probability is summed until a random value is reached. When this sum is greater than the reference value, the rectangle chosen to be positioned is the last but one.

The objective of the 100 sequences is to have a wide search space and boost the quality of the solution, avoiding local optima solutions. Finally, the best solution of the random weight local search is the one with the smallest height from all 100 solutions, and is defined as the reference strip height for the regression analysis, represented by H^{ref}. The best solution is used in the calculation of the response variable for the regression model.

The relaxation of the original problem in simple formats allows the calculation of the lower bounds. As an example, the 2D-SPP can be relaxed in a one-dimensional contiguous bin-packing problem, successively cutting the rectangles' height (or width) for one unit, while maintaining the second dimension in the original format. In this case, the sum of the height of all bins is the solution for the new problem. While it is not feasible for the original 2D problem, it can be considered as a valid lower bound (Martello et al., 2003). Other concepts explored in the literature determine the geometrical differences between the rectangles and the strip width W through linear combinations, using dual feasible functions, rectangle area or rectangle height (Boschetti and Montaletti, 2010; Bekrar and Kacem,

2009).

The continuous lower bound[3] L_c is calculated using the total area of all rectangles R ($r \in R$) and the strip width, computing in a linear time. Considering $d1_r$ and $d2_r$, respectively, as the tallest and smallest dimensions of the rectangle, the maximum dimension $d2_r$ between all rectangles R is defined as lower bound L_h. The area lower bound, L_0, is the maximum value found between L_c and L_h (Martello et al., 2003; Boschetti and Montaletti, 2010). Despite being a limited area lower bound, the continuous lower bound is the only one in the rectangular 2D-SPP that considers rectangle rotation. Therefore, this lower bound will be used as a reference value to calculate the response variable of the regression model.

With the values of the best solution found using the random weight local search, H^{ref}, and the area lower bound, the response variable can be described in a relative normalised format. The use of a single absolute variable, i.e. H^{ref}, is not recommended to answer the characteristics described by the predictors, because of its high variability even when problem instances have similar characteristics.

Thus, the use of area lower bound as a reference value (L_0) must be proposed to evaluate the quality of the solutions H^{ref} found. In our research, the gap[4] $gap^{ref}(L_0)$ is the normalised value to be used as a known response variable in the regression analysis.

4.5. Data mining approach

This section aims to incorporate the information about the problem in order to develop the regression analysis and predict the strip height. Firstly, the process of extracting characteristics of the problem and the dimension reduction using the principal component analysis are described in Section 4.5.1. A total of 19 predictors were selected to be used as explanatory variables in the regression analysis. The response variable is calculated by taking into consideration the $gap^{ref}(L_0)$ between the best solution of the random weight local search, H^{ref}, and the area lower bound L_0.

A data mining selection step was conducted in Section 4.5.2 with the aim of selecting the technique that best fits the Dataset problem instances. In Section 4.5.3, the regression model was fitted. Finally, in Section 4.5.4, the level of generalisation (given by the coefficient of determination R^2) of the regression model and an absolute predictions' accuracy (given by the root-mean-square-error $RMSE$) were validated using a new set of 6,000 problem instances generated using the 2DCPackGen (Silva et al., 2014) problem generator and the set of the most used non-zero-waste benchmark problem instances from the literature.

4.5.1 Problem characteristics and dimension reduction

The study of the most important 2D-SPP characteristics aims to find combinations between the rectangles and the strip size and shape characteristics (Hall and Posner, 2007; Smith-Miles and Lopes, 2012). The main idea was to transform all the characteristics into a limited number of quantitative measures, named as descriptive variables.

[3] $L_c = (\sum_{r \in R} d1_r \cdot d2_r)/W$.
[4] $gap^{ref}(L_0) = (H^{ref} - L_0)/L_0$.

Since the number of descriptive variables can be very high, a Principal Component Analysis (PCA) was proposed in order to reduce the problem complexity and to facilitate a regression analysis interpretation. In addition, the reduction of descriptive variables in components helps to better understand the relation between the predictions for each problem instance and the problem characteristics.

PCA finds patterns in high dimension data, retaining most of the original information of each descriptive variable. The aim is to maximise the variance explained by each uncorrelated component extracted, boosting a wide representation of the different problem characteristics (López-Camacho and Terashima-Marín, 2013).

A recent work using PCA for the 2D-SPP considering the 1,270 benchmark problem instances most used in the literature was proposed in Neuenfeldt Júnior et al. (2017). A total of 56 descriptive variables were defined and divided into six groups (*Area*, *Perimeter*, *Dimensions*, *Widthdimensions*, *Proportions*, and *Other*) according to the similarity of each descriptive variable within the group.

The descriptive variables were defined based on the parameters and concepts found in the problem generators that have been proposed over the years in the literature, not only for the 2D-SPP, but also for other cutting and packing problems (Wang and Valenzela, 2001; Bortfeldt and Gehring, 2001; Berkey and Wang, 1987; Silva et al., 2014; Leung et al., 2011; Beasley, 1985b; Ferreira and Oliveira, 2005; Hall and Posner, 2001).

In this work, the methodology proposed in Neuenfeldt Júnior et al. (2017) will be used. The same 56 variables will be used to characterise the 30,000 problem instances generated for Dataset.

As suggested in Neuenfeldt Júnior et al. (2017), the PCA was individually applied to the *Area*, *Perimeter*, *Dimensions*, *Widthdimensions*, and *Proportions* groups, and two components with eigenvalues greater than or equal to one were extracted for each group, resulting in a total of 10 components. The division into five groups maintains the main characteristics of the descriptive variables and ensures a high mean variance explained for all groups. Table 4.1 shows the components extracted by each group. Additionally, in Appendix 4.B we introduce a representation of the explanatory variables' projections along the extracted components for groups *Area*, *Perimeter*, *Dimensions*, *Widthdimensions*, and *Proportions*.

Group	Component	Eigenvalue	Variance (%)	Cumulative (%)
Area	*areacomp*	7.05	78.31	
	areastats	1.18	13.14	91.45
Perimeter	*perimcomp*	5.80	64.39	
	perimstats	2.25	24.94	89.33
Dimensions	*dimcomp*	6.60	66.03	
	dimstats	2.52	25.24	91.27
Width dimensions	*widthdimcomp*	6.24	69.33	
	widthdimstats	1.66	18.45	87.78
Proportions	*propcomp*	5.45	54.52	
	propstats	3.73	37.25	91.77

Table 4.1 – Variance explained by each component for all groups.

Specifically, the group *Area* has the higher difference between the components' eigenvalues, while in group *Proportions*, the components have the most similar eigenvalues, being completely influenced by different descriptive variables. The group *Other* is atypical and does not present significant similarity between descriptive variables (small correlation coefficients). The PCA does not effectively reduce the group *Other* to a small number of components. Besides, descriptive variables with low correlations originate the extraction of components that are not intuitive and of difficult interpretation. For the group *Other*, the cumulative variance for the four components is only moderate (81.55%), and it is not comparable with the remaining groups.

For factor loading projections, all groups present a similar behaviour. Composition and ratio descriptive variables have a more significant influence on the first components (*areacomp*, *perimcomp*, *dimcomp*, *widthdimcomp* and *propcomp*). Instead, most classical statistical descriptive variables (e.g. mean, median) have more influence on the second components (*areastats*, *perimstats*, *dimstats*, *widthdimstats* and *propstats*). The behaviour observed in the PCA extraction is not only dependent on the groups, but also on the nature of the descriptive variables. This fact is more visible in group *Proportions*, in which none of the descriptive variables have a high influence for both components at the same time. The standard deviation descriptive variable (e.g. *areastdev*) is a special case where the impact on groups *Area*, *Dimensions* and *Widthdimensions* is higher for the first components, and on groups *Perimeter* and *Widthdimensions*, the factor loading affects the second components.

Five descriptive variables (*areamean*, *areastdev*, *perimstdev*, *dimmean* and *widthdimmean*) have a complex structure, influencing both components in groups *Area*, *Perimeter*, *Dimensions* and *Widthdimensions*. As an example, the *perimstdev* factor loading is equal to 0.6 for *perimcomp* and 0.7 for *perimstats*. By means of variance, the *perimstdev* descriptive variable can accurately explain both components, even without a high factor loading. Furthermore, *perimstdev* can influence a third component with an eigenvalue lower than one, which will not be extracted by the PCA. Finally, none of the groups has a non-influential descriptive variable for the extracted components. In at least one of the components, the factor loadings of the descriptive variables are always higher than or equal to 0.4.

For group *Other*, the descriptive variables' dispersion is high, and the components are explained by a small number of descriptive variables with high factor loading. This dispersion also shows the need for more components to provide a better representation of the group characteristics.

Four components were extracted with an eigenvalue higher than or equal to 1. If new components were added, the cumulative variance explained would increase. However, PCA would not significantly reduce the dimensions of the group *Other*.

Regarding the last analysis, the components proposed by the PCA to the group *Other* are not considered in our study. For the regression analysis, the 19 predictors are composed of ten components provided by the PCA applied in the groups of descriptive variables *Area*, *Perimeter*, *Dimensions*, *Widthdimensions* and *Proportions*, and directly the nine descriptive variables that defines the group *Other* (*n*, *heterog*, *heterognt*, *coefficient*, *difcoefficient*, *objdimratio*, *itdimratio*, *maxcoefficient* and *mincoefficient*).

4.5.2 Technique selection

The data mining technique that better adjusts the Dataset was selected taking into consideration three phases. Firstly, Friedman non-parametric tests were conducted based on Demšar (2006) procedures. Secondly, if the null-hypothesis was rejected, a post-test using the critical difference, *CD*, verified which technique could be distinguished from one another. Finally, in the group of non-significant distinguished techniques, the higher coefficient of determination, R^2, was used to select the data mining technique.

In R^2, the results for the $gap^{ref}(L_0)$ were compared with the predicted $\hat{gap}^{ref}(L_0)$, obtained using different data mining techniques. If the predictions have a good fit with the calculated $gap^{ref}(L_0)$, then the R^2 is high. In contrast, lower R^2 indicates that the $gap^{ref}(L_0)$ variation is not well explained by the predictors. This low level of adjustment can be explained by the lack of predictors capable of accurately explaining the phenomena, resulting in an incomplete representation of the main problem characteristics. Another reason is the intrinsic data mining technique incapacity of fitting a regression model capable of precisely predicting the $\hat{gap}^{ref}(L_0)$.

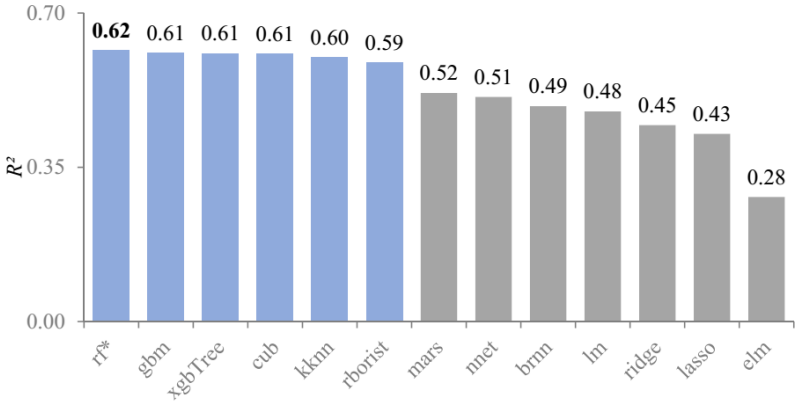
A total of 13 parametric and non-parametric techniques were previously selected for testing: two Random forest (*rf** and *rborist*); the Stochastic and Extreme gradient boosting (*gbm* and *xgbTree*); the Extreme learning machine (*elm*), the Back-propagation and Bayesian regularised neural networks (*nnet* and *brnn*); the Ridge regression (*ridge*); the Least absolute shrinkage and selection operator (*lasso*); the Linear and Multivariate adaptive regression models (*lm* and *mars*); the k-nearest neighbours (*kknn*); and the Cubist (*cub*). All techniques are capable of predicting the strip height based on supervised regression analysis concepts, tested using the software *RStudio*, specifically the function "train" in package "caret" (Kuhn, 2008). To avoid interfering in the results, all techniques were tested without the definition of any specific input parameter.

From the 30,000 Dataset problem instances, a small sample of 1,000 problem instances was selected (100 of each class). To proceed with Friedman's hypothesis tests, this sample of 1,000 problem instances was randomly divided into five subsamples, each with two different types of problem instances: training and test. Regression models were fitted for each technique using the five training subsamples described, and Figure 4.5a shows the mean R^2 values from the difference between $gap^{ref}(L_0)$ and $\hat{gap}^{ref}(L_0)$ of test subsamples.

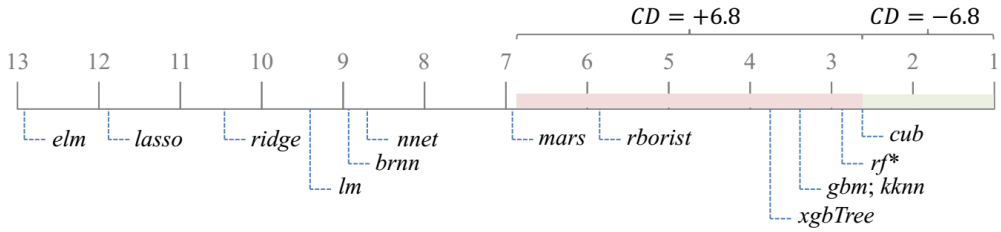
When the calculation of the performance of the techniques in each subsample shows a significance level equal to 0.05, the null-hypothesis is rejected, and there is a significant difference in the R^2 performance between the data mining techniques.

The Nemenyi (Demšar, 2006) post hoc test uses the critical difference, *CD*, to verify which techniques are significantly similar. Figure 4.5b shows in the Nemenyi scale the mean positioning performance of all techniques compared with the calculated *CD*. Using as a reference value the technique with the best positioning performance (*cubist*), techniques are significantly equal if the position performance is located between +6.8 and -6.8.

The first six techniques with the highest R^2 (*rf**, *gbm*, *xgbTree*, *cub*, *kknn*, and *rborist*) do not have significant differences. Using the five subsamples adopted, none of these techniques stood out as having superior performance. As an alternative, the mean coefficient of determination was defined as an alternative selection measure. Therefore, with R^2 equal



(a) Coefficient of determination (R^2).



(b) Critical difference (CD) measure.

Figure 4.5 – For each data mining technique.

to 0.62, the Random Forest (RF) was selected to perform the height prediction, as it is the data mining technique with the best mean R^2 from all available options.

4.5.3 Regression analysis

The RF is a supervised and non-parametric ensemble data mining technique to fit classification or regression models, using series of individual random decision tree structures. Each decision tree has nodes and arcs. The nodes are labelled by input features (predictors), organised in levels. The arcs coming from the nodes are labelled by possible values obtained for the response variable, the gap $gap^{ref}(L_0)$.

The main objective is to select in each node the predictor that produces the prediction $\hat{gap}^{ref}(L_0)$ most similar to the calculated $gap^{ref}(L_0)$. For the regression model, the difference between the mean-square-error before and after the addition of a specific predictor in the node is the measure used to select the best predictor. The decision tree grows until a stopping condition is reached (e.g. maximum number of levels, or root-mean-square-error lower than or equal to a specific value). Each decision tree has the same weight, and the final RF regression model are defined by the mean predictions $\hat{gap}^{ref}(L_0)$ obtained by the construction of all individual random decision trees developed.

In this research, the RF regression analysis was fitted using a concept called “bagging”, to improve the regression model level of generalisation. The 30,000 Dataset problem instances were randomly divided into two parts: training (24,000) and testing (6,000). A

total of 150 trees were considered, with all predictors evaluated at each split. The hold-out method was applied to create trees with 66% of training problem instances ("bag" observations), defined in a random process. The remaining 33% of training problem instances excluded from this tree, "out-of-bag" (OOB) observations, were used to predict the mean-square-error between $gap^{ref}(L_0)$ and $\hat{gap}^{ref}(L_0)$ in each tree. The sampling with replacement option allows any problem instance to appear multiple times within the "bag" or "out-of-bag" groups.

To improve the quality of regression models, the 5-fold cross-validation process was conducted, folding the training sample into five different parts. The problem instances were randomly inserted into only one fold. A total of ten regression models were generated, two for each fold, and the regression model with the lowest RMSE between $gap^{ref}(L_0)$ and $\hat{gap}^{ref}(L_0)$ was selected as the RF regression model.

To verify the predictors' performance, we adopt the proportional *IncNodePurity* relevance index. The *IncNodePurity* measures the difference between the mean-square-error (for "out-of-bag problem instances) between $\hat{gap}^{ref}(L_0)$ and $gap^{ref}(L_0)$ before and after the tree split (node). The most relevant predictors are more efficient in reducing the mean-square-error and have higher proportional *IncNodePurity* relevance values. The average value obtained for all RF trees is calculated for each predictor.

The results of the proportional *IncNodePurity* relevance index can vary from one RF regression model to another. Three main parameters affect the performance: (1) the number of trees; (2) the total number of predictors tested in each node; and (3) the random process of developing the decision trees. In this research, we fixed the first two parameters, the number of trees (150) and the total number of predictors tested in each node (19). For the third parameter, the random processes of developing the decision trees, the RF was run ten times, developing random and different trees for each RF. Figure 4.6 shows the mean proportional value *IncNodePurity* for the 19 predictors after running the RF ten times.

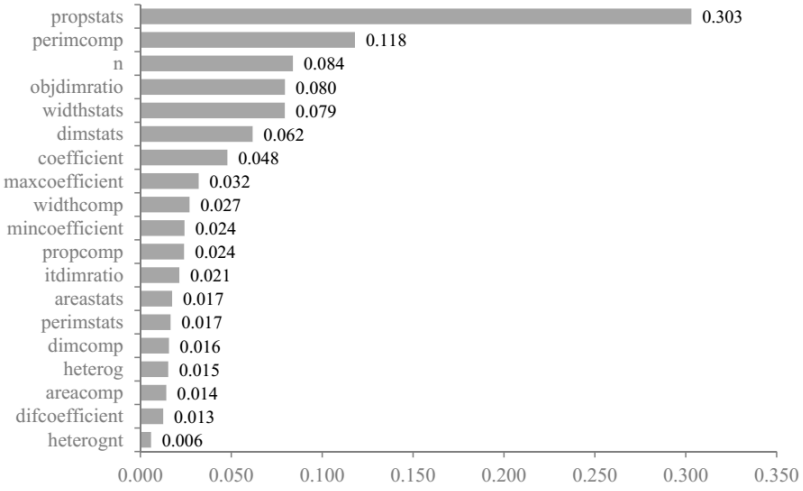


Figure 4.6 – The proportional *IncNodePurity* predictors' importance.

The *propstats* and *objdimratio* generated the largest mean-square-error reductions before and after split the tree, which is sufficient to consider that these predictors explain more

accurately most of the problem characteristics. In addition, the selection of *propstats* and *objdimratio* as the most important predictors is directly related to the manner in which the RF processes all input data to fit the regression model. In general, none of the six groups of predictors is predominant, with one predictor of each group included in the five highest proportional *IncNodePurity* values.

The regression model generated using the RF for the 24,000 Dataset training problem instances was applied to obtain the predictions. Figure 4.7 shows the dispersion graph obtained for the calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ response variable for the 6,000 Dataset test problem instances. The coefficient of determination, $R^2 = 0.68$, and the predictions' accuracy $RMSE = 0.04$ are on a good level. Additionally, in Appendix 4.C we introduce the projections of the dispersion between $gap^{ref}(L_0)$ and $\hat{gap}^{ref}(L_0)$ for for the remaining 12 data mining techniques not explored in details in this research.

For almost all problem instances (5,865 out of 6,000), the calculated $gap^{ref}(L_0)$ is in the range between 0.02 and 0.3. The regression model performs more accurate predictions ($RMSE = 0.03$) for problem instances located within this range. Conversely, for problem instances with calculated $gap^{ref}(L_0)$ higher than 0.3 (97 out of 6,000) or lower than 0.02 (38 out of 6,000), the predictions are not accurate ($RMSE = 0.16$).

An important characteristic of the problem with impact on the results obtained for the predictions' accuracy is the shape of the strip. Among the different formats, two are described in detail: the narrow strip and the square strip. The narrow strip problem instances have a strip width W that is much smaller than the calculated height H^{ref}, where the strip aspect ratio[5] is equal to or higher than 100. This specific case was observed in 304 test problem instances, and the predictions' accuracy ($RMSE = 0.10$) are not well fitted for almost all problem instances. For the square strip instances in which the strip width dimension W is almost equal to the calculated height H^{ref}, the strip aspect ratio is between 0.8 and 1.2. Considering all 233 square strip problem instances, the predictions have a good fit quality ($RMSE = 0.02$). For example, the problem instance $pt11\_21\_39$ has a narrow strip and has the worst prediction accuracy of all test problem instances ($RMSE = 0.54$). On the other hand, the problem instance $pt7\_38\_1$ has a square strip and the prediction accuracy is almost equal to zero ($RMSE = 0.004$).

In the case of the ten problem instance classes, the predictions' behaviours are very similar, because most of the calculated $gap^{ref}(L_0)$ are located in a very dense space between $gap^{ref}(L_0) = 0.02$ and $gap^{ref}(L_0) = 0.3$. This fact is directly related with the problem instance generation, which was based on the combination of four parameters (the rectangle size, the strip width, the number of different rectangle types, and the rectangle type demand), producing similar problem instances in each class.

4.5.4 Additional validation

A completely different problem instance dataset (named as Validation dataset) was used in the regression model fitted in Section 4.5.3 with the goal of verifying the level of generalisation R^2 between $\hat{gap}^{ref}(L_0)$ and $gap^{ref}(L_0)$ and the predictions' accuracy $RMSE$.

[5] H^{ref}/W.

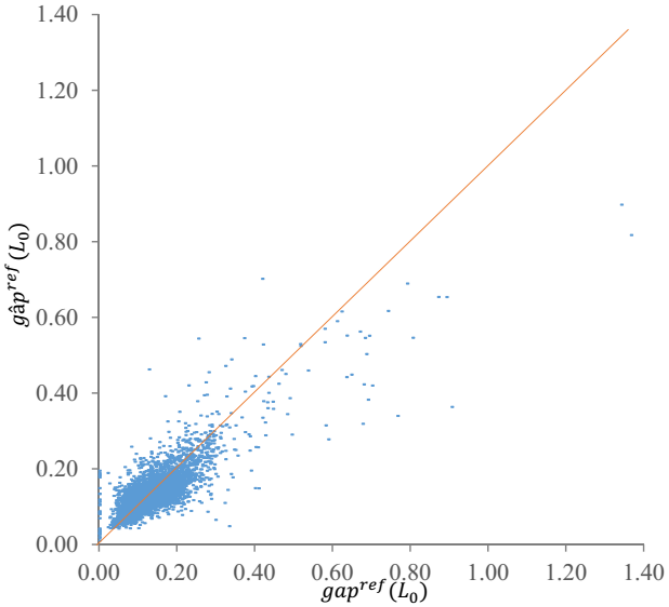


Figure 4.7 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the 6,000 Dataset test problem instances.

A total of 6,986 Validation dataset problem instances from two different datasets were used, in which 6,000 problem instances were additionally generated using the 2DCPack-Gen problem generator (Silva et al., 2014). The uniform distribution was adopted for all problem generator parameters, allowing the generation of problem instances different from the 30,000 Dataset problem instances used to generate the regression model. This prevents a specific non-uniform distribution from interfering in the generation of different problem instances. The remaining 986 problem instances were obtained from non-zero-waste benchmark problem instances of the 2D-SPP, considering: 500 *bwmv* (Berkey and Wang, 1987), 360 *AH* (Bortfeldt, 2006), 72 *nice* and *path* (Wang and Valenzela, 2001), 16 *zdf* (Leung and Zhang, 2011), 13 *gcut* (Beasley, 1985b), 12 *ngcut* (Beasley, 1985a), 10 *beng* (Bengtsson, 1982), and 3 *cgcut* (Christofides and Whitlock, 1977). In 254 benchmark problem instances, the rectangles and the strip dimensions exceed the maximum dimension parameters defined in the 30,000 Dataset problem instances generated to develop the regression model. To maintain the minimum and maximum parameter standards, the rectangle and strip values were divided by 10 or by 100.

The next step was to find predictions for these new 6,986 Validation dataset problem instances, to verify the level of adjustment of the regression model fitted using the Dataset. As expected, the level of generalisation ($R^2 = 0.55$) and the predictions' accuracy ($RMSE = 0.08$) are lower than the results verified in the 6,000 Dataset test problem instances, and the dispersion of results is higher when compared with the results obtained in Figure 4.7.

These differences are due to the high degree of similarity between the test and training problem instances of the regression model, both generated simultaneously using same non-uniform distributions. Instead, benchmark problem instances were generated using different problem generators, each one with specific parameters and characteristics. For

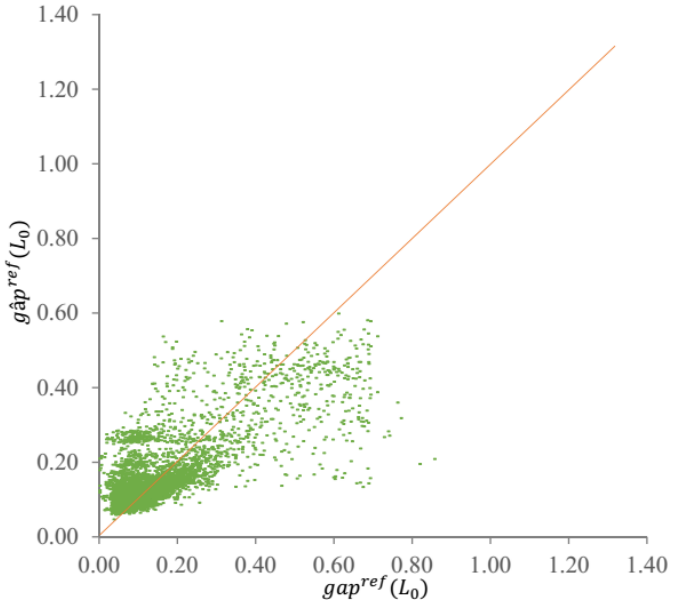


Figure 4.8 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Validation dataset, composed by 6,000 new problem instances generated and 986 non-zero-waste benchmark problem instances.

the 6,000 problem instances additionally generated, the use of uniform distributions was proposed, thus differentiating their behaviour from Dataset.

As verified for the 6,000 Dataset test problem instances, in the Validation dataset problem instances (6,474 out of 6,986) where the $gap^{ref}(L_0)$ is less than 0.3, the predictions are very accurate ($RMSE = 0.06$). This happens because of the high density of problems instances with similar $gap^{ref}(L_0)$ used to fit the regression model. For the Validation dataset problem instances (512 out of 6,986) with calculated $gap^{ref}(L_0)$ higher than 0.3, the predictions' accuracy is not well adjusted ($RMSE = 0.19$).

4.6. Assessing framework performance

The main objective of this research is to propose a data mining-based framework to assess the quality of heuristic solutions for the 2D-SPP with 90 degree rotations. Predictions were obtained by a regression model fitted with the Random forest data mining technique, providing a quality measure to heuristic solutions and a stopping criterion to be used in local search algorithms in order to avoid long computational times.

To conduct the data mining approach presented in Section 4.5, a deeper knowledge about the problem was acquired through a 2D-SPP literature review (Oliveira et al., 2016; Neuenfeldt Júnior et al., 2017). After characterising the problem, a total of 19 predictors were defined, and one known response variable was calculated using the gap $gap^{ref}(L_0)$ between the lowest solution H^{ref} found using the random weight local search and the area lower bound L_0.

Random forest was selected as best data mining technique option, based on the coefficient of determination, R^2, measure and Friedman's hypothesis tests. The regression analysis was developed in order to find predictions $\hat{gap}^{ref}(L_0)$ of the response variable. Finally, the level of generalisation of the fitted regression model and the predictions' accuracy were verified by calculating the predictions $\hat{gap}^{ref}(L_0)$ for 6,000 Dataset test problem instances and 6,000 Validation dataset problem instances.

To assess the framework performance, in this section the aim is to test the efficiency of the regression model fitted in order to predict the strip height in the 2D-SPP. Section 4.6.1 shows the conversion of $\hat{gap}^{ref}(L_0)$ in the strip height predictions[6] $\hat{H}^{ref}$, and the use of $\hat{H}^{ref}$ as a stopping criterion in the exchange procedure for three different local search algorithms (random weight, completely random, and dynamic random). In Section 4.6.2, a prediction performance analysis was conducted by varying the prediction value, $\hat{H}^{ref}$, in accordance with a multiplier γ in order to evaluate the robustness of the predictions according to the behaviour of the quality of the solutions of the dynamic random local search.

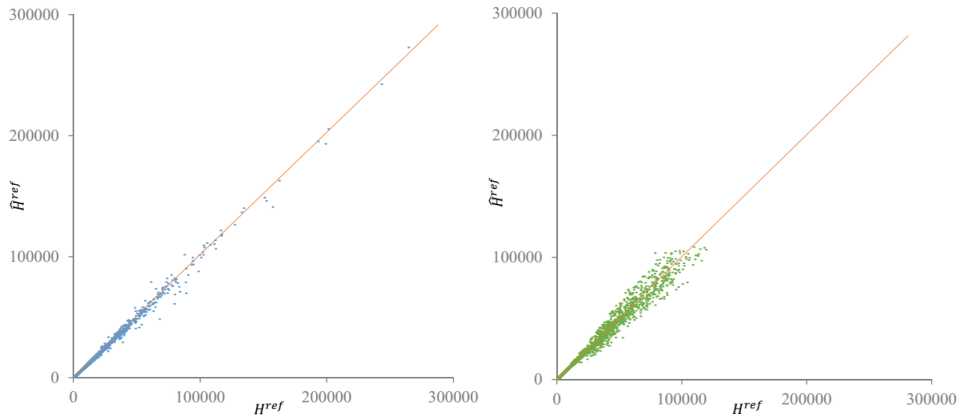
4.6.1 Framework usage

In this section, the use of the proposed data mining-based framework was proposed in two steps. Firstly, the regression model fitted in Section 4.5 is used to predict the strip height for the 6,000 Dataset test problem instances of Section 4.5.3 and 6,986 Validation dataset problem instances of Section 4.5.4. Finally, the height predictions' performance as stopping criterion was evaluated in three naive local search algorithms.

To test the efficiency of the regression model fitted in order to predict the strip height, Figure 4.9 shows the results of the comparison between calculated H^{ref} and predicted $\hat{H}^{ref}$. For the 6,000 Dataset test problem instances (Figure 4.9a), almost all predictions $\hat{H}^{ref}$ are very similar to the values H^{ref} obtained by the random weight local search. The level of generalisation is very high ($R^2 = 0.99$) and the measured predictions' accuracy is low ($RMSE = 955$). This high level of generalisation is closely related with L_0, which is a fixed reference value used in both H^{ref} and $\hat{H}^{ref}$. In addition, for almost all 6,000 Dataset test problem instances, $\hat{gap}^{ref}(L_0)$ and $gap^{ref}(L_0)$ have small $RMSE$ values, resulting in small differences between the predicted strip height and the solution found using the random weight local search.

For the 6,986 Validation dataset problem instances' predictions (Figure 4.9b), the level of generalisation ($R^2 = 0.98$) is also very high, and the accuracy of the predictions can be considered of good quality ($RMSE = 2,345$), mainly for the 2,727 problem instances ($RMSE = 116$) with lower strip heights, where the calculated H^{ref} is less than 5,000. The results of the regression model's lack of capability in accurately predicting the strip height for Validation dataset in comparison with Dataset can be seen mainly in problem instances with narrow strip, strips with high heights and strips with small widths. However, the adjustment obtained using the Validation dataset is acceptable and, as expected in this context, it does not significantly affect the good adjustment obtained for the predicted strip height $\hat{H}^{ref}$.

[6] $\hat{H}^{ref} = L_0(1 + \hat{gap}^{ref}(L_0))$.



(a) 6,000 Dataset test problem instances. (b) 6,986 Validation problem instances.

Figure 4.9 – Calculated H^{ref} and predicted $\hat{H}^{ref}$.

The regression model can be used to predict reference values for the local search algorithms exchange procedure, which can be used as the stopping criterion to avoid long computational times. Figure 4.10 describes an evaluation of the predictions' performance as a stopping criterion for 1,000 problem instances randomly provided by the Validation dataset (Section 4.5.4). The x-axis represents the 100 test iterations, and in the y-axis, the cumulative frequency containing the total number of problem instances that reached the predicted $\hat{H}^{ref}$ is presented.

Three naive local search algorithms were adopted to improve the initial solution found with the BLF constructive heuristic with fixed sequences (by area, perimeter, width or height dimensions). Firstly, in the complete random local search, the sequence of rectangles is defined completely randomly and independently. Secondly, the random weight local search was used with the same characteristics defined to calculate the strip height of each problem instance for the $gap^{ref}(L_0)$ (response variable) of the regression model, as described in Section 4.4.2.

Finally, the dynamic random local search concepts were proposed, randomly change the current rectangles' sequence position at each iteration by 5%. The initial solution is the best solution found that considers four fixed sequences, sorted by non-decreasing area, perimeter, maximum rectangle dimension, $d1_r$, and minimum rectangle dimension $d2_r$. The dynamic random local search explores the solution space applying local and small changes. The main advantage is maintaining significant information about the current best solution, which does not occur for the completely random and random weight local search algorithms, where the solution space is explored in a more general manner.

The adoption of more than one local search strategy was proposed in order to verify the behaviour of the predictions as stopping criterion for different situations. As expected, in all local search algorithms, most of the problem instances reached the predicted $\hat{H}^{ref}$. The most effective criterion to stop the local search was to sort the rectangles by decreasing maximum and minimum dimensions. The dynamic random local search reached a higher number of heights, equal to or less than the predicted $\hat{H}^{ref}$, stopping a total of 630 prob-

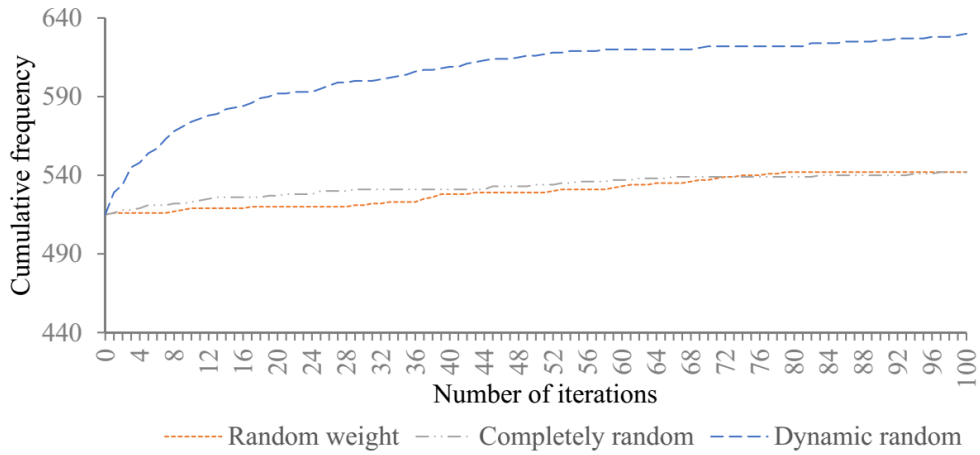


Figure 4.10 – Cumulative frequency of stopping process for the 1,000 problem instances selected from Validation dataset.

lem instances out of 1,000, while the completely random and random weight local search algorithms have almost the same number of stops after 100 iterations.

The use of non-accurate predictions can have two different effects on the behaviour of the stopping criterion. Overly conservative predictions stop the local search early and the optimal solution is never reached. Conversely, overly optimistic predictions have smaller values than the calculated heuristic solutions, which may even be lower than the optimal solution. When using this criterion, the iterations will never stop.

4.6.2 Prediction performance analysis

The main objective of this section is to verify if small perturbations in the strip height predictions are capable of substantially changing the stopping process behaviour of the local search, considering the relation between the quality of the solutions obtained with the dynamic random local search, the number of iterations and variations in the predictions.

The predictions were not calculated by parametric models, which do not allow the use of metrics as the mean-square-error of the standard deviation as a means of verifying the predictions' robustness. The solutions obtained by the dynamic random local search, using the $\hat{H}^{ref}$ as stopping criterion, are represented by $H^{drls}(\hat{H}^{ref})$. The quality of these solutions is measured by the gap $gap^{drls}(\hat{H}^{ref})$[7] $gap^{drls}(\hat{H}^{ref})$ between the predictions $\hat{H}^{ref}$ and the solutions $H^{drls}(\hat{H}^{ref})$. To evaluate the robustness of the predictions, small changes in the predictions were defined. For each problem instance, a γ value is multiplied by the prediction $\hat{H}^{ref}$ and is used as the reference value to calculate the quality of the solution, $gap^{drls}(\hat{H}^{ref})$, provided by the dynamic random local search.

Figure 4.11 shows the relation between the increase in the number of iterations (x-axis) and the reduction of the mean gap (y-axis) during the dynamic random local search algorithm, for the 1,000 problem instances considered in Section 4.6.1, with the four different γ values adopted ($\gamma = 0.90$, $\gamma = 0.98$, $\gamma = 1.02$, and $\gamma = 1.10$). The study of the quality of

[7] $gap^{drls}(\hat{H}^{ref}) = (H^{drls}(\hat{H}^{ref}) - \hat{H}^{ref})/\hat{H}^{ref}$.

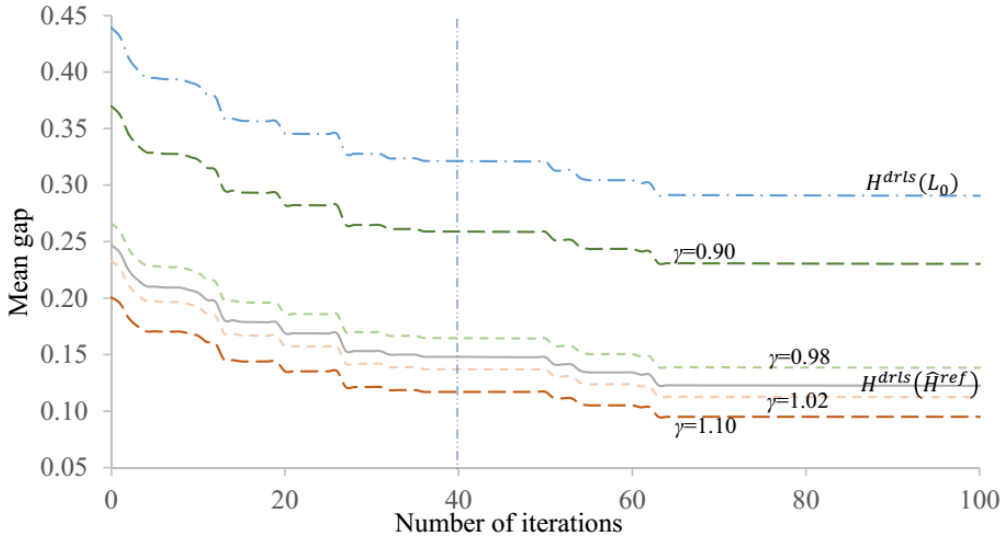


Figure 4.11 – Trade-off between the number of iterations and the quality of the solutions for the 1,000 problem instances selected from Validation dataset, using different variations of the predictors and the area lower bound as stopping criterion. When the prediction $\hat{H}^{ref}$ is adopted as a reference value for the local search, the gap is given by $gap^{drls}(\hat{H}^{ref})$. When the area lower bound is adopted as reference value, the gap is given by $gap^{drls}(L_0)$.

the solutions $H^{drls}(\hat{H}^{ref})$ found using the dynamic random local search and the area lower bound as stopping criterion is also proposed in Figure 4.11. It is measured using the gap[8] $gap^{drls}(\hat{H}^{ref})$.

All curves present a similar behaviour, independently of the γ value considered. For $\gamma = 0.90$ and $\gamma = 1.10$, a variation of 10% in the original prediction $\hat{H}^{ref}$ is considered, and, as expected, this has a high impact on the mean $gap^{drls}(\hat{H}^{ref})$, because with a high γ the local search reached the stopping criterion too early and a low γ presents optimistic predictions. Therefore, in the robustness analysis, $\gamma = 0.90$ and $\gamma = 1.10$ are not considered.

For $\gamma = 0.98$ and $\gamma = 1.02$, the mean $gap^{drls}(\hat{H}^{ref})$ is very similar for almost all problem instances during the local search. This means that the quality of the solutions obtained is not greatly affected by small variations of the predicted $\hat{H}^{ref}$, stressing that the predictions were robust.

The graphic in Figure 4.11 is also relevant to analyse how the reduction in the number of iterations affects the quality of the solution. Specifically, the mean gap $gap^{drls}(\hat{H}^{ref})$ considering prediction $\hat{H}^{ref}$ is 12% after 100 iterations. If the number of iterations is limited to 40, the mean gap $gap^{drls}(\hat{H}^{ref})$ would be 15%, which means that saving 60 iterations would have an impact of only 3% in the quality of the solution. Also, the graphic shows the inefficiency of L_0 when used as a stopping criterion, demonstrated by the high values found for the gap[9] $gap^{drls}(L_0)$ in comparison with the results obtained for all variations of $\hat{H}^{ref}$ predictions. The L_0 is too optimistic for almost all problem instances explored,

[8] $gap^{drls}(\hat{H}^{ref}) = (H^{drls}(\hat{H}^{ref}) - \hat{H}^{ref})/\hat{H}^{ref}$.
[9] $gap^{drls}(L_0) = (H^{drls}(L_0) - L_0)/L_0$.

thus requiring high computational processing times to provide the quality of the solutions $H^{drls}(L_0)$ found by the local search.

4.7. Conclusions

This research proposes a data mining-based framework capable of predicting a reference value to be used as stopping criterion in local search algorithms for the 2D-SPP, taking into consideration the main problem characteristics. The total height necessary to pack a set of small rectangular rectangles into a rectangular strip is the reference value used in the 2D-SPP. In addition, the predictions can provide a measure to verify the quality of solutions found by heuristics.

Data mining techniques were tested and the Random forest was statistically inferred as the best choice for developing the regression analysis, based on its ability to generalise the predictors for the normalized gap $gap^{ref}(L_0)$. Other data mining techniques, such as the Extreme gradient boosting, Cubist and k-nearest neighbours, also have significant generalisation ability. The input data was composed of three parts: the predictors to provide a numeric measure of the problem characteristics, the gap calculated using the area lower bound and the random weight local search, and 30,000 Dataset problem instances generated with 2DCPackGen (Silva et al., 2014) to represent different 2D-SPP variations found in real-world applications.

The 5-fold cross-validation in Section 4.5.3 was adopted to verify if the regression model has a good level of generalisation and if it accurately predicts the strip height for new problem instances. An additional validation proposed in Section 4.5.4 with the use of 6,000 problem instances was generated together with 986 non-zero-waste benchmark problem instances. Both 5-fold cross-validation and additional validation confirmed the good level of generalization of the regression model and the predictions' accuracy. The quality of the level of adjustment obtained was fundamental to conclude that the extraction of problem characteristics was well performed.

In the assessing performance section, it was verified that the strip height predictions are highly influenced by the solution obtained by the constructive heuristic, by local search algorithms and by the reference values (area lower bound or the predictions found with the regression model fitted) used to calculate the gap. Therefore, the behaviour of the quality of the solutions was proved to be robust even when small changes in the predictions are applied. In addition, it was confirmed that the area lower bound is an overly conservative stopping criterion for almost all 1,000 problem instances from Validation dataset used to test the predictions' robustness.

Widely used in literature over the years, the BLF was selected as the constructive heuristic due to the good cost-benefit ratio between its simplicity of implementation and the quality of the solutions found. Other constructive heuristics could have been used instead of BLF.

This research contributes to the identification of the characteristics that most affect constructive heuristics in solving the 2D-SPP. A better understanding of these characteristics brings new elements capable of shedding light on the design of new heuristics for

the 2D-SPP, mainly when the optimal solution is unknown, reducing the computational time required to verify the quality of the solution. Furthermore, this research was used to validate the dimension reduction methodology proposed in Neuenfeldt Júnior et al. (2017).

In this study, we demonstrate that the data mining-based framework proposed is consistent and can be applied to predict response variable values in other problems. For future research, our framework needs to be tested in other problems in order to verify the level of adjustment of the predictions obtained by the regression analysis in other contexts. Specifically in the 2D-SPP, it is important to apply the predictions using the regression model as a stopping criterion in different local search algorithms.

References

Almeida, R. D. and Steiner, M. T. A. (2016). Resolution of one-dimensional bin packing problems using augmented neural networks and minimum bin slack. *International Journal of Innovative Computing and Applications*, 7(4):214–224.

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

Alvarez-Valdés, R., Parreño, F., and Tamarit, J. M. (2008). Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, 35(4):1065–1083.

Baker, B. S., Coffman, Jr, E. G., and Rivest, R. L. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855.

Bastos, P., Lopes, I., and Pires, L. (2014). Application of data mining in a maintenance system for failure prediction. *Safety, Reliability and Risk Analysis: Beyond the Horizon*, 1(1):933–940.

Beasley, J. (1985a). Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society*, 36(4):297–306.

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

Bekrar, A. and Kacem, I. (2009). An exact method for the 2D guillotine strip packing problem. *Advances in Operations Research*, 2009(1).

Bengtsson, B.-E. (1982). Packing rectangular pieces: A heuristic approach. *The Computer Journal*, 25(3):353–357.

Berkey, J. O. and Wang, P. Y. (1987). Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38(5):423–429.

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3):814–837.

Bortfeldt, A. and Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161.

Boschetti, M. A. and Montaletti, L. (2010). An exact algorithm for the two-dimensional strip-packing problem. *Operations Research*, 58(6):1774–1791.

Brazdil, P. B., Soares, C., and Da Costa, J. P. (2003). Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277.

Cadima, J., Cerdeira, J. O., and Minhoto, M. (2004). Computational aspects of algorithms for variable selection in the context of principal components. *Computational Statistics & Data Analysis*, 47(2):225–236.

Chazelle, B. (1983). The bottomn-left bin-packing heuristic: An efficient implementation. *IEEE Transactions on Computers*, 100(8):697–707.

Christofides, N. and Whitlock, C. (1977). An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30.

Fekete, S. P. and Schepers, J. (2001). New classes of fast lower bounds for bin packing problems. *Mathematical programming*, 91(1):11–31.

Ferreira, E. P. and Oliveira, J. F. (2005). A note on Fekete and Schepers' algorithm for the non-guillotinable two-dimensional packing problem. Technical report, FEUP.

Gupta, A. K. and Nadarajah, S. (2004). *Handbook of beta distribution and its applications*. CRC press.

Hall, N. G. and Posner, M. E. (2001). Generating experimental data for computational testing with machine scheduling applications. *Operations Research*, 49(6):854–865.

Hall, N. G. and Posner, M. E. (2007). Performance prediction and preselection for optimization and heuristic solution procedures. *Operations Research*, 55(4):703–716.

Hopper, E. and Turton, B. C. H. (2001a). An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57.

Hopper, E. and Turton, B. C. H. (2001b). A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4):257–300.

Kuhn, M. (2008). Caret package. *Journal of Statistical Software*, 28(5):1–26.

Kumar, S. A. and Vijayalakshmi, M. (2011). Efficiency of decision trees in predicting student's academic performance. In *First International Conference on Computer Science, Engineering and Applications*, volume 2, pages 335–343.

Leung, S. C. H. and Zhang, D. (2011). A fast layer-based heuristic for non-guillotine strip packing. *Expert Systems with Applications*, 38(10):13032–13042.

Leung, S. C. H., Zhang, D., and Sim, K. M. (2011). A two-stage intelligent search algorithm for the two-dimensional strip packing problem. *European Journal of Operational Research*, 215(1):57–69.

Lodi, A., Martello, S., and Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252.

López-Camacho, E. and Terashima-Marín, H. (2013). Evolving feature selection for characterizing and solving the 1D and 2D bin packing problem. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2094–2101. IEEE.

López-Camacho, E., Terashima-Marín, H., Ochoa, G., and Conant-Pablos, S. E. (2013). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics*, 145(2):488–499.

López-Camacho, E., Terashima-Marín, H., and Ross, P. (2010). Defining a problem-state representation with data mining within a hyper-heuristic model which solves 2D irregular bin packing problems. In *Ibero-American Conference on Artificial Intelligence*, pages 204–213. Springer.

López-Camacho, E., Terashima-Marin, H., Ross, P., and Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889.

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

Neuenfeldt Júnior, A., Silva, E., Miguel Gomes, A., and Oliveira, J. F. (2017). The two-dimensional strip packing problem: What matters? *Operational Research, Springer Proceedings in Mathematics & Statistics (Accepted for publication)*, 223(11):1.

Neveu, B., Trombettoni, G., Araya, I., and Riff, M. C. (2008). A strip packing solving method using an incremental move based on maximal holes. *International Journal on Artificial Intelligence Tools*, 17(5):881–901.

Ntene, N. and van Vuuren, J. H. (2009). A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem. *Discrete Optimization*, 6(2):174–188.

Oliveira, J. F., Neuenfeldt Júnior, A., Silva, E., and Carravilla, M. A. (2016). A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, 36(2):197–226.

Perez, J., Frausto, J., Cruz, L., Fraire, H., and Santiago, E. (2004). A machine learning approach for modeling algorithm performance predictors. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 70–80. Springer.

Rice, J. R. (1976). The algorithm selection problem. *Advances in computers*, 15(1):65–118.

Riff, M. C., Bonnaire, X., and Neveu, B. (2009). A revision of recent approaches for two-dimensional strip-packing problems. *Engineering Applications of Artificial Intelligence*, 22(4-5):833–837.

Santoyo, A. M., Ortega, J. P., Vargas, D. R., and Reyes, L. C. (2015). Towards a characterization of difficult instances of the bin packing problem. *IEEE Latin America Transactions*, 13(7):2454–2462.

Silva, E., Oliveira, J. F., and Wäscher, G. (2014). 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, 237(3):846–856.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45(1):12–24.

Smith-Miles, K. and Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers & Operations Research*, 39(5):875–889.

Wang, P. Y. and Valenzela, C. L. (2001). Data set generation for rectangular placement problems. *European Journal of Operational Research*, 134(2):378–391.

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

Wei, L., Zhang, D., and Chen, Q. (2009). A least wasted first heuristic algorithm for the rectangular packing problem. *Computers and Operations Research*, 36(5):1608–1614.

Appendix 4.A Notation

- r: rectangles' index;
- n: number of rectangles in the problem instance;
- $d1_r$: maximum rectangle dimension (for 2D-SPP with 90 degree rotations);
- $d2_r$: minimum rectangle dimension (for 2D-SPP with 90 degree rotations);
- w_1, w_2, or w_3: rectangle width dimension (for 2D-SPP without 90 degree rotations);
- h_1, h_3, or h_4: rectangle height dimension (for 2D-SPP without 90 degree rotations);
- W: strip width;
- OS: optimal solution;
- L_h: height lower bound, measured by the maximum dimension $d2_r$ between all rectangles for the problem instance;
- L_c: continuous lower bound[10];
- L_0: area lower bound;
- H^{ref}: random weight local search strip height lowest solution;
- $\hat{H}^{ref}$: predicted strip height related with the area lower bound L_0;
- $H^{drls}(\hat{H}^{ref})$: dynamic random local search strip height solutions based on the predicted $\hat{H}^{ref}$;
- $H^{drls}(L_0)$: dynamic random local search strip height solutions based on the area lower bound L_0;
- $gap^{ref}(L_0)$: gap[11] between the solution H^{ref} and the area lower bound L_0;
- $\hat{gap}^{ref}(L_0)$: predicted $gap^{ref}(L_0)$ related with the area lower bound L_0;
- $gap^{drls}(\hat{H}^{ref})$: gap[12] between the solution $H^{drls}(\hat{H}^{ref})$ and the predicted $\hat{H}^{ref}$;
- $gap^{drls}(L_0)$: gap[13] between the solution $H^{drls}(\hat{H}^{ref})$ and the area lower bound L_0;
- γ: multiplier to vary the predictions $\hat{H}^{ref}$ (for the robustness verification);
- CD: critical difference to distinguish the data mining techniques (for technique selection);

[10] $L_c=(\sum_{r\in R} d1_r \cdot d2_r)/W$.
[11] $gap^{ref}(L_0) = (H^{ref} - L_0)/L_0$.
[12] $gap^{drls}(\hat{H}^{ref}) = (H^{drls}(\hat{H}^{ref}) - \hat{H}^{ref})/\hat{H}^{ref}$.
[13] $gap^{drls}(L_0) = (H^{drls}(L_0) - L_0)/L_0$.

- R^2: level of generalisation given by the coefficient of determination (for data mining approach);
- $RMSE$: predictions' accuracy given by the root-mean-square-error (for data mining approach).

Appendix 4.B Descriptive variables' projections

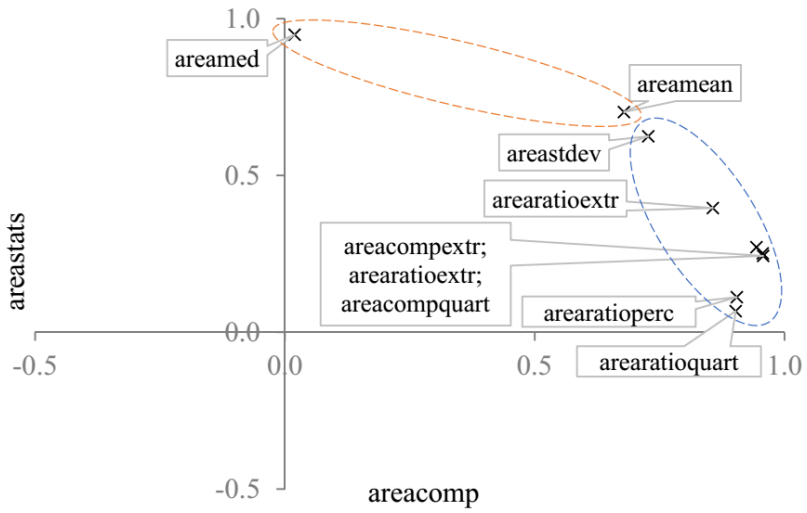


Figure 4.12 – Relations between variables and components for group *Area*.

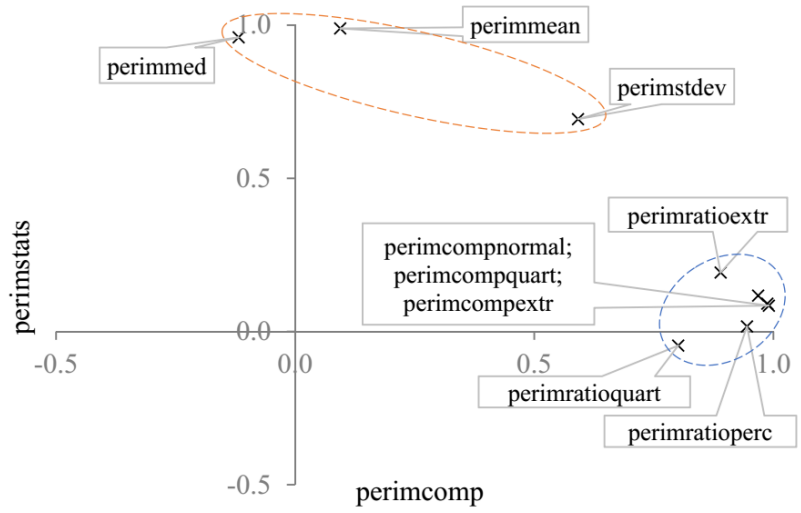


Figure 4.13 – Relations between variables and components for group *Perimeter*.

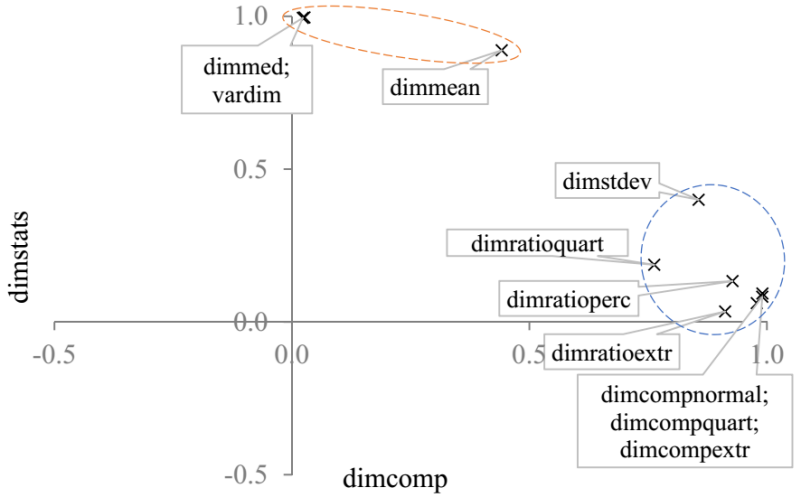


Figure 4.14 – Relations between variables and components for group *Dimensions*.

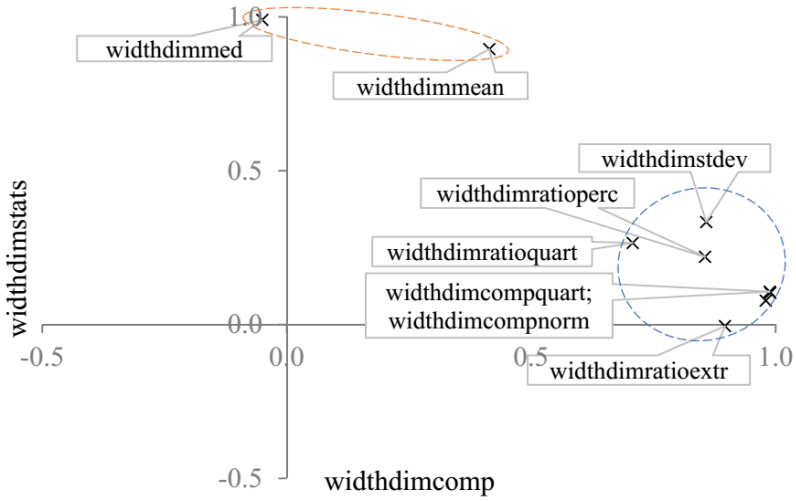


Figure 4.15 – Relations between variables and components for group *Widthdimensions*.

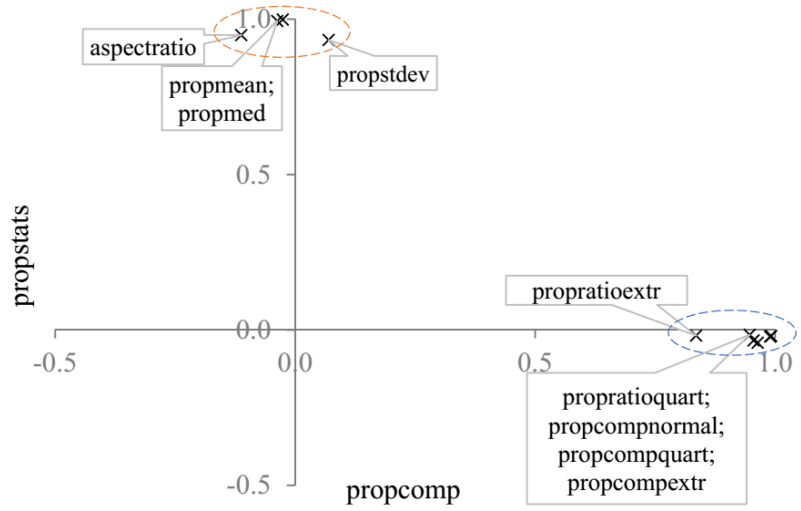


Figure 4.16 – Relations between explanatory variables and components for group *Proportions*.

Appendix 4.C Dispersion graphs between calculated and predicted gap

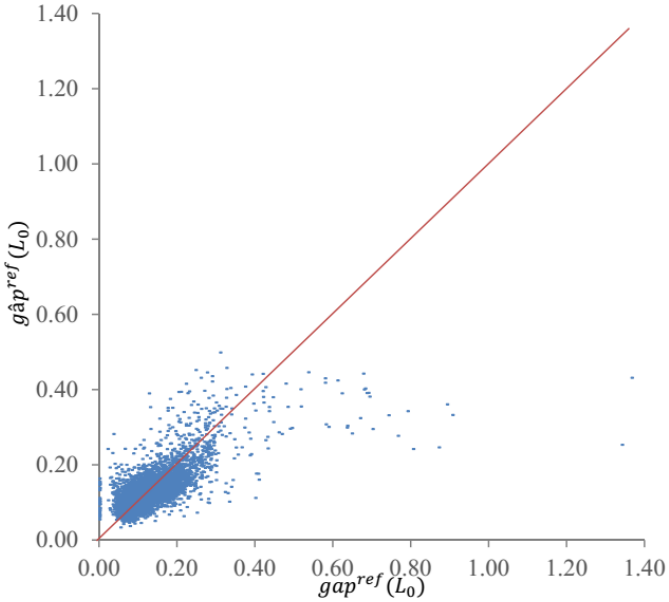


Figure 4.17 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Bayesian regularized neural net (*brnn*) ($R^2 = 0.52$ and $RMSE = 0.05$).

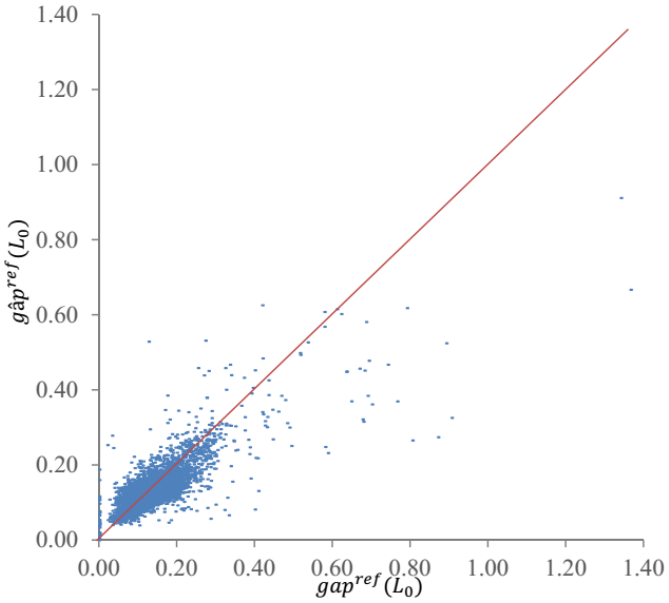


Figure 4.18 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the *Cubist* ($R^2 = 0.67$ and $RMSE = 0.04$).

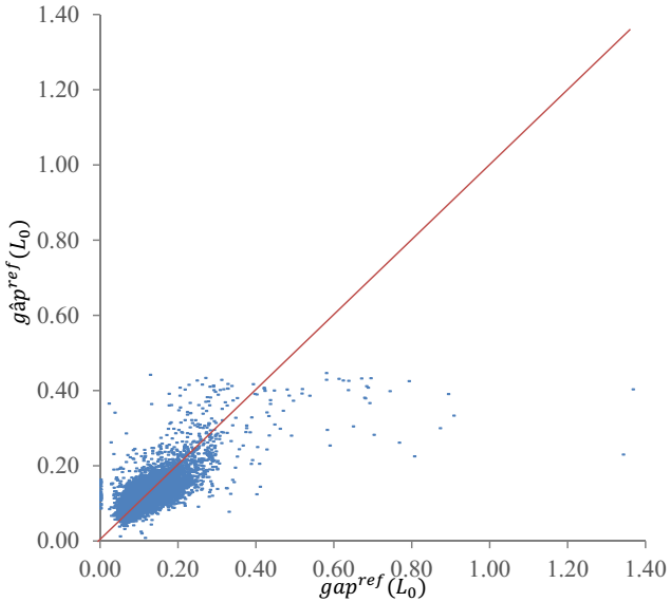


Figure 4.19 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Multivariate adaptive regression splines (*earth*) ($R^2 = 0.52$ and $RMSE = 0.05$).

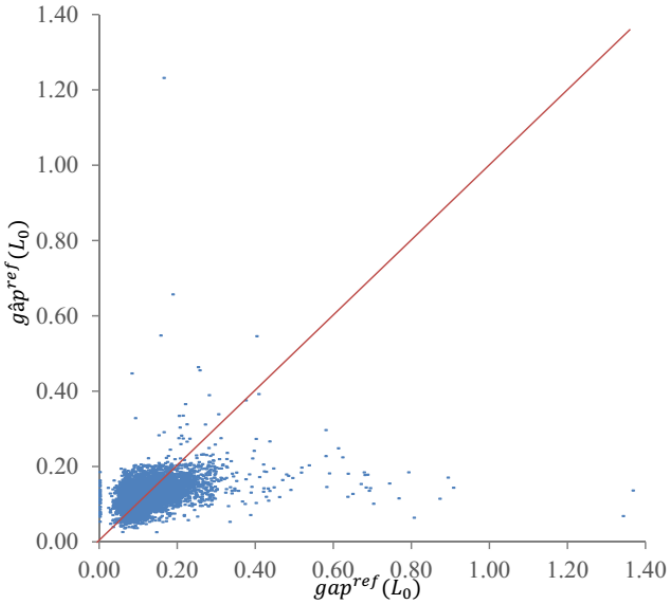


Figure 4.20 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Extreme learning machine (*elm*) ($R^2 = 0.14$ and $RMSE = 0.07$).

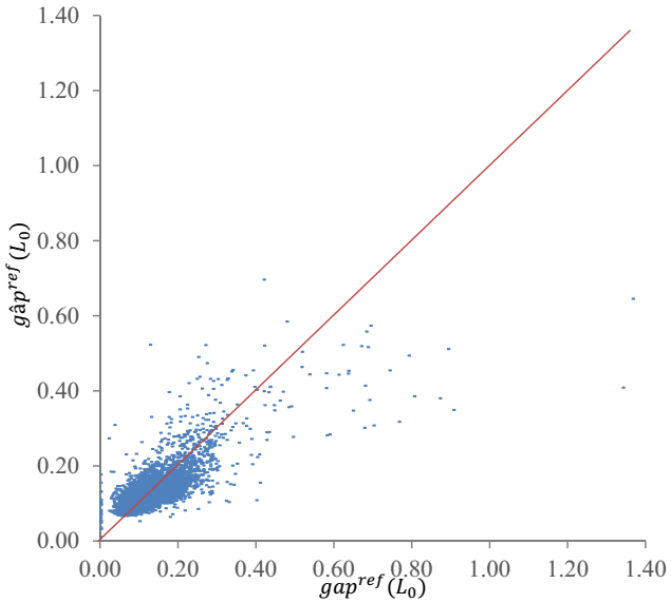


Figure 4.21 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Stochastic gradient boosting machine (gbm) ($R^2 = 0.60$ and $RMSE = 0.05$).

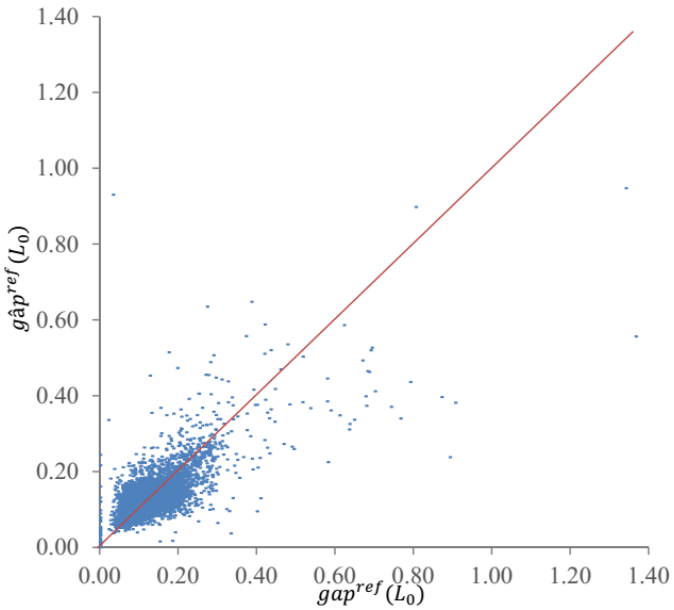


Figure 4.22 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the k-nearest neighbors ($kknn$) ($R^2 = 0.54$ and $RMSE = 0.05$).

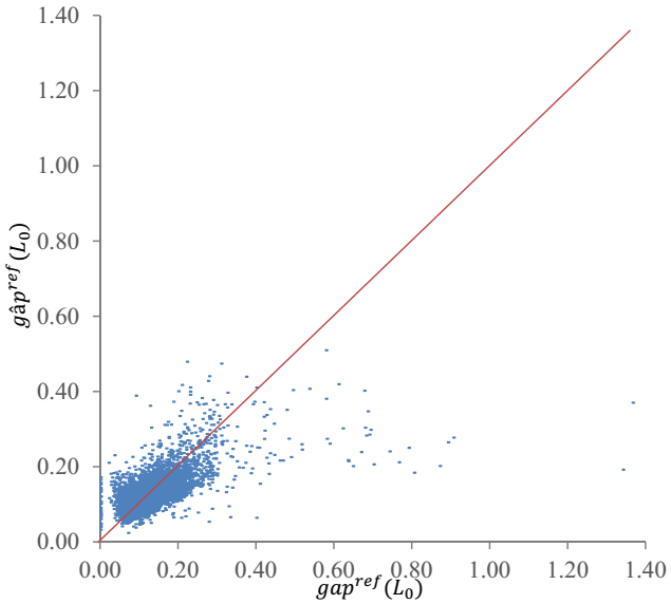


Figure 4.23 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Least absolute shrinkage and selection operator (*lasso*) ($R^2 = 0.42$ and $RMSE = 0.06$).

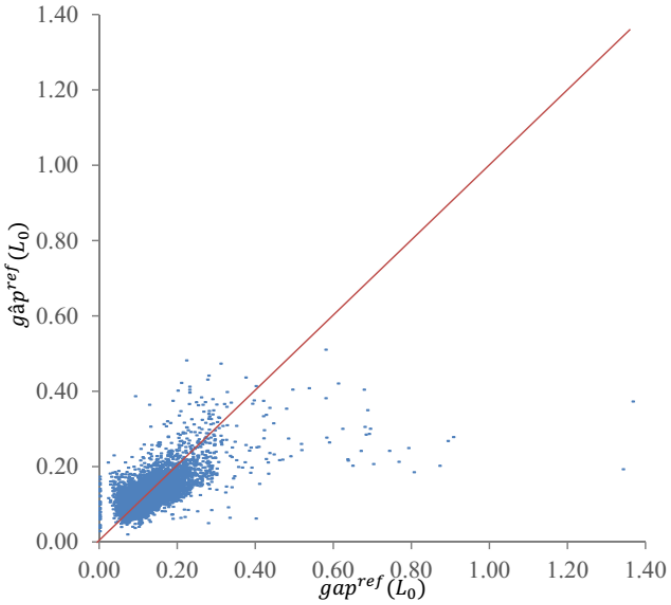


Figure 4.24 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Linear model (*lm*) ($R^2 = 0.42$ and $RMSE = 0.06$).

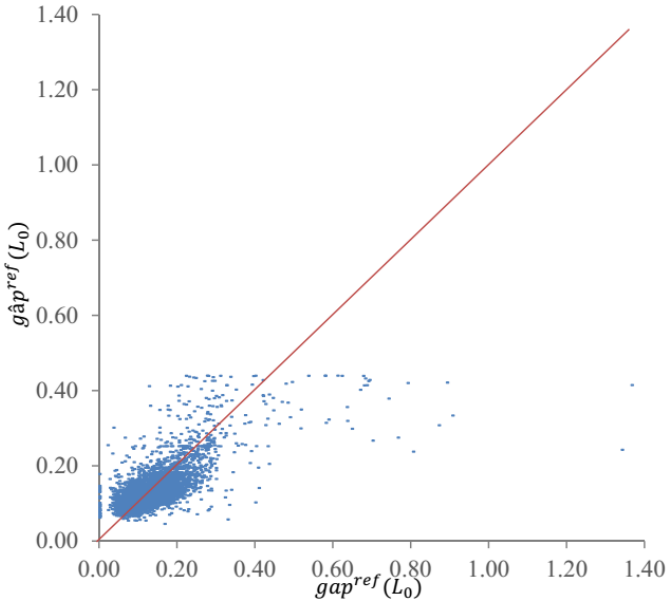


Figure 4.25 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Back-propagation neural networks (*nnet*) ($R^2 = 0.52$ and $RMSE = 0.05$).

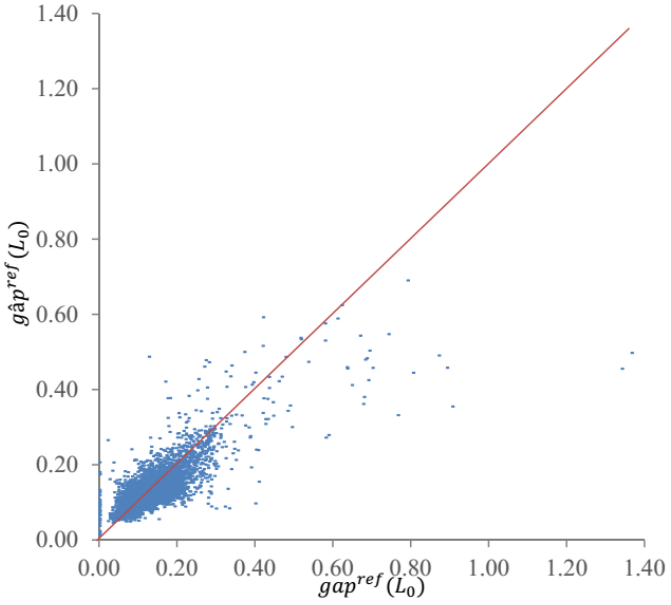


Figure 4.26 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Random forest (*Rborist*) ($R^2 = 0.65$ and $RMSE = 0.04$).

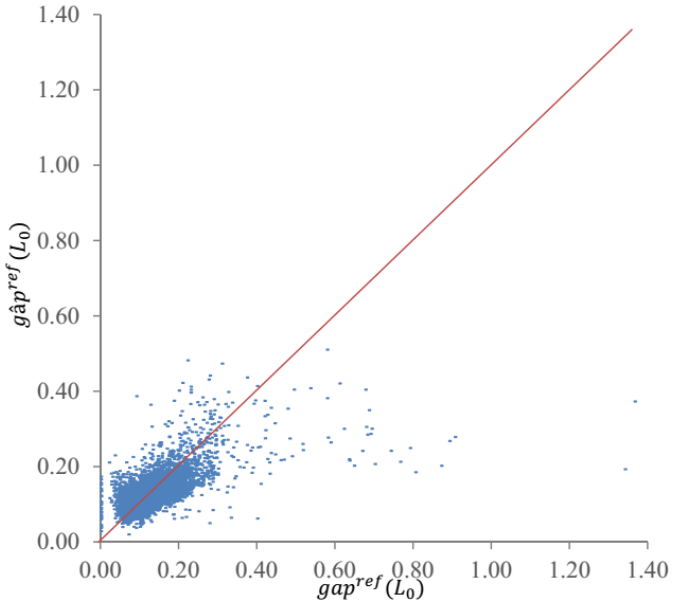


Figure 4.27 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Ridge regression ($ridge$) ($R^2 = 0.42$ and $RMSE = 0.06$).

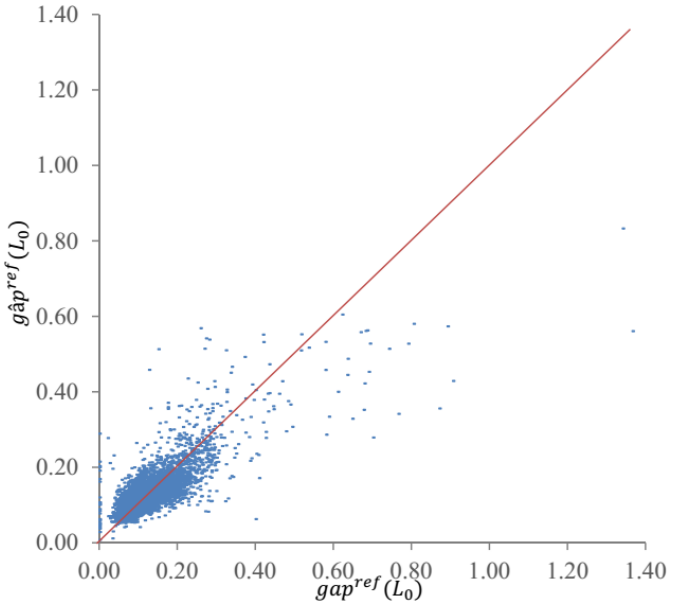


Figure 4.28 – Calculated $gap^{ref}(L_0)$ and predicted $\hat{gap}^{ref}(L_0)$ for the Extreme gradient boosting machines ($xgbTree$) ($R^2 = 0.63$ and $RMSE = 0.04$.

CHAPTER 5

Conclusions and Future Work

The main objective of this thesis is to develop a methodology based on data mining and machine learning concepts to obtain reference values for cutting and packing problems. The feasibility of the use of data mining and machine learning concepts was verified in a "pilot test", conducted using the context of the rectangular 2D-SPP with 90 degrees rotations to develop a methodology to predict a reference value, in this case the strip height. This alternative reference value can be used to assess the quality of the solutions found by heuristics, and also as stopping criterion in improvement heuristics to avoid long computational times.

To achieve the proposed objective, the thesis was divided in three parts. In a first part (Chapter 2), a literature review about the most important constructive heuristics and improvement heuristics found in the literature for 2D-SPP was developed. The second part (Chapter 3) presents the explanatory variables developed after explore the 2D-SPP characteristics. Finally, in the last part (Chapter 4) a data mining based framework to assess solution quality for 2D-SPP was proposed, by fitting regression models to predict the strip height for the 2D-SPP. These predictions used as reference values to measure the quality of solutions and as stopping criterion in improvement heuristics.

In the remaining of this chapter, a brief discussion about the main contributions of each chapter is proposed. Also, some directions for future research are provided.

5.1. Contributions

A literature overview about the most relevant constructive and improvement heuristics developed for 2D-SPP was conducted in Chapter 2. This research was fundamental to understand the problem and the main concepts considered in the last decade of publications using heuristics for the 2D-SPP.

From the simplest constructive heuristics to the most complex improvement heuristics with search over the layout, all types of strategies have been used over the years. However, most probably due to the complexity of dealing with the geometric feasibility of the layouts, the latter strategy has been less frequently used. This may be the main reason for the ratio of 1 to 4 that we have found when the number of improvement heuristics that search over sequences is compared with the number of improvement heuristics that search over layouts. When looking in more detail at the constructive heuristics, it can be seen that more than 40% of them are fitness-based heuristics. It is surely due to their good performance, when compared to positioning-based heuristics.

Research gaps are identified, mainly associated to the use of more sophisticated search algorithms, specially metaheuristics, to the use of profile representations of solutions and to the development of improvement heuristics with search over the layout.

The literature review about heuristics for the 2D-SPP was important to provide technical resources to implement the bottom-left-fill constructive heuristic and the random weight local search to compute solutions for the response variable (gap) of the regression analysis presented in Chapter 4. Also, the first concepts about problem characteristics and the lack of efficient lower-bounds for 2D-SPP with 90 degrees rectangles rotation were obtained.

The research conducted over Chapter 2 originated the following research paper:

- José Fernando Oliveira, Alvaro Neuenfeldt Júnior, Elsa Silva, and Maria Antónia Carravilla. A Survey on Heuristics for the Two-Dimensional Rectangular Strip Packing Problem. *Pesquisa Operacional*, 36(2), 197-226, 2017.

Chapter 3 is motivated by three important tasks: an exploratory approach; the dimension reduction; and the problem instances distribution. The exploratory approach was conducted to study and identify the main 2D-SPP characteristics. Parameters from seven problem generators proposed in the literature for the 2D-SPP and other cutting and packing problems served as a base for the development of 56 variables. All variables were aggregated in six groups (*Area*, *Perimeter*, *Dimensions*, *Widthdimensions*, *Proportions*, and *Other*) through a qualitative knowledge study about each variable intrinsic characteristic.

This study contributes with a methodology description of how to reduce, dynamically, the dimension of a problem. Firstly, a linear correlation method was used as a quantitative measure to validate the assignment of variables in groups. Secondly, the PCA reduced the dimensions of each group by taking advantage of the relations between variables from the same group. Finally, the 56 variables were reduced to 19 explanatory variables, retaining most part of the total variance in each group.

A total of 1,217 benchmark problem instances for the 2D-SPP were analyzed, by using the explanatory variables of each group (*Area*, *Perimiter*, *Dimensions*, *Widthdimensions*, *Proportions*, and *Other*) in order to visualize the benchmark problem instances distribution behavior. The scores for each problem instance were normalized to a scale between zero and one, according to the maximum and minimum values found for each component. Most of the benchmark problem instances found in the literature were very similar and represent only specific characteristics of the problem, although problem instances were generated using different problem generators. To fulfill this research gap, 30,000 new problem instances were generated with 2DCPackGen (Silva et al., 2014) problem generator. These new problem instances were also used as observations for the regression analysis in Chapter 4 representing different 2D-SPP variations found in real-world applications.

The 2D-SPP characterization, the 56 variables and the methodology proposed to reduce the variables' dimensions for 19 explanatory variables presented in Chapter 3 are described in the paper accepted for publication:

- Alvaro Neuenfeldt Júnior, Elsa Silva, A. Miguel Gomes, and José Fernando Oliveira. The Two-Dimensional Strip Packing Problem: What Matters?. *Accepted for publication: Operational Research, Springer Proceedings in Mathematics & Statistics*, 2018.

Chapter 4 presents a data mining based framework to predict a reference value, that can be used as a stopping criterion and to evaluate the quality of solutions found in an

improvement heuristic for 2D-SPP. The 19 explanatory variables proposed in Chapter 3 were used as predictors in order to provide a proper measure of the 2D-SPP characteristics. Also, the dynamic reduction methodology proposed in Chapter 3 was validated with 30.000 problem instances generated to fit the regression model.

A BLF constructive heuristic with a random weight local search procedure with 100 runs was run using different sequences. The lowest value found with the heuristics and area lower bound was used to calculate the normalized response variable (gap). Finally, the observations were the 30,000 problem instances generated with 2DCPackGen to represent different 2D-SPP variations found in real-world applications.

For the data mining approach, the predictors, the gap, and the problem instances served as input to fit the regression model. The output value was the prediction of the response variable, that can be converted to the total height necessary to pack a set of rectangular small rectangles into a rectangular strip. The good level of generalization (R^2) and accuracy ($RMSE$) of the regression model were confirmed through 5-fold cross-validation process ($R^2 = 0.68$ and $RMSE = 0.04$) and with an additional validation process ($R^2 = 0.55$ and $RMSE = 0.08$) using 6,000 new problem instances generated with 2DCPackGen and 986 problem instances found in the literature.

Different data mining techniques were used to fit the regression model. The Random forest was inferred statistically as the best choice to develop the regression analysis, based on its capacity to generalize the explanatory variables for the gap. The Stochastic and Extreme gradient boosting, Cubist, and k-nearest neighbors also have significant generalization capacity.

In the assessment of the framework performance, the regression model was adopted to predict the strip height for new problem instances, demonstrating the good performance of the predictions as a stopping criterion. On the other hand, the predictions' robustness was verified considering the relation between the quality of the solutions obtained with a local search procedure, the number of iterations and variations in the predictions. For a variation of $\pm 2\%$ in the predictions, the mean gap is very similar for almost all problem instances during the local search procedure, meaning that the quality of the solutions obtained is not greatly affected by small variations of the predictions, stressing that the predictions were robust.

During the assessment of the framework performance section, it was verified that the difference between the optimal solution and the area lower bound increased substantially with the increase in the space wasted in the arrangement of the rectangles into the strip. In addition, the predictions are more accurate than the lower bound. Also, it was confirmed that the area lower bound works well if none or few waste spaces are available in the arrangement of rectangles in the strip.

Finally, we demonstrate that the data mining based framework proposed is consistent and can be applied to predict an output variable value for other cutting and packing problems, as the bin packing problem, knapsack problem or cutting stock problem.

Chapter 4 resulted in the following research paper:

- Alvaro Neuenfeldt Júnior, Elsa Silva, A. Miguel Gomes, Carlos Soares, and José Fernando Oliveira. Data Mining Based Framework to Assess Solution Quality for

the Rectangular 2D-SPP. *Submitted for publication*, 2017.

In Appendix A a literature review was developed to complement the knowledge about data mining, in specific for the most important supervised machine learning techniques available to fit regression models, divided in six groups according to the similarities between the concepts of each technique.

Finally, a practical guide to fit regression models using *R* and *RStudio* software was presented, together with a small explanation and the specific codes to reproduce "caret" functions. The simple structure of the practical guide allows adaptations to fit regression models for different types of problems with different cross-validation configurations.

5.2. Future work and research opportunities

In this section, some opportunities for future studies in 2D-SPP are presented. Regarding theoretical and bibliographical research studies, it can be identified the lack of literature reviews addressing exact and hybrid methods for 2D-SPP.

A methodology based on linear correlations and PCA method was proposed to reduce the problem dimensions in Chapter 3. One alternative is to replace the linear correlation and the PCA for features selection methods, given by different approximation criterion indicators used to measure how similar are two or more predictors (Cadima et al., 2004). The objective would be to select only predictors capable to describe the most important problem characteristics, avoiding the interference of irrelevant predictors.

The 2D-SPP characterization considering the 19 predictors obtained with methodology of dimension reduction, proposed in Chapter 3, can have an important contribution in the development a new unified hyper-heuristic framework for the 2D-SPP. The hyper-heuristic could be guided by the predictors in the design of heuristics to solve more efficiently different types of problem instances. A similar approach was developed by López-Camacho et al. (2014) for the bin-packing problems and Hall and Posner (2007) to select the best algorithm for 0-1 knapsack problem.

The predictors developed can be also used in the development of a framework to analyze the strengths and weaknesses of optimization algorithms performance for different types of 2D-SPP problem instances, as proposed by Smith-Miles et al. (2014) for the graph coloring problem, López-Camacho et al. (2013) for 1D or 2D irregular bin-packing problems, and Perez et al. (2004) for the bin packing problem.

In what concerns the use of supervised machine learning techniques, emerged as special interest the exploration in more details of neural network techniques to fit regression models. It would be necessary to increase the number of observations in training dataset, in order to improve the neurons' learning capacity to find patterns in the information provided by the training dataset. It would be also interesting to explore other data mining techniques not considered in this thesis (e.g. Radial Basis Function Net, Projection Pursuit Regression, or L2 Regularized Support Vector Machine with Linear Kernel).

Additional studies could explore different methodologies to select the best data mining technique. It would be interesting to compare the performance of the results obtained with the Friedman's hypothesis tests and the Nemenyi post hoc test method (Demšar, 2006) used

in this thesis with new technique selection methodologies (e.g. ANOVA for the parametric hypothesis and the Bonferroni-Dunn for the post hoc test).

For each problem instance used to fit the regression model, a replacement process, similar to the backforward step found in the back-propagation neural networks technique, can be a good strategy to adjust the response variable value (the gap). The objective with this iterative process is to replace the area lower bound by the predictions obtained after fit the regression model, finding new and a more accurate value for the gap.

Other constructive heuristics could be applied to fit regression models and obtain predictions for 2D-SPP problem instances. Also, test the predictions' performance as a stopping criterion for different improvement heuristics is another future research work possibility, assessing the data mining based framework performance and the predictions' robustness for different situations.

Finally, the data mining based framework developed for the 2D-SPP can be extended to fit regression models for other types of cutting and packing problems, as the bin packing problem, knapsack problem cutting, and cutting stock problems.

References

Cadima, J., Cerdeira, J. O., and Minhoto, M. (2004). Computational aspects of algorithms for variable selection in the context of principal components. *Computational Statistics & Data Analysis*, 47(2):225–236.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30.

Hall, N. G. and Posner, M. E. (2007). Performance prediction and preselection for optimization and heuristic solution procedures. *Operations Research*, 55(4):703–716.

López-Camacho, E., Terashima-Marín, H., Ochoa, G., and Conant-Pablos, S. E. (2013). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics*, 145(2):488–499.

López-Camacho, E., Terashima-Marin, H., Ross, P., and Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications*, 41(15):6876–6889.

Perez, J., Frausto, J., Cruz, L., Fraire, H., and Santiago, E. (2004). A machine learning approach for modeling algorithm performance predictors. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 70–80. Springer.

Silva, E., Oliveira, J. F., and Wäscher, G. (2014). 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research*, 237(3):846–856.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45(1):12–24.

Appendix A

Machine Learning: A Review of Regression Techniques

A.1. Introduction

Predict the future based on current information and experience provided about a problem is a difficult task for decision-makers. Inserted in the knowledge discovery context, data mining and machine learning are multidisciplinary computer science concepts used to explore a problem. Generally, a data mining process, using machine learning techniques, manipulates huge volumes of information in order to understand the input data structure, looking for patterns aiming to find useful associations and systemic relations for further analysis, benefited by a high logical data processing capacity provided by computer systems (Fayyad et al., 1996; Kuhn and Johnson, 2013).

The main idea is to use machine learning techniques capable of automatically understand the data about a problem, trying to identify patterns in the input data, according to the output data. These techniques learn the behavior of the dataset based on hypothesis and can be separated into two groups: non-supervised and supervised machine learning techniques.

The non-supervised techniques aim to explore a dataset, taking only in consideration the information available in the dataset (not considering any output variable), associating explanatory (input data) variables by similarity. In the supervised machine learning techniques, a previous knowledge about how a response variable was obtained allows the identification of which technique can be applied to study the behavior of a specific input data. Additionally, a measure of error (e.g. mean-square-error) is used as external supervisor to evaluate the differences between the known response variable and the prediction of this response value.

According to the type of data, data mining process based on supervised machine learning techniques are used for classification analysis when data is discrete (or categorical), or for regression analysis when data is continuous. In a general form, the regression analysis involves the use of three different types of variables: explanatory variables (x_1, x_2, ..., x_n), one response variable (y), and unknown coefficients (β_0, β_1, ..., β_n) to represent the relevance of each explanatory variable to explain the response variable.

This chapter aims to present an overview of supervised machine learning techniques used to fit regression models and predict a response variable for different problems. A practical guide about the *RStudio* software (RStudio, 2017) was developed as a "quick-start" support for future users, exploring in specific the functionalities found in the "caret" package (Kuhn, 2017). To complement the practical guide, a literature review about supervised machine learning techniques used to develop regression analysis was conducted,

aiming to explain how each technique manipulates the input data in order to obtain the outputs.

Over the years, few data mining applications for cutting and packing problem context are found in the literature. The main applications are focused on reducing the dimensional space of variables for specific problems using principal components analysis (López-Camacho et al., 2010; Smith-Miles et al., 2014), which is based on classification supervised machine learning concepts. No relevant publications about the use of regression analysis in cutting and packing problems were found.

The chapter is organized as follows. Section A.2 describes data mining terminologies used during the research thesis. Section A.3 presents a methodological knowledge discovery process about how to apply supervised machine learning techniques to develop a regression analysis, together with some relevant concepts about cross-validation and technique selections procedures. Section A.4 presents a brief discussion about important supervised machine learning techniques for regression analysis. A discussion about how the techniques can be applied to compute regression models using *RStudio* software is presented in Section A.5. Finally, Section A.6 shows the most important findings and some future research ideas obtained during the development of this chapter.

A.2. Terminology

Since the data mining and machine learning processes are products of research and contribution of different authors, many concepts are very similar. In this section is presented the terminologies adopted in Chapter A and during all research thesis:

- The term observation is used to refer a single and independent unit of data about the problem, such as the problem instances for 2D-SPP;
- The terms dataset or sample is used to refer a set of observations;
- Explanatory, independent, input variables, or predictors are the variables used as input data source about the problem to be analyzed;
- Response, dependent, target, or output variables are the quantifiable output event or measure to be predicted;
- Data mining techniques, supervised machine learning techniques or only techniques are the models used to fit regression models;
- Predictions or estimations are the response variables calculated after fit regression models.

A.3. Knowledge discovery for supervised machine learning

The data mining process based on supervised machine learning techniques was developed to evaluate the inductive hypothesis capacity for the prediction of one response variable,

from exploratory variables and observations about the problem provided as input data (Gama et al., 2015).

Over the years, different reasons were identified for the development of inaccurate regression models. Kuhn and Johnson (2013) described four facts that can produce inaccurate regression models: (1) unjustified extrapolation for other problems; (2) inadequate pre-processing of the data; (3) inadequate model validation; (4) overfit model only for current dataset observation.

To answer these situations and to avoid fit inaccurate regression models, Figure A.1 describes the methodological knowledge discovery, developed based on concepts found in the Cross-Industry Standard Process for Data Mining (CRISP-DM) (Chapman et al., 2000; Wirth and Hipp, 2000). Other relevant studies used to develop the methodological knowledge discovery are the classification system proposed by Kotsiantis (2007) for classification supervised machine learning techniques, and the general overview about knowledge discovery proposed by Fayyad et al. (1996).

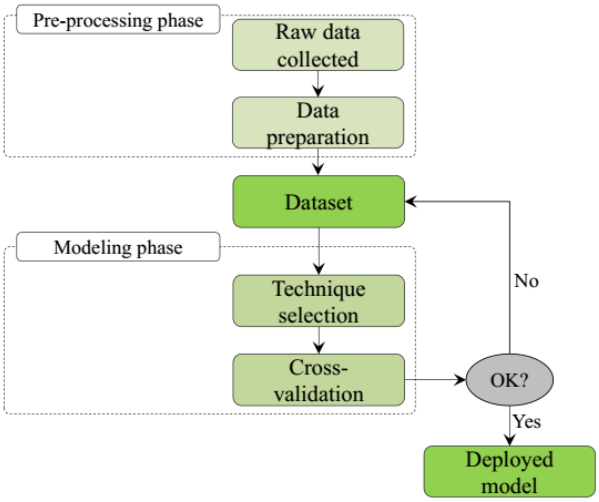


Figure A.1 – The methodological knowledge discovery for supervised machine learning regression analysis (adapted from Fayyad et al. (1996), Chapman et al. (2000), and Kotsiantis (2007).)

To process the information about the problem, the methodological knowledge discovery for supervised machine learning was divided into two parts: the pre-processing phase and modeling phase. The pre-processing phase begins with problem recognition by collecting the data based on the problem characteristics, parameters, and factors, by getting relevant information available about the problem context. After the raw data collection, the raw data must be prepared, by cleaning and handling the missing data, the outliers (noise values) and, if necessary, by removing possible missing values or misleading information. At the end of this phase, all data must be consistent and coherent with the problem characteristics.

The first modeling step is to select the best technique from a set of candidates to fit the regression model. This approach is closely related to selection process found for statistical

analysis, named as model selection problem (Cox, 2006). Different measures to evaluate a technique efficiency can be adopted, as the total time spent to fit a model, the model accuracy and/or the model level of generalization. In specific, the last two options were extensively explored by the literature over the years.

The model accuracy describes how good is the adjustment between calculated and predicted response variable values, using the Mean-square-Error (MSE), the Mean-Absolute Distance (MAD) or the Root-Mean-Square-Error (RMSE) measures (Kuhn and Johnson, 2013). The level of generalization can be measured by the coefficient of determination R^2, in which the proportion of variance in the explanatory variables compared with the response variable is calculated (Demšar, 2006).

Regardless the selection measure, the performance of a technique is highly dependent on the problem, and controlled experimental evaluations must be conducted to select the technique that fits better for the regression analysis of the problem.

To test the techniques' performance, small representative subsamples from the dataset are used. Based on the holdout cross-validation predictor concept (Devroye and Wagner, 1979; Arlot and Celisse, 2010), each subsample is divided into two parts, assign 2/3 of observations to training models and the remaining 1/3 of observations to test models. The performance of all techniques is measured considering only the results obtained for test subsamples.

For simple problems, individual selection measures can define the superiority or inferiority of a technique in comparison with a set of candidates. However, for complex problems, the use of individual selection measures is not recommended. A common manner to compare the performance of different techniques is to perform statistical comparisons, based on predictions' accuracy measures (e.g. RMSE) or a level of generalization measure (e.g. R^2).

Hypothesis tests are a useful tool to verify if one technique is significantly different from other techniques. In Nemenyi post hoc hypothesis test (Demšar, 2006) a critical difference measure was proposed to verify which techniques are significantly similar or not. If the techniques' performance is significantly similar, then any technique can be selected to fit the regression model. Also, statistical measures based on resampling datasets using pairwise comparison tests can be defined to visualize significant differences between techniques (Yildiz and Alpaydin, 2006; Hothorn et al., 2006; Eugster et al., 2008; Wang et al., 2015).

A general scheme describing the most important steps to conduct a technique selection process is shown in Figure A.2.

The second modeling step is the cross-validation. The main objective is to validate the regression model level of generalization to avoid overfit models and obtain, after concluding the modeling phase, more accurate predictions for new problem instances.

The cross-validation process proposed in this step is similar to the process used in the technique selection step. The difference is that the regression models are fitted in the cross-validation process with a more detailed description of tuning parameters and model post-processing analysis. Tuning parameters are factors that impact on the regression model fitting, in order to enable the technique to obtain a better level of generalization and accuracy of the predictions. Each technique has specific tuning parameters, which vary according to

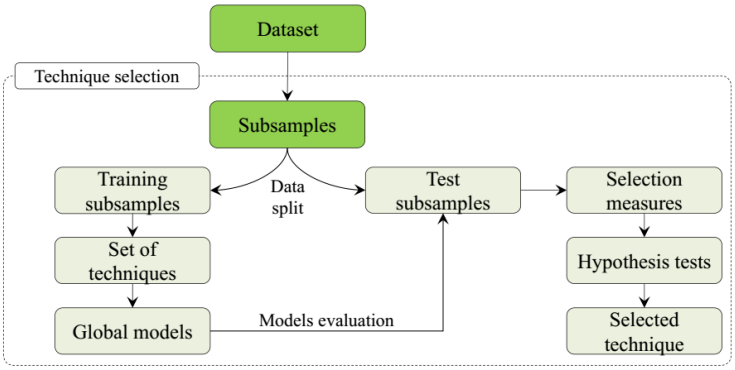


Figure A.2 – Technique selection process.

their characteristics.

Thus, the cross-validation splits the dataset into two different parts: Training and test datasets. The training dataset is used to fit the regression model, while in the test dataset the regression model performance is evaluated.

Regression models are fitted using the explanatory variables, observations provided by the training dataset, and specific tuning parameters of the selected technique. A post-processing step is conducted to evaluate and select the best tuning parameters values for the problem. For almost all techniques more than one regression model is fitted, and the final global regression model will be the model that produces the smallest error between the predicted and known response variable values.

The performance of the global regression model is evaluated using the test dataset, in which only information about the explanatory variables is provided. A general description of the most important steps to conduct a cross-validation process is shown in Figure A.3.

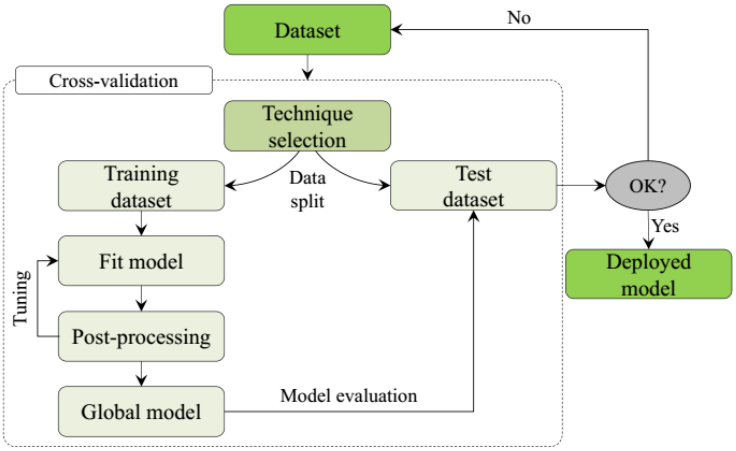


Figure A.3 – Cross-validation process (adapted from Kotsiantis (2007)).

After finishing both pre-processing and modeling phases, the deployed regression model can represent the main problem characteristics and predict the response variable value for new observations.

A.4. Supervised Machine Learning Techniques

In this section concepts about supervised machine learning techniques used to fit regression models are described in details. The supervised machine learning techniques were associated based on their similarities. Six concepts have raised:

- Memory-based;
- Neural networks-based;
- Ensemble learning-based;
- Regression-based;
- Regularization-based;
- Rule-based.

The focus in this section is to present details for each one of the six concepts of supervised machine learning techniques.

A.4.1 Memory-based techniques

The memory-based (also known as instance-based) techniques consider the distance between all data to calculate the predictions. With this affirmation, it is possible to assume that similar data are within a close distance. The learning process memorizes the distance between observations and uses this information to predict the response variable for new observations (Daelemans and Van den Bosch, 2005).

The most used memory-base technique is the k-nearest neighbor's algorithm (Altman, 1992). Four components must be defined to develop a learning process using k-nearest neighbors algorithms: (1) the number of nearest neighbors, (2) a similarity distance function metric, (3) a weighting function, and (4) a cost-function to calculate the predictions. The total number of neighbors (k) is defined to calculate the distance function to predict the response variable value for new observations.

The similarity distance functions (e.g. Euclidean, Manhattan or Minkowski) is calculated between its exploratory variables of a new observation and the k-neighbors observations. Since the predictions are obtained from a small subset of k neighbors observations, the next step is to verify the weight w of each k observation. The weight can be associated with an inverse relation to the distances calculated for all k neighbors, based on kernel regression non-parametric statistics. All k neighbors observations receive a weighted value between 0 and 1, varying according to a weighting function (e.g. Gaussian function).

The cost-function to predict the response value (y') for new observations is given by the average value found considering the contribution of all k neighbours and its response values (y_i) proportionally to the distance from the new observation[1], in which closer neighbor observations (i) have a greater contribution to the prediction (higher weights).

[1] $y' = \sum_{i=1}^{k} w_i y_i / \sum_{i=1}^{k} w_i$

The choice of k is fundamental to predict correctly new observations. Larger k values reduce the effect of noise but make boundaries between different observations less distinct, the weight w of all neighbor observations is almost equal to one, and the computational time required to process the distances for all predictions increase. Instead, for small k values almost all weights are equal to zero. Both larger or small k values result in inaccurate predictions. The challenge is to select the best k considering the intrinsic characteristics of each problem.

A.4.2 Neural networks-based techniques

Based on the structure and functions of the human nervous system, the neural networks techniques simulate the ability of neurons to obtain knowledge of already known observations about a problem. An Artificial Neural Network (ANN) is composed of nodes (neurons) that compute mathematical functions, located in a computational processing layer. Each connection (synapse) between nodes have controllable weighted values in the information received by each neuron.

In general, two aspects are considered to process an ANN: the architecture and the learning process. The architecture is related to the type and number of neurons, and the number and type of layers (input, hidden or output) needed to learn information from the problem. In the learning process, the rules for adjusting the connection weights are defined.

The hidden layer can be composed by more than one layer to process the input information to obtain more precise output values (predictions), in a concept named as Multilayer Perceptron (MLP). The MLP was developed to solve problems that cannot be linearly separable, adding a new intermediate layer capable to approximate any function. However, the use of intermediate nonlinear, continuous, differential and descending functions to connect intermediate hidden layers is necessary (e.g. sigmoid function). From the last intermediate hidden layer to the output layer, a linear function converts the information provided on the same scale of the response variable.

The Back-propagation algorithm (Schmidhuber, 2015) was developed to training the ANN for MLP applications, based on the gradient-based optimizer, as the Limited Broyden Fletcher Goldfarb Shanno (L-BFGS) method (Liu and Nocedal, 1989).

Two steps are needed to train the network: the feedforward and backward. For the feedforward step, the input data are processed in hidden layers by the weight of each connection. The neurons process the information based on the activation function to produce output values, to be used for the next layer until the output layer is reached. The output values (predictions) are compared with the known response variables values to measure the predictions' error.

To adjust the input connections weights, in the backward step the error found in the feedforward step is used to improve the accuracy of the regression model. This process in cycles improves the neural network learning capacity, using the output values already obtained in past cycles, to adjust the weight of each connection between neurons at each cycle, until a stopping criterion is reached (e.g. the maximum number of cycles, the maximum error rate).

Figure A.4 shows a small representation of the back-propagation neural networks. With

one input layer containing three input data, two hidden layers with five neurons (three in the first hidden layer and two in the second hidden layer) and, finally, one output layer to predict the response variable values. The backward step can be found between the input layer and the first hidden layer and between hidden layers.

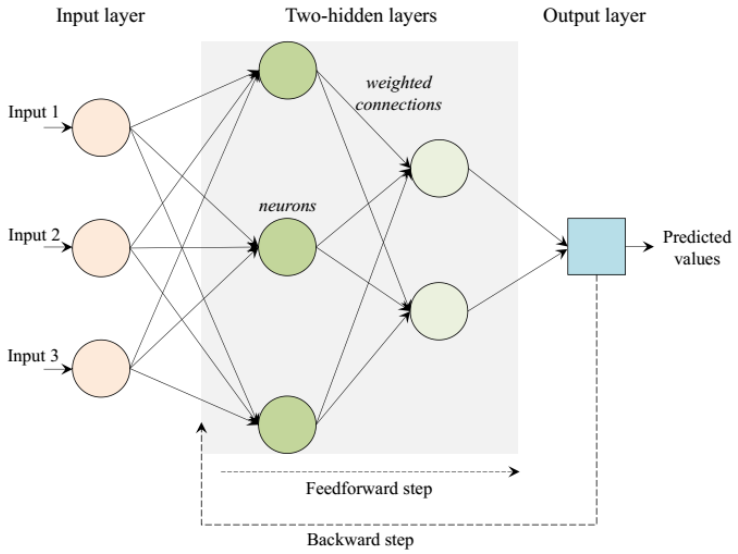


Figure A.4 – Example of back-propagation neural networks process.

Although the dataset is splitted in training and test datasets, the major drawback of the back-propagation algorithm is the potential overfit and overtraining effects, which reduce the neural network capacity to predict more precise output values for new observations. Also, the process in cycles can be slow to converge to good results when the training dataset is very large. Over the years, hybrid neural network techniques were developed to solve the potential effects of overfitting the neural network weights.

The Extreme learning machine technique (Huang et al., 2006) is a feedforward neural network technique with a single hidden layer and random neurons. The learning process has only one cycle, where the weights of the connections between neurons are based on linear functions. This simple format is the main advantage of the extreme learning machine technique when compared with back-propagation algorithms, allowing to compute much faster the neural network.

Another well-known technique is the Bayesian regularized neural networks (MacKay, 1992). A probabilistic density function is added to measure the weight of the connections between the known dataset and the neural network, given by the Baye's rules (Foresee and Hagan, 1997). Also, overly complex models are penalized as unnecessary linkage weights are effectively driven to zero.

A good example of the application of the Bayesian regularized neural network can be found in Ticknor (2013). The financial stock market behavior is forecast considering as input daily market prices and financial technical indicators and as output the one-day future closing price. The data from two specific stock market companies were selected to test and visualize the effectiveness of the proposed regression model.

A.4.3 Ensemble learning-based techniques

The ensemble learning techniques adopt multiple algorithms with the same base learner, to improve the predictive performance when compared with single algorithms, taking advantage of the best qualities of each algorithm. Weak and strong learner performance qualities of the methods used to develop the ensemble technique are capable to produce more accurate predictions.

The Random forest (Breiman, 2001) is one of the most important ensemble techniques that combine multiple decision trees as learner approach algorithms. The final prediction is the result of the mean prediction value of the individual results obtained in each decision tree. A single decision tree is a simple and organized representation of input data to obtain as output predictions. Each tree is composed of nodes containing explanatory variables. For a specific node, the explanatory variable is selected by the difference between the mean-square-error of the predictions before and after the node. The explanatory variable that reaches highest mean-square-error difference is selected to be placed in this node. The decision trees are composed of a hierarchical structure, in which nodes (explanatory variables) located in the first levels are more relevant to fit the regression model.

To improve the random forest level of generalization, the decision trees are generated using the holdout method. For each tree, 2/3 of observations ("bag" observations) are randomly defined to training and fit the regression model. All rest 1/3 observations ("out-of-bag" observations) are used to test the regression model, predicting the mean-square-error between the known response variable and the prediction. The replacement option allows that any observation can appear multiple times within the "bag" or "out-of-bag" observations.

Aiming to reduce overfitting regression models, the random forest level of generalization is measured by the mean-square-error of "out-of-bag" observations for each decision tree. Another possibility is to introduce a gradient value to reduce the error during the training phase.

For the Stochastic gradient boosting machines, a constant f ($0 < f \leq 1$) defines subsamples of training dataset (Friedman, 2002). More random subsamples with small trees are created with lower f, reducing the computational processing times and introducing more overall randomness. Also, small subsamples have different characteristics when compared with the training dataset, which increases the possibility of fit more inaccurate models. Instead, a high level of similarity between the subsample and the training dataset is obtained with higher f, which is not convenient to avoid overfitting regression models. The challenge is to find the best gradient f for the problem.

The Extreme gradient boosting machines (Friedman, 2001) uses formal regularized parameters or techniques formalization to avoid overfitting models. A gradient boosting M (e.g. the maximum number of trees for techniques based on decision trees) is proposed to control the size of training dataset subsamples. As the parameter f considered in stochastic gradient boosting machines technique, the challenge is to find the best parameter M to avoid fit too much generic or overfitted regression models.

For stochastic and extreme gradient boosting machines based on decision trees, the predictions' error is calculated using the "out-of-bag" observations, and the final regression

model is fitted by the generalization of regression models of each tree.

A.4.4 Regression-based techniques

The regression-based techniques are based on parametric and non-parametric statistics. In the linear models, the dataset is parametric and the regression model function is defined taking into account a linear relation between the explanatory variables and the response variable. The Multivariate Adaptive Regression Splines functions (MARS) is composed by piecewise linear kinks, considering the nonlinear distribution behavior between all observations, which results in nonlinear hinge functions intervals *max(0,x-ct)* and *max(0,ct-x)*, in which *ct* is a constant value called knot (cut point).

Figure A.5 shows an example of linear and MARS regression models curves for the same dataset, considering one explanatory variable (x) and one known response variable (y). The MARS curve finds the plane that minimizes the sum-of-square error, adjusting the coefficients β according to the observations' distribution, comparing the known response variable and the explanatory variables.

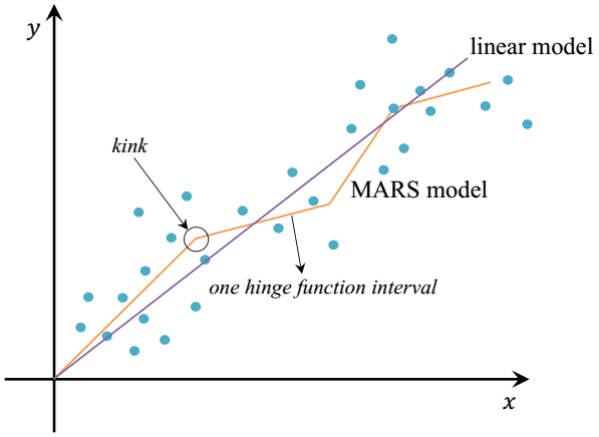


Figure A.5 – Example of linear and MARS curves.

Negative and positive slopes are created for each piecewise linear curves. When a problem is composed by more than one explanatory variable, then MARS will remove (if the backward pass is selected) or add (if the forward pass is selected) sequentially the interval which reduces most significantly the sum-of-square error between the known and predicted response variables. The addition of new intervals is penalized, to avoid overfit models. An additional group of hinge functions intervals is obtained from the relation of two or more explanatory variables if the number of degrees is greater or equal to 2. The problem of allowing a number of degrees greater than 1 is to understand the cause-effect relation between the explanatory variables and the known response variable.

In general, MARS when compared with neural networks and most of the ensemble techniques, requires low computational processing times to fit regression models with a good level of generalization.

A.4.5 Regularization-based techniques

The regularization-based techniques applies penalization functions to control the variance between explanatory variables, aiming to fit more simple and generalized regression models. Applications of regularized techniques are recommended when the number of explanatory variables defined for the problem is very high compared with the number of observations. The Ridge technique and the Least Absolute Shrinkage and Selection Operator (LASSO) technique are two of the most notorious regularization-based techniques found in the literature.

For both Ridge and LASSO, Tikhonov regularization functions are used as a rule to define the regularization functions curve behavior, varying for different p parameter values. For example, if p=*1* then the regularization (L2 regularization) curve is a circle, used as regularization curve to define the Ridge regression coefficients. A diamond curve is obtained if p=*2*, used as regularization (L1 regularization) curve to define the LASSO regression coefficients.

In Ridge regression function (Hoerl and Kennard, 1970) the objective is to minimize the regression function β, considering the minimization of the explanatory variables coefficients (β_j, $\forall\ j \in J$) between all observations (i, $\forall\ i \in n$). The β function is given by,

$$\beta = \sum_{i=1}^{n} (y_i - y_i')^2 + \lambda \sum_{j=0}^{J} \beta_j^2 \tag{A.1}$$

in which y_i is the known response variable and y_i' is the prediction obtained varying the β function. The combination of the Ordinal Least Square (OLS) error function curve and Ridge L2 regularization curve is analyzed at the same time.

The lower is the value of the coefficients, less representative is the explanatory variable for the regression model. The Ridge L2 regularization curve uses as a constraint function that varies according to a quadratic penalty parameter applied as a tuning parameter (λ).

Different solutions can be found varying λ. The challenge is to find the best trade-off between the minimum number of explanatory variables needed to reduce the residual sum of squares and a good λ. For $\lambda = 0$ there is no penalization. Instead, if λ is large, then the explanatory variables are heavily constrained and the regression model degrees of freedom will be lower.

The second regularization-based technique explored was LASSO. As the same manner verified in Ridge regression functions, in LASSO technique the main objective is to minimize the explanatory variables coefficients values (Tibshirani, 1996), as in the Ridge regression technique. The fundamental difference between Ridge and LASSO is the shape of the regularization curve. While in Ridge the shape of the curve is a circle (p=*1*), in LASSO the curve assumes a diamond shape (p=*2*). Once again, β function is given by the combination of the OLS error function curve and LASSO L1 regularization curve:

$$\beta = \sum_{i=1}^{n}(y_i - y_i')^2 + \lambda \sum_{j=0}^{J}|\beta_j| \tag{A.2}$$

Figure A.6 shows representations of Ridge (left graph) and LASSO (right graph) curves contours. The diamond shape curve brings an important consequence to fit regression models using LASSO, in which the best results for the β minimization are located in one of the four corners available by the regularization curve. If the problem is defined by two explanatory variables (x_1 and x_2), at least one of the two coefficient value must be equal to zero ($\beta_1 = 0$ or $\beta_2 = 0$). For more than two explanatory variables, more than one coefficients can be equal to zero.

Different from Ridge regression technique, in which the best β function value does not eliminate any explanatory variable, in LASSO the process works as a feature selection. Only the most important explanatory variables are selected to fit the regression model, while the less important variables are excluded (coefficients equal to zero).

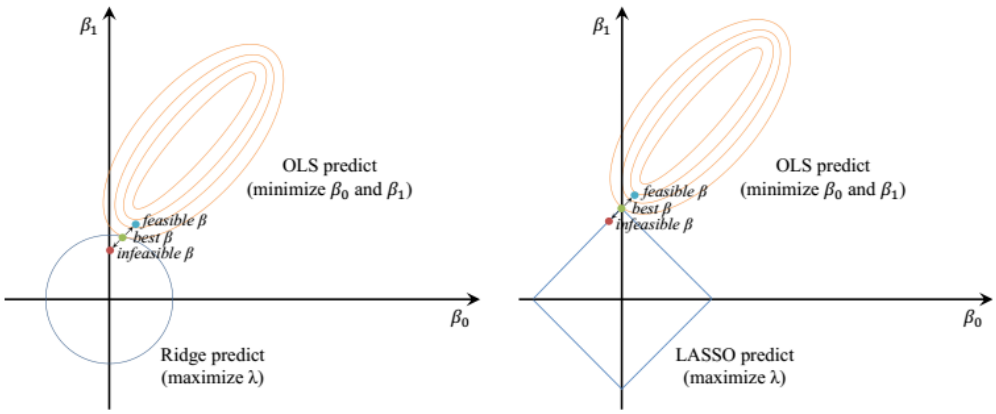


Figure A.6 – Ridge and LASSO β minimization process.

Finally, the Elastic Net technique(Zou and Hastie, 2005) combines the representation curves used in Ridge and LASSO regression techniques, in order to obtain a unique regression model, incorporating the Ridge effective regularization capacity and LASSO feature selection process.

A.4.6 Rule-based techniques

A rule-based technique is developed based on a set of relational rules to simplify the representation of an original problem (Weiss and Indurkhya, 1995).

Cubist (Kuhn et al., 2014) is a technique based on decision trees that are capable of dealing with missing values in the datasets. The outputs of the Cubist technique are linear models in each tree terminal nodes. Intermediate linear regression models can be obtained at each split of the tree and can be useful to smooth the effects of regression models generated in terminal nodes, in a recursive process. Finally, the tree is reduced to regression

models for different rules to predict the response variable values.

Similar to Ensemble Learning techniques, the Cubist combines the characteristics of regression models created by different rules conditions to avoid improve the level of generalization and overfit regression models.

A.5. *R* and *RStudio* practical guide

The *RStudio* (RStudio, 2017) is a free and open-source interface for data analysis software based on *R* programming language, with a friendly interface that allows the user to analyze datasets in different situations. The most common application of *RStudio* is through available packages already developed by other users. The Comprehensive *R* Archive Network (CRAN) repository features more than 10,000 available packages for different types of applications (CRAN, 2017).

One of the most important packages to develop supervised machine learning models is the "caret" package. In "caret", functions are available to streamline the predictive model's development process, by a uniform interface that allows standardizing both input parameters and output data. Also, the functions are performed using parallel logic to improve computational efficiency. More information, examples and complementary sources about the "caret" package are available in Kuhn (2008). Also, in Kuhn (2017) is described details about how to apply some of the most important functions of "caret" package, since the pre-processing phase until all possible output data available about the model developed.

Our objective in this section is to propose a practical guide to implement the modeling phase of the methodological knowledge discovery framework using the *RStudio* interface and the "caret" package. The main focus is to describe a small explanation to develop the cross-validation phase, after the technique selection step, using the "caret" package.

The first step is to format in a spreadsheet (generally the excel) the dataset with all input data obtained about the problem after the pre-processing phase (the known response variable, the explanatory variables, and the observations). The dataset is organized in columns and rows, in which each row is reserved for the observations and the columns for the explanatory variables and the known response variable. For this practical guide, it is important to take into consideration that the last column must be reserved for the known response variable.

All data provided in the spreadsheet can be used directly to perform the regression analysis. Therefore, to simplify the input of all dataset into the *RStudio*, the data is saved as .txt file, respecting the formatting in rows and columns proposed in the spreadsheet. This .txt file (named as *dataset.txt*) is read by *RStudio* using the function read.table, assuming the name of dataset for future uses.

```
> dataset=read.table("dataset.txt",head=T)
```

After load the dataset with all observations, the explanatory variables and the known response variable, the cross-validation starts by splitting the dataset in training and test datasets. In our case, the data pre-processing phase were fulfilled before load the dataset.

The function createDataPartition define the number of observations designed (p) for training or test datasets. If p=0.8, 80% of dataset is allocated for training dataset, and 20% for test dataset. Also, the number of times in which the dataset will be partitioned (times=1) or the number of folds (k=1) in each partition are defined. The TrainIndex will be show on a matrix format if list=FALSE, and in a list format if list=TRUE. In TrainIndex all information about the dataset partition is saved. The dataTraining is defined for training dataset, and the dataTest is composed by all rest observations not listed in TrainIndex. A complete guide containing all specifications about how split the dataset, including other splitting options can be found in Kuhn (2017), *section 4*.

```
> TrainIndex<-createDataPartition(dataset$gap,p=0.8,times=1,list=FALSE)
> dataTraining<-data[TrainIndex,]
> dataTest<-data[-TrainIndex,]
```

The cross-validation control parameters is defined in the next step, using the function trControl function. The repeated cross-validation (method=repeatedCV) was the resampling method adopted, but other options (e.g. boot, LGOCV) are available to control how models are fitted. Also, it was defined the total number of training dataset folds (number=5) and the number of models to be fitted in each fold (repeats=2). The training dataset will be divided in more folds with a small number of observations if the number of folds increase, the challenge is found the best number to avoid fit overfit models in each fold. Other functions can be used to specify other types of tuning parameters and control formats. In trControl, a complete guide containing all specifications about how to customize the tuning and control process can be found in Kuhn (2017), *section 5.5*.

```
> fitControl <- trainControl(method="repeatedCV",number=2,repeats=5)
> modelFit <- train(gap~.,data=dataTraining,method="rf",trControl=fitControl)
```

The global regression model fitted is available in the function modelFit, since the training dataset is available, the function train assemble all information previously defined, to fit regression models over different tuning parameters closely related to each technique. Excluding gap, all other variables are read as explanatory variables (gap~.). The dataset is the training set (dataTraining) obtained in the last step. The selected technique is defined in method. Load the specific technique's package is required. For the example, the selected technique was the Random forest (method=rf) from "randomForest" package.

For each technique, variations of tuning parameters fit different regression models. As the number of folds is equal to 2 and the number of models in each fold is equal to 5, a total of 10 regression models are fitted for each variation of tuning parameters. The regression model with the lowest RMSE between know response variable and the predictions is selected as a global regression model.

For a specific technique, fit regression models with the function train allows to evaluate the effect of different explanatory variables variations (varImp), which helps to choose the best regression model across these parameters and evaluate model performance from

training dataset, using the coefficient of determination (R^2) or the RMSE measures. The output values obtained about the importance of each explanatory variable is verified in modelImp. For the same dataset, the explanatory variables importance change according to each technique.

```
> modelImp <- varImp(modelFit)
> predictions <- predict(modelFit,data=dataTest)
```

With modelFit, it is possible to perform the predictions for test dataset predict and, in the next step, evaluate the model performance by the coefficient of determination R^2 and the RMSE. If R^2 is low, the model will not predict well new observations. This fact can be explained by the inefficiency of explanatory variables to describe the problem, the lack of ability of the technique to interpret the input data, or a weak dataset split process, in which training dataset have very homogeneous observations. More specifications about variables importance and how to extract predictions for new observations can be found in Kuhn (2017), *sections 14* and *section 5.7*.

The output containing all results obtained can be saved in different formats, but the most usual is the text .txt file format. In the function modelFit, the global regression model is saved, and predictions for new observations can be developed.

Actually, the "caret" package supports more than 100 techniques to fit regression models. Table A.1 summarizes the characteristics of 13 of the most important supervised machine learning techniques explored during this chapter. In function train, the techniques working associated with specific packages, as can be observed in the third column of Table A.1. For each technique, specific tuning parameters are defined. The "caret" package tests variations for tuning parameters and different regression models are obtained. As an example, for Random Forest *rf* method, *mtry* is a tuning parameter that defines the total number of predictors tested in each tree node.

An interesting investigation comparing the efficiency of data mining techniques to perform classification analysis is presented in Kotsiantis (2007). The conclusions presented are important to visualize the potential gains of one specific technique in comparison with other options before to start the regression analysis.

Figure A.7 shows all techniques available in "caret" package to fit regression models using the classification proposed in Section A.4. Most of the techniques (28) are based on ensemble concepts, not only using concepts found in Random forest, but also characteristics provided by bagging (Breiman, 1996) and boosting (Schapire, 1990) aggregating concepts. The traditional regression-based concept has 25 techniques, mainly through adaptations of the traditional splines modeling process or techniques based on linear models.

The memory-based techniques are found in the literature mainly for classifications analysis, because of the good trade-off between the computational time consumed and the quality of predictions. All the available techniques to develop regression analysis are listed in the Supplementary material at the final of this chapter, separated by each classification concept.

The last part of this section is dedicated to describing some good references found in

Technique	Function	Package	Tuning parameters
k nearest neighbors	kknn	kknn	*k*
Back-propagation neural networks	nnet	nnet	*decay*, *size*
Extreme learning machines	elm	elmNN	*nhid*, *actfun*
Bayesian regularized neural networks	brnn	brnn	*neurons*
Random forest	rf	randomForest	*mtry*
Random forest	Rborist	Rborist	*predFixed*
Stochastic gradient boosting machines	gbm	gbm, plyr	*n.trees*, *interaction.depth*, *shrinkage*, *n.minobsinnode*
Extreme gradient boosting machines	xgbTree	xgboost, plyr	*nrounds*, *max_depth*, *eta*, *gamma*, *colsample_bytree*, *min_child_weight*, *subsample*
Ridge regression	ridge	elasticnet	*lambda*
Least absolute shrinkage and selection operator	lasso	elasticnet	*fraction*
Linear model	lm		*intercept*
Multivariate adaptive regression splines	earth	earth	*nprune*, *degree*
Cubist	cubist	Cubist	*committees*, *neighbors*

kknn: Schliep and Hechenbichler; nnet: Venables et al. (1999); elmNN: Gosso and Martinez-de Pison (2012); brnn: Pérez-Rodríguez et al. (2013); randomForest: Liaw and Wiener (2002); Rborist: Seligman (2015); gbm: Ridgeway (2006); plyr: Wickham (2011); xgboost: Chen et al. (2015); elasticnet: Zou and Hastie (2012); earth: Milborrow (2014); Cubist: Kuhn et al. (2014).

Table A.1 – Supervised machine learning techniques used in train.

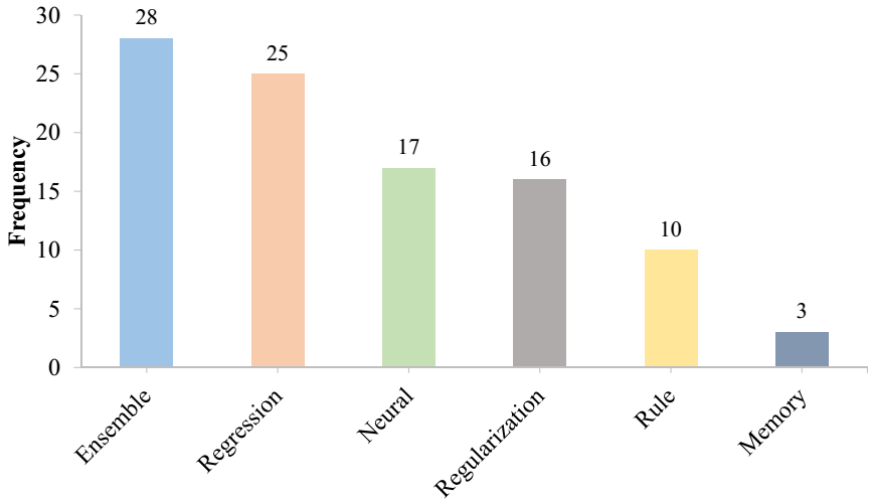


Figure A.7 – Number of techniques for each concept.

literature about techniques, concepts, and applications of the *RStudio*.

An excellent reference about data mining is the book written by Williams (2011), that describes a complete overview of different data mining applications using *Rattle* and *RStudio* software. The author proposed an extensive discussion since how to get started the first steps to use *Rattle* and *RStudio* until how to explore the raw data collected, visualize graphics and, if necessary, transform or normalize the dataset. In a second part, the process of fit models (for regression or classification analysis) using different supervised machine learning techniques is discussed. A third part is dedicated to describing some performance metrics to evaluate the quality of solutions found after fit models.

A second reference is a book written by the author of "caret" package (Kuhn and Johnson, 2013) about the process of fit and evaluate predictive models, since the pre-processing and resampling phase until the model performance measure. A case study describing a practical application of some techniques in a context of the comprehensive strength of concrete mixtures is explored. For the application steps, the development of models using *R* programming language is proposed.

Not less important are the manuals available for each technique package, explaining how each technique read and interpret the available input data and specific parameters to obtain output data. Additional instructions and a tutorial about *R* programming language, *RStudio* software, machine learning and data mining issues can be found in Venables et al. (1999), Verzani (2002), Maindonald and Braun (2006), Spector (2008), Muenchen (2011) and CRAN (2017).

A.6. Conclusion

During this chapter, an overview about the use of supervised machine learning techniques was conducted. A total of 13 techniques were described to facilitate the comprehension of how input data is performed using different techniques to fit regression models and obtain output data (the predictions).

A practical guide to supervised machine learning techniques used for regression analysis was proposed. This practical guide proposed helps the understanding of the regression analysis based on knowledge discovery (presented in section A.3) using *RStudio* software. This methodological knowledge discovery framework can also be computed by another software, as *Python* or *Rattle*. This practical guide using *R* and *RStudio* is a generic tool, that can be easily adapted to be applied to different problems.

Future work will describe in more details techniques to fit regression models not explored in this study, using techniques with more potential to fit generic regression models without loss the predictions' accuracy. The discussion covering the 13 of the most important supervised machine learning techniques can be expanded and improved. Another possibility to be explored in future works is the understanding and description of practical guides using different software, that can perform regression analysis.

References

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4(1):40–79.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). CRISP-DM 1.0 step-by-step data mining guide.

Chen, B., Wang, Y., and Yang, S. (2015). A hybrid demon algorithm for the two-dimensional orthogonal strip packing problem. *Mathematical Problems in Engineering*, 2015(1):1–14.

Cox, D. R. (2006). *Principles of statistical inference*, volume 1. Cambridge University Press.

CRAN (2017). CRAN: Available packages.

Daelemans, W. and Van den Bosch, A. (2005). *Memory-based language processing*, volume 1. Cambridge University Press.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30.

Devroye, L. and Wagner, T. (1979). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604.

Eugster, M. J., Hothorn, T., and Leisch, F. (2008). Exploratory and inferential analysis of benchmark experiments. *Technical report*, 30(1):1–28.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park.

Foresee, F. D. and Hagan, M. T. (1997). Gauss-Newton approximation to Bayesian learning. In *International Conference on Neural Networks*, volume 3, pages 1930–1935. IEEE.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

Gama, J., Carvalho, A. C. P. d. L., Faceli, K., Lorena, A. C., and Oliveira, M. (2015). *Extração de conhecimento de dados: Data mining*, volume 2. Edições Silabo.

Gosso, A. and Martinez-de Pison, F. (2012). elmNN: Implementation of ELM (Extreme Learning Machine) algorithm for SLFN (Single Hidden Layer Feedforward Neural Networks). *R Package Version*, 1(3):1–10.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674.

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1):489–501.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica*, 31(1):249–268.

Kuhn, M. (2008). Caret package. *Journal of Statistical Software*, 28(5):1–26.

Kuhn, M. (2017). The caret package.

Kuhn, M. and Johnson, K. (2013). *Applied predictive modeling*, volume 1. Springer.

Kuhn, M., Weston, S., Keefer, C., Coulter, N., and Quinlan, R. (2014). Cubist: Rule-and instance-based regression modeling. *R Package Version*, 1(3):1–13.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R news*, 2(3):18–22.

Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528.

López-Camacho, E., Terashima-Marín, H., and Ross, P. (2010). Defining a problem-state representation with data mining within a hyper-heuristic model which solves 2D irregular bin packing problems. In *Ibero-American Conference on Artificial Intelligence*, pages 204–213. Springer.

MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

Maindonald, J. and Braun, J. (2006). *Data analysis and graphics using R: An example-based approach*, volume 10. Cambridge University Press.

Milborrow, S. (2014). Earth: Multivariate adaptive regression spline models. *R Package Version*, 3(1):1–49.

Muenchen, R. A. (2011). *R for SAS and SPSS users*, volume 1. Springer Science & Business Media.

Pérez-Rodríguez, P., Gianola, D., Weigel, K., Rosa, G., and Crossa, J. (2013). An r package for fitting bayesian regularized neural networks with applications in animal breeding. *Journal of Animal Science*, 91(8):3522–3531.

Ridgeway, G. (2006). Gbm: Generalized boosted regression models. *R Package Version*, 1(3):1–34.

RStudio (2017). Rstudio software.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Schliep, K. and Hechenbichler, K. Kknn: Weighted k-nearest neighbors. *R Package Version*, 1(3):1–15.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61(1):85–117.

Seligman, M. (2015). Rborist: Extensible, parallelizable implementation of the random forest algorithm. *R Package Version*, 1(3):1–15.

Smith-Miles, K., Baatar, D., Wreford, B., and Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45(1):12–24.

Spector, P. (2008). *Data manipulation with R*, volume 1. Springer Science & Business Media.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Ticknor, J. L. (2013). A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 40(14):5501–5506.

Venables, B., Ripley, B., Hornik, K., and Gebhardt, A. (1999). VR: Bundle of mass, class, nnet, spatial. *R Package Version*, 1(3):2–7.

Verzani, J. (2002). SimpleR - using R for introdutory statistics.

Wang, Y., Li, J., Li, Y., Wang, R., and Yang, X. (2015). Confidence interval for f_1 measure of algorithm performance based on blocked $3x2$ cross-validation. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):651–659.

Weiss, S. M. and Indurkhya, N. (1995). Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3(1):383–403.

Wickham, H. (2011). The split-apply-combine strategy for data analysis. *Journal of Statistical Software*, 40(1):1–29.

Williams, G. (2011). *Data mining with Rattle and R: The art of excavating data for knowledge discovery*. Springer Science & Business Media.

Wirth, R. and Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39.

Yildiz, O. T. and Alpaydin, E. (2006). Ordering and finding the best of k>2 supervised learning algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):392–402.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Zou, H. and Hastie, T. (2012). Elastic-Net for sparse estimation and sparse PCA. *R Package Version*, 1(3):1–14.

Tuning parameters

Technique	Tuning parameter	Definition
k-nearest neighbors	*k*	Number of neighbors
Backpropagation neural networks	*decay*	Weight decay
Backpropagation neural networks	*size*	Number of units in the hidden layer
Extreme learning machines	*nhid*	Number of hidden neurons
Extreme learning machines	*actfun*	Type of activation function
Bayesian regularized neural networks	*neurons*	Number of neurons
Random forest	*mtry*	Number of trial predictors for a split
Random forest	*predFixed*	Number of trial predictors for a split
Stochastic gradient boosting machines	*n.trees*	Number of trees used to generate the plot
Stochastic gradient boosting machines	*interaction.depth*	Maximum depth of variable interactions (additive model, 2-way interactions, ...)
Stochastic gradient boosting machines	*shrinkage*	Learning rate or step-size reduction applied to each tree in the expansion
Stochastic gradient boosting machines	*n.minobsinnode*	Minimum number of observations in the trees terminal nodes
Extreme gradient boosting machines	*nrounds*	Maximum number of iterations
Extreme gradient boosting machines	*max_depth*	Maximum depth of a tree
Extreme gradient boosting machines	*eta*	Control the learning rate: scale the contribution of each tree
Extreme gradient boosting machines	*gamma*	Minimum loss reduction required to make a further partition on a leaf node of the tree
Extreme gradient boosting machines	*colsample_bytree*	Subsample ratio of columns when constructing each tree
Extreme gradient boosting machines	*min_child_weight*	Minimum sum of instance weight (hessian) needed in a child
Extreme gradient boosting machines	*subsample*	Ratio of the training instance
Ridge regression	*lambda*	Quadratic penalty parameter
Least absolute shrinkage and selection operator	*fraction*	Values of abscissa values at which cross-validated curve should be computed
Linear model	*intercept*	
Multivariate adaptive regression splines	*nprune*	Maximum number of terms (including intercept) in the pruned model
Multivariate adaptive regression splines	*degree*	Maximum degree of interaction
Cubist	*committees*	Number of committee models (e.g. boosting iterations) should be used
Cubist	*neighbors*	Number of neighbors

Table A.2 – Tuning parameters definition.

Supervised machine learning techniques

Technique	Function	Type[a]	Libraries
Bagged CART	treebag	C&R	ipred, plyr, e1071
Bagged Logic Regression	logicBag	C&R	logicFS
Bagged MARS	bagEarth	C&R	earth
Bagged MARS using gCV Pruning	bagEarthGCV	C&R	earth
Bagged Model	bag	C&R	caret
Boosted Generalized Additive Model	gamboost	C&R	mboost, plyr
Boosted Generalized Linear Model	glmboost	C&R	plyr, mboost
Boosted Linear Model	BstLm	C&R	bst, plyr
Boosted Smoothing Spline	bstSm	C&R	bst, plyr
Boosted Tree	blackboost	C&R	party, mboost, plyr
Boosted Tree	bstTree	C&R	bst, plyr
Conditional Inference Random Forest	cforest	C&R	party
Ens. of Generalized Linear Models	randomGLM	C&R	randomGLM
eXtreme Gradient Boosting	xgbLinear	C&R	xgboost
eXtreme Gradient Boosting	xgbTree	C&R	xgboost, plyr
Extreme Learning Machine	elm	C&R	elmNN
Gradient Boosting Machines	gbm_h2o	C&R	h2o
Parallel Random Forest	parRF	C&R	e1071, randomForest, foreach
Random Forest	ranger	C&R	e1071, ranger
Random Forest	Rborist	C&R	Rborist
Random Forest	rf	C&R	randomForest
Random Forest by Randomization	extraTrees	C&R	extraTrees
Random Forest Rule-Based Model	rfRules	C&R	randomForest, inTrees, plyr
Regularized Random Forest	RRF	C&R	randomForest, RRF
Regularized Random Forest	RRFglobal	C&R	RRF
Stochastic Gradient Boosting	gbm	C&R	gbm, plyr
Tree-Based Ensembles	nodeHarvest	C&R	nodeHarvest
Quantile Random Forest	qrf	R	quantregForest

[a] C&R: Technique works for both classification and regression analysis. R: Technique works only for regression analysis.

Table A.3 – Ensemble learning-based techniques.

Technique	Function	Type[a]	Libraries
Model Averaged Neural Network	avNNet	C&R	nnet
Monotone Multi-Layer Perceptron Neural Network	monmlp	C&R	monmlp
Multi-Layer Perceptron	mlp	C&R	RSNNS
Multi-Layer Perceptron	mlpWeightDecay	C&R	RSNNS
Multi-Layer Perceptron, multiple layers	mlpWeightDecayML	C&R	RSNNS
Multi-Layer Perceptron, with multiple layers	mlpML	C&R	RSNNS
Multi-Step Adaptive MCP-Net	msaenet	C&R	msaenet
Multilayer Perceptron Network by Stochastic Gradient Descent	mlpSGD	C&R	FCNN4R, plyr
Neural Network	nnet	C&R	nnet
Radial Basis Function Network	rbf	C&R	RSNNS
Radial Basis Function Network	rbfDDA	C&R	RSNNS
Stacked AutoEncoder Deep Neural Network	dnn	C&R	deepnet
Bayesian Regularized Neural Networks	brnn	R	brnn
Dynamic Evolving Neural-Fuzzy Inference System	DENFIS	R	frbs
Hybrid Neural Fuzzy Inference System	HYFIS	R	frbs
Neural Network	neuralnet	R	neuralnet
Quantile Regression Neural Network	qrnn	R	qrnn

[a] C&R: Technique works for both classification and regression analysis. R: Technique works only for regression analysis.

Table A.4 – Neural networks-based techniques.

Technique	Function	Type[a]	Libraries
Bayesian Generalized Linear Model	bayesglm	C&R	arm
Gaussian Process	gaussprLinear	C&R	kernlab
Gaussian Process with Polynomial Kernel	gaussprPoly	C&R	kernlab
Gaussian Process with Radial Basis Function Kernel	gaussprRadial	C&R	kernlab
Generalized Additive Model using LOESS	gamLoess	C&R	gam
Generalized Additive Model using Splines	bam	C&R	mgcv
Generalized Additive Model using Splines	gam	C&R	mgcv
Generalized Additive Model using Splines	gamSpline	C&R	gam
Generalized Linear Model	glm	C&R	
Logic Regression	logreg	C&R	LogicReg
Multivariate Adaptive Regression Spline	earth	C&R	earth
Multivariate Adaptive Regression Splines	gcvEarth	C&R	earth
Penalized Ordinal Regression	ordinalNet	C&R	ordinalNet, plyr
Least Angle Regression	lars	R	lars
Least Angle Regression	lars2	R	lars
Linear Regression	lm	R	
Linear Regression with Backwards Selection	leapBackward	R	leaps
Linear Regression with Forward Selection	leapForward	R	leaps
Linear Regression with Stepwise Selection	leapSeq	R	leaps
Linear Regression with Stepwise Selection	lmStepAIC	R	MASS
Negative Binomial Generalized Linear Model	glm.nb	R	
Penalized Linear Regression	penalized	R	penalized
Projection Pursuit Regression	ppr	R	
Robust Linear Model	rlm	R	MASS
Spike and Slab Regression	spikeslab	R	spikeslab, plyr

[a] C&R: Technique works for both classification and regression analysis. R: Technique works only for regression analysis.

Table A.5 – Regression-based techniques.

Technique	Function	Type[a]	Libraries
glmnet	glmnet	C&R	glmnet, Matrix
glmnet	glmnet_h2o	C&R	h2o
L2 Regularized Support Vector Machine (dual) with Linear Kernel	svmLinear3	C&R	LiblineaR
Bayesian Ridge Regression	bridge	R	monomvn
Bayesian Ridge Regression (Model Averaged)	blassoAveraged	R	monomvn
Elasticnet	enet	R	elasticnet
Non-Convex Penalized Quantile Regression	rqnc	R	rqPen
Non-Negative Least Squares	nnls	R	nnls
Polynomial Kernel Regularized Least Squares	krlsPoly	R	KRLS
Quantile Regression with LASSO penalty	rqlasso	R	rqPen
Radial Basis Function Kernel Regularized Least Squares	krlsRadial	R	KRLS, kernlab
Relaxed LASSO	relaxo	R	relaxo, plyr
Ridge Regression	ridge	R	elasticnet
Ridge Regression with Variable Selection	foba	R	foba
The Bayesian lASSO	blasso	R	monomvn
The LASSO	lasso	R	elasticnet

[a] C&R: Technique works for both classification and regression analysis. R: Technique works only for regression analysis.

Table A.6 – Regularization-based techniques.

Technique	Function name	Type[a]	Libraries
k-Nearest Neighbors	kknn	C&R	kknn
k-Nearest Neighbors	knn	C&R	
Knn regression via sklearn. neighbors. KNeighborsRegressor	pythonKnnReg	R	rPython

[a] C&R: Technique works for both classification and regression analysis. R: Technique works only for regression analysis.

Table A.7 – Memory-based techniques.

Technique	Function	Type[a]	Libraries
Cubist	cubist	R	Cubist
Fuzzy Inference Rules by Descent Method	FIR.DM	R	frbs
Fuzzy Rules via MOGUL	GFS.FR.MOGUL	R	frbs
Fuzzy Rules via Thrift	GFS.THRIFT	R	frbs
Genetic Lateral Tuning and Rule Selection of Linguistic Fuzzy Systems	GFS.LT.RS	R	frbs
Model Rules	M5Rules	R	RWeka
Model Tree	M5	R	RWeka
Simplified TSK Fuzzy Rules	FS.HGD	R	frbs
Wang and Mendel Fuzzy Rules	WM	R	frbs
Adaptive-Network-Based Fuzzy Inference System	ANFIS	R	frbs

[a] C&R: Technique works for both classification and regression analysis. R: Technique works only for regression analysis.

Table A.8 – Rule-based techniques.