FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Optimization Algorithms for the Shelf Space Allocation Problem

Ana Carolina Reis Janeiro

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: José Fernando da Costa Oliveira

Second Supervisor: Maria Teresa Peixoto Braga Bianchi de Aguiar

July 31, 2014

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"Optimization Algorithms for the Shelf Space Allocation Problem"

foi aprovada em provas realizadas em 23-07-2014

o júri

Presidente Professor Doutor Fernando Manuel Ferreira Lobo Pereira
Professor Catedrático do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Maria Helena Gonçalves da Silva Correia
Professora Auxiliar da Faculdade de Economia e Gestão da Universidade Católica Portuguesa

Professor Doutor José Fernando da Costa Oliveira
Professor Catedrático do Departamento de Engenharia de Gestão Industrial da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Ana Carolina Reis Janeiro

Faculdade de Engenharia da Universidade do Porto

Abstract

In retail stores, shelves have a limited capacity to place products. Due to this reason, it is extremely important to manage them carefully.

According to experimental studies, there are many in-store factors that can influence costumer purchases. Among these are the number of facings of a product (ie. visible items), its position on shelves (horizontal and vertical), the price and product adjacencies. To improve sales it is relevant for retailers to capitalize on these factors by optimizing the allocation of products on shelves.

Although there are software applications available to help in shelf space planning, these still have a number of limitations and require significant human interaction. On the other hand, this problem has also been tackled in the literature under the name *Shelf Space Allocation Problem*. However, models and solution approaches can hardly be adapted to reality as they usually lack key practical constraints and require parameters hard to estimate.

Inspired by the case of a Portuguese Supermarket Chain, two novel Biased Random-key Genetic Algorithms are proposed to tackle the Shelf Space Allocation Problem. The algorithms aim to allocate the products on shelves bearing in mind the best location and number of facings to boost their sales. Additionally, products are allocated within families that should form rectangular shapes on the shelves. The main difference between both algorithms lies in the way these families of products are placed on the shelves.

The algorithms are tested with real case study instances. Among the key benefits of both approaches is their applicability in practice, both in terms of constraints and execution times.

Resumo

Nas lojas a retalho, as prateleiras têm uma capacidade limitada para alocar produtos. Por essa razão, é necessário geri-las de uma forma cuidada.

De acordo com estudos experimentais, há vários fatores, dentro da loja, que influenciam as compras de clientes. Entre estes consta o número de frentes de um produto (ou seja, os itens visíveis), a sua posição nas prateleiras (horizontal e vertical), o preço e as adjacências de produtos. Para melhorar as vendas, é importante para os retalhistas lucrar com estes fatores, optimizando a alocação de produtos nas prateleiras.

Apesar de existirem aplicações de software disponíveis para ajudar no planeamento do espaço das prateleiras, estas ainda têm algumas limitações e requerem uma interação humana significativa. Por outro lado, este problema já foi tratado na literatura, com o nome *Problema de Alocação de Espaço em Prateleiras*. Apesar disso, os modelos e estratégias de solução são pouco adaptáveis à realidade, uma vez que, normalmente, não consideram restrições práticas chave e requerem parâmetros difíceis de estimar.

Inspirados pelo caso de uma cadeia portuguesa de supermercados, dois novos Algoritmos Genéticos de Chaves Viciadas são propostos para resolver o Problema de Alocação de Espaço em Prateleiras. Os algoritmos têm como objetivo alocar os produtos nas prateleiras, tendo em conta a melhor localização e o número de frentes, de forma a aumentar as vendas. Além disso, os produtos são alocados em famílias que apresentam formas retangulares nas prateleiras. A maior diferença entre os dois algoritmos reside na forma como as famílias de produtos são alocadas nas prateleiras.

Os algoritmos são testados com instâncias de um caso de estudo real. Entre os benefícios chave das duas estratégias está a sua aplicação na realidade, tanto em termos de restrições como tempo de execução.

Acknowledgments

I would like to thank my supervisor, Professor José Fernando Oliveira, for giving me the opportunity to work in such an interesting project and for integrating me in the Operations Research world. Above all, I would like to thank my co-supervisor, Teresa Bianchi de Aguiar, who always helped me in the development of this dissertation and pushed me to do my best. She also gave up a lot of her time so that the final result would meet the objectives. To her, my sincere thank you. I also owe my thanks to Elsa Silva and Luís Guimarães who helped improve the ideas for the project and were there whenever it was necessary. I would also like to thank my friends for always giving me their help and support, even through hard times. Finally, my special thanks goes to my family, particularly my parents and my sister, who have always believed in me and have always done everything they could to see me happy.

Ana Carolina Reis Janeiro

"Knowledge isn't power until it is applied."

Dale Carnegie

Contents

List of Figures

List of Tables

Abbreviations and Symbols

SSAP	Shelf Space Allocation Problem
BRKGA	Biased Random-key Genetic Algorithm
APED	Portuguese Association of Distribution Companies
2D-BPP	Two-dimensional Bin Packing Problem
3D-BPP	Three-dimensional Bin Packing Problem
GA	Genetic Algorithm
EA	Evolutionary Algorithm
BL	Bottom-Left
LB	Left-Bottom
ERS	Empty Rectangular Space
DP	Difference Process

Chapter 1

Introduction

In this chapter the motivation and aims of this dissertation are explained. Furthermore, the organization of chapters is described.

1.1 Motivation

Shelf space is a very important and limited resource of every retail store. In-store factors including number of facings (i.e. visible items), horizontal and vertical shelf position and price can affect costumers attention and evaluation of products when making a purchase [7]. This attractiveness of the layout is particularly relevant in "out-of-stock" situations or in cases of unplanned purchases [8]. Based on these reasons, it is possible to conclude that retail stores with optimized shelf space management may see improvements in their financial performance. In the extremely competitive retail market, retailers strive to display products on the shelves in such a way that will translate in a boost in sales, by finding the right number of facings for each product and placing it in the best location.

Retailers have software applications available that provide tools to help manage shelf space more efficiently. In spite of this, there are still limitations in these applications, which require significant human interaction [8].

The problem of allocating products on shelves is tackled in the literature under the name Shelf Space Allocation Problem. However, the existing mathematical models and solution methods lack key practical constraints and have parameters hard to estimate. Therefore, there still exists a strong margin for improvement until these approaches can have a real application.

1.2 Objectives

This dissertation aims to study the possibility of using metaheuristics to tackle the shelf space allocation problem and bridge the gap between theory and practice. In particular, the goal is to apply a Biased Random-key Genetic Algorithm. Although many different approaches have been used to tackle the Shelf Space Allocation Problem, these algorithms have not yet been tested. In

spite of this, there is evidence in literature that they are successful in solving similar problems, particularly Cutting and Packing Problems [4].

Inspired by the case study of a major Portuguese Retail company, the problem is defined according to the reality faced by the company, giving to this dissertation the necessary proximity to the practice of shelf space allocation. To check its applicability, the algorithm is tested with real-world instances provided by the case study.

1.3 Structure

This dissertation is organized in six chapters. The first of these chapters corresponds to the introduction, in which the motivation and objectives of the work developed are explained. The remaining chapters are organized as follows:

In chapter 2, *Shelf Space Management*, the problem under analyzes is presented and current practices in retail are described. Furthermore, other related problems are also reviewed and a short description of the case study is made.

In chapter 3, *Literature Review*, experimental studies that have been conducted to analyze the impact of shelf space on sales are reviewed. Afterwards, the mathematical models and optimization methods present in the literature to help retailers in their decisions concerning shelf space allocation are also studied. Finally, Genetic Algorithms and, particularly, Biased Random-key Genetic Algorithms and their application in Shelf Space management and related problems are analyzed.

In chapter 4, *Biased Random-key Genetic Algorithm for the Shelf Space Allocation Problem*, two original solution decoders are presented for applying Biased Random Key Genetic Algorithms to the Shelf Space Allocation Problem.

In chapter 5, *Tests and Results*, real case study instances are tested and the results obtained are described.

In chapter 6, *Conclusion and Future Work*, the knowledge obtained throughout the development of the dissertation is presented and suggestions for future work are made.

Chapter 2

Shelf Space Management

In this chapter the importance of shelf space management is described. For that purpose, current practices concerning processes and software applications are reviewed. Afterwards, the Shelf Space Allocation Problem is analyzed in detail and other related problems are briefly explained. Finally, a description of the case study under analysis is presented.

2.1 Introduction

Retail is an extremely competitive industry in which retailers look for every possible advantage they can get. An example of this can be found in the two biggest Portuguese food retailers that detained the biggest volume of business in 2011, according to the Portuguese Association of Distribution Companies (APED) ranking (Continente and Pingo Doce) [9]. In recent years, both companies have resorted to aggressive marketing campaigns to compete for costumers purchases.

In order to face the fierce competition, retailers strive to boost their sales by every means possible. Although out-of-store media advertising and loyalty to a product can influence costumers purchases, there are situations in which in-store factors are crucial in leading a consumer to buy a product. In particular, when the store does not have the product preferred by the costumer in stock or in cases of unplanned purchases, the store's layout and space planning can boost the attractiveness of the products and improve their sales [8].

According to [7], the number of facings of a product, its price and shelf position (horizontal and vertical) are examples of in-store factors that may influence attention and evaluation of costumers at the point of purchase. Studies show that increasing the total shelf space has a strong effect on sales, but the available shelf space is limited and increasing it requires a high investment both in terms of construction and maintenance, which the retailer might not be willing to make. Furthermore, the location of a product has an important impact on sales as well as maintaining a minimum number of facings on shelves, in order to avoid stockouts [10]. Particularly, top and eye-levels positions of products on shelves are considered to have a stronger influence in consumers attention.

Retailers use diagrams to represent the location and number of facings of the products. These diagrams are called planograms [7]. An example of a planogram is presented in Figure 2.1.

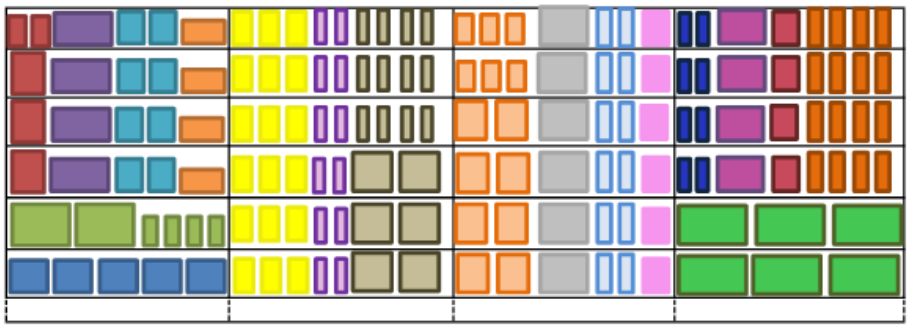

Figure 2.1: Example of a planogram.

New products are constantly being introduced and competing for shelf space in retail stores, requiring shelf displays to be frequently changed. According to [1], the average number of items, between 2000 and 2009, increased by 30% in overall store assortments. In particular, in the food retail industry the number of products to be displayed in supermarket shelves is, in general, very large and demands a careful management. To do this, large amounts of time have to be spent, particularly if planograms are developed with very significant human interaction. As a result, most retailers suffer from decreasing space productivity [1]. If the shelf layout is well planned, stores can see an improvement in their financial performance and reduce the probability of products being out of stock [11].

In this context, efficient methods for allocating products in shelves can bring an important advantage to companies, giving them the upper hand in today's extremely competitive market.

2.2 Current Practices

2.2.1 Problems in Retail

When planning the layout of a store there are different factors to take into account, including which products to offer, how many facings should be allocated, where to place them and their restocking frequencies. These factors are interrelated since a large number of products will implicate that less space will be allocated to each. As a consequence, items will be less visible on shelves and the risk of stockouts will increase. According to [1], these questions correspond to different problems of master category planning, whose interdependencies and planning horizons are presented in figure 2.2.

- Category sales planning defines the major categories in which products should be organized in the stores and their corresponding role and depth. This type of planning is strategic and is usually done for a longer horizon.

- Assortment planning consists in deciding which products should be present in each category, based on the consumer choices and the substitution effects that happen when a product is not present or becomes unavailable.
- Shelf Space Planning determines products allocation on shelves, based on the limited capacity, space effects on demand and operational restocking constraints. This kind of planning is generally done for a shorter horizon, together with assortment planning.
- In-store logistics planning is the most operational problem and defines the replenishment planning for each store.

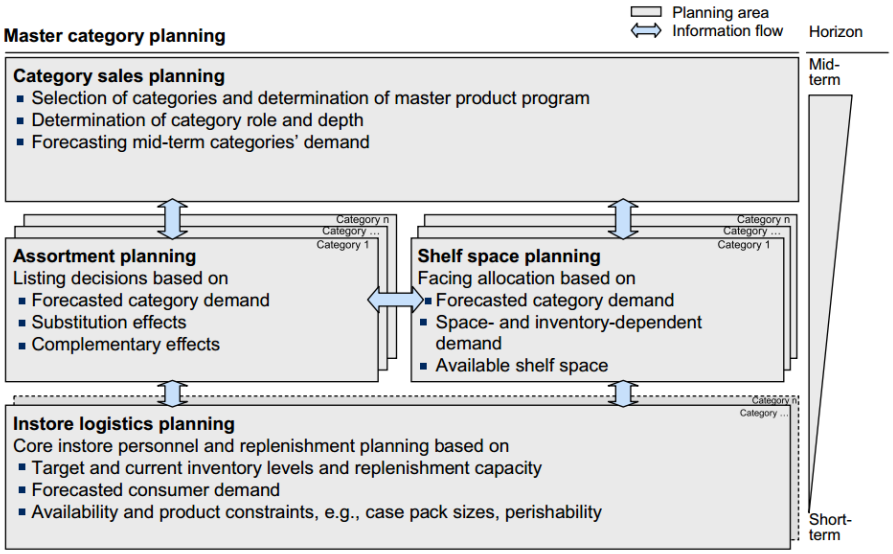

Fig. 1. Interdependencies in master category planning.

Figure 2.2: Interdependencies in master category planning.[1]

Since retailers deal with a large number of products, they started to group the ones with similar characteristics into categories in the late 1980s [12]. To better manage these categories, many retailers and suppliers started to name leading manufacturers as category captains. Their function is to give insight into the best way of organizing a particular category. In spite of the helpful information, there are concerns that a category captain might be biased. The reason for this lies in the fact that category captains have to make recommendations concerning not only their own brands, but rival brands as well.

2.2.2 Software Applications for Shelf Space Management

Category planning can potentially involve large amounts of products. As such it can be an extremely complex problem. To help retailers in their tasks there are commercial software solutions available. Despite this, according to [1], in 2009, 30% still did not use any kind of IT support to manage shelf space, and only 13%, planned to invest on it (Figure 2.3).

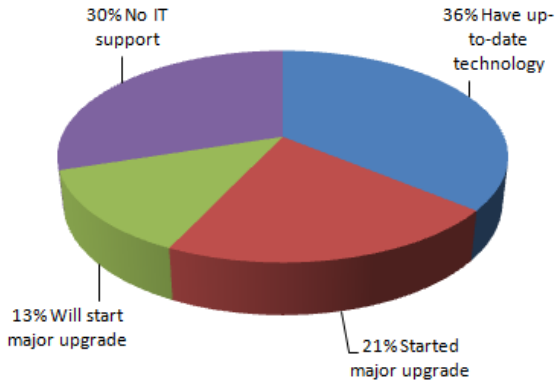

Figure 2.3: Information concerning IT usage to manage shelf space by retailers [1].

The existing software for assortment and shelf space management is poorly interrelated although in reality both problems are connected. Concerning shelf planning software, the applications available present several advantages, providing realistic views of the shelves, data analysis tools and the ability to quickly handle products on shelves. In spite of this, these applications have limitations. This happens because software developers focus more on processing large quantities of data than on obtaining efficient algorithms to help the process of decision making. In these applications, the automatic allocation of products in shelves is based on simple rules related to the product's market share, sales, profit or a combination of these factors [8]. This leads to the necessity of manual adjustments done by the user which can be considerably time consuming. As a result, planograms are most of the time generated manually.

Examples of software applications for space management include solutions from top software vendors such as AC Nielson (Spaceman), JDA (Space Planning), MEMRB/IRI (Apollo professional) and SAS Institute (SAS Space Planning).

2.3 The Shelf Space Allocation Problem

The Shelf Space Allocation Problem consists of allocating the right amount of items to each product (also known as stock keeping unit - SKU) in the best position, taken into consideration the limited shelf space available on a retail store. It is a very complex problem that bears a similarity to Cutting and Packing problems, which, in general, are NP-hard.

The variety of products to be placed on shelves is usually very large, particularly in supermarkets, which are the aim of this dissertation. Solving the problem for all products would require very high computational requirements. Due to this reason, products are usually grouped into categories, depending on their functional characteristics. The problem is then solved for each category.

The Shelf Space Allocation Problem may vary widely depending on the company under consideration. This happens because many factors can change from one company to another, including the dimension of stores, the preferred strategy, the agreements with vendors, the criteria used for the store's layout, among others. Therefore, it is difficult to develop a definition of the Shelf Space

Allocation Problem that would apply to all cases. This section will be focused on the definition that applies to the case study, although always having in mind other realities.

2.3.1 Decisions

The Shelf Space Allocation Problem aims to determine the number of facings and their corresponding location for all the products of a category. The number of facings corresponds to the quantity of a product that can be seen on a shelf (Figure 2.4). Behind it are more units of the product, representing the depth of the facings. Since the depth of each facing is not directly seen, it does not influence the demand function [8] and can usually be ignored. In most cases, the height of the facings does not have to be considered as well, because the height of the shelves is adjustable. This leads to a problem in which it is necessary to find the number of facings, the horizontal position and the shelf where each product will be placed.

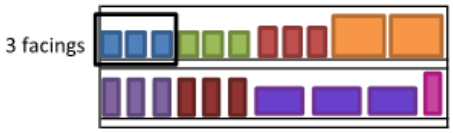

Figure 2.4: Facings of products on a shelf

2.3.2 Objectives

According to [10], the objective of the Shelf Space Allocation Problem depends on whether the point of view of the retailer or the manufacturer is considered. On the one hand retailers aim to maximize category sales and profits without regarding any brand in particular. On the other hand, manufacturers' goal is to improve sales of their own brands and, therefore, want to allocate as much space as possible to their own products. In both points of view, sales are maximized based on the effect of space, namely space and location elasticities.

Another goal that might be considered is the reduction of costs. However, as pointed by [8], if a product's demand depends on the decision variables, by minimizing the cost the model might reduce the number of facings. As a result, a reduction in sales and profit would be experienced.

This dissertation focuses on the retailers point of view and considers the objective of allocating shelf space in such a way that will maximize profit or sales for the retailer.

2.3.3 Constraints

The goal of the Shelf Space Allocation Problem is subject to many possible constraints. These can vary widely depending on the retailer under consideration, although some of them are commonly considered by most. Since this dissertation takes into account the point of view of a particular case study, their specific constraints will be referred:

- Integrality constraints: Product facings are integers and cannot be divided to fit available shelf space;
- Physical constraints: products allocated to a shelf cannot exceed its physical capacity. Another aspect to take into consideration arises when consecutive segments of shelves are not aligned and therefore products cannot transpose to the adjacent segment;
- Control constraints: when allocating products, a minimum and maximum number of facings is usually established. This is done in order to guarantee the necessary exposure and, at the same time, control inventory and replenishment costs and avoid stockouts, by keeping facings within reasonable limits. According to [10], "an improper location or an under-allocation of space might kill a product before it achieves full sales potential. And retailers work hard to maximize return on their investment: allocating too many facings is a waste, while allocating too few will result in lost sales due to out of stocks.";
- Family Block constraints: retailers group products in families according to a variety of characteristics [13] . These characteristics include, for example, brand, subcategory, color, size, among other criteria. Retailers believe that these families should form rectangular shapes on shelves in order to increase the shelf layout's attractiveness. In the case study, it is also believed that family blocks belonging to the same criteria should be placed either in uniform and complete columns (see Figure 2.1) or in complete lines within the criteria that contains them. There is also a preference for facings of the same product to be placed horizontally on the shelves. Small deviations can be allowed.

There is the necessity to distinguish between hard constraints which have to be respected or the solution will be infeasible and soft constraints which preferably would be respected, but can have small deviations. The constraints referred above are organized from the hardest to the softest [2].

The incorporation of all these constraints results in a very complex problem, although some assumptions can be made. In typical shelf space management problems it is assumed that the product assortment within a category has already been made [14]. Due to this, the problem consists of finding the number of facings and location of products already assorted within a category, respecting, as much as possible, the constraints presented and maximizing the profit or sales for the retailer.

2.4 Other Related Problems

The Shelf Space Allocation Problem bares resemblances to other problems in literature, namely Cutting and Packing Problems. According to [15], given a set of large objects (input, supply) and a set of small items (output, demand), the objective is to place some or all small items entirely within one or more large object, without overlapping. Similarly, in the Shelf Space Allocation Problem there are small items (products) that need to be placed on large objects (shelves). The resemblance

to Cutting and Packing Problems is particularly noticeable if the only restrictions considered are the integrality of products and the capacity of shelves. In general, Cutting and Packing Problems are considered NP-hard.

Examples of similar packing problems are the Bin Packing Problem and the Knapsack Problem [8] [16].

In the Bin Packing Problem, there is a number of items, each with a corresponding size or weight and a number of bins with a certain capacity. The goal is to use the smallest number of bins possible to pack all the items, while respecting the capacity of the bins. This problem can be extended to two-dimensional (2D-BPP) and three-dimensional bin packing (3D-BPP). In the 3D-BPP, a set of three-dimensional rectangular shaped boxes is packed with no overlap into the minimum number of three-dimensional rectangular shaped bins. All the bins have identical dimensions and each box has a particular dimension. An example of the 3D-BPP can be found in Figure 2.5. The 2D-BPP can be treated as a special case of 3D-BPP in which the depth of the bin is considered equal to the depth of each box [17].

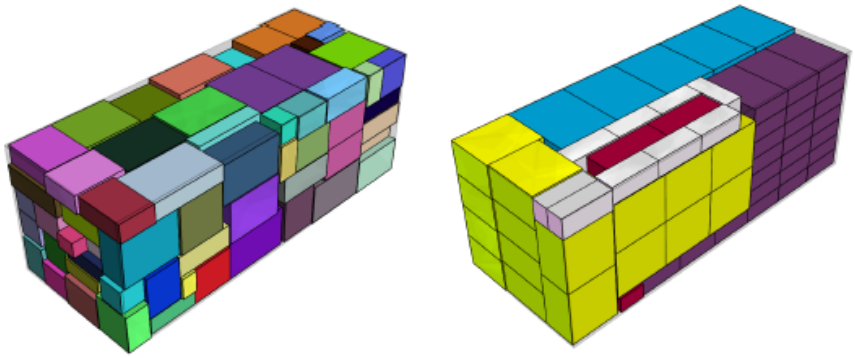

Figure 2.5: Example of Three-dimensional Bin Packing Problem [17]

On the other hand, there are different types of Knapsack Problems. The most commonly studied Knapsack Problem is the 0-1 [8]. In this problem, there is a number of items, similarly to the Bin Packing Problem, with a corresponding weight and profit. The objective is to choose the items to be placed in the knapsack, such that the total profit is maximized. Other types of Knapsack Problems include the Bounded Knapsack Problem that differs from the 0-1 version in the fact that the number of items of the same kind can vary within a specific range whereas in the 0-1 type the variable representing each item can only take the values 0 or 1. The 0-1 Multiple Knapsack Problem is another problem that is similar to the 0-1, but in which instead of only one knapsack there is a set.

Concerning Cutting Problems, there are also examples that bare a resemblance to the Shelf Space Allocation Problem, such as the Two-dimensional Non-guillotine Orthogonal Cutting Problem. This problem consists of cutting rectangular pieces from a large rectangular material, so that

the value of the rectangles cut is maximized [5]. In this special case, it is considered that items cannot be rotated and when cut, their edges must be parallel to the edges of the larger rectangular material.

There are examples in the literature where the formulation of the Shelf Space Allocation Problem was based on the Knapsack Problem formulation [8]. In spite of this, in reality, there are significant differences between the problems, since the Shelf Space Allocation Problem takes into consideration additional aspects such as the product's family blocks and impacts such as space elasticity.

Chapter 3

Literature Review

In this chapter, the work previously developed concerning the Shelf Space Allocation Problem is analyzed. Research has been focused on two major streams: experimental studies and optimization methods. Experimental studies have been conducted with the aim to understand and measure the effect of a store's display on costumers purchases whereas optimization methods seek to help retailers in the process of allocating products on shelves. The Biased Random-key Genetic Algorithm is further reviewed in this chapter as this dissertation focuses on its use to tackle the Shelf Space Allocation Problem.

3.1 Experimental Studies

In a retail store, there are many visual stimuli competing for a costumer's attention. Experimental studies try to explain how these stimuli, more specifically those related with space, may affect costumer purchases (e.g. [10] [7] [18]). In general, results show that space variables have a positive effect on demand [2]. In spite of this, due to the high costs of implementing controlled experiments, the knowledge concerning the impact of shelf space allocation on sales is limited [10]. Three main space related factors have been consistently reported: product location (vertical and horizontal), number of facings of a product and products adjacency.

Product Location

Retailers believe that products placed at eye-level shelves have a stronger effect on sales [18]. An eye tracking experiment conducted in [10] validated this belief as results showed that the vertical location of a product had a large impact on sales, with an increase of 39% from the worst to the best vertical location on shelves. Concerning horizontal location, the impact on sales was not as strong as vertical location with a 15% increase from the worst to the best horizontal location. The best horizontal location depends on the retailers perspective, but is generally considered to be around the horizontal center of a shelf.

Number of facings (defined as space elasticity)

A measurement used to analyze the impact of space variation on sales is space elasticity. Space elasticity is defined as the "ratio of relative change in unit sales to relative change in shelf space" [19]. Experiments have concluded that as space allocated to a product increases, the marginal returns will increase at first and then decrease in a S-shaped curve (Figure 3.1) [20]. Space elasticity varies depending on the products category and store layout [20] [2]. However, in [18] the space elasticity of all products is on average 0.212, while in [10] is considered to be 0.086. More contradictory conclusions can be found in literature, for instance, in [10] results show that changing the number of facings of a product has little impact on sales as long as a minimum threshold is maintained. On the other hand, in [7], experimental results show that the size of the display (number of facings) has a strong reliability in driving attention.

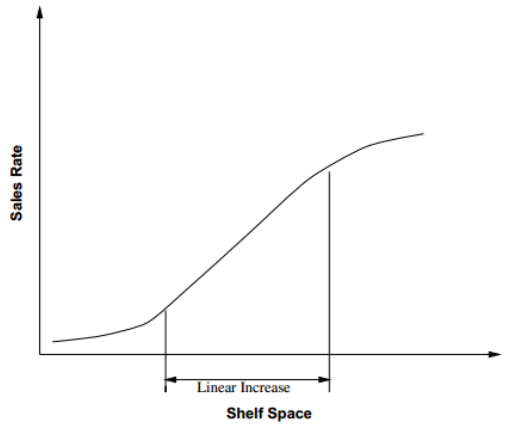

Figure 3.1: Variation of sales rate with shelf space for a single products [2].

Products adjacency (defined as cross-elasticity)

Another effect taken into account in experimental studies is the interdependency between adjacent products, defined as cross-elasticity. When two adjacent products are complementary, the sales of one of those products may lead to the sales of the other along with it (e.g. camera and batteries). On the other hand, if two adjacent products are substitute the sales of one product may cannibalize the sales of the other. Cross-elasticities are assumed to vary between -1 and 0 for substitute products and between 0 and 1 for complementary products. Drèze et al., in [10], reported increased sales in complementary merchandising above 5%. However, cross-elasticities are extremely difficult to estimate and therefore are disregarded in most of the studies [8]. Moreover, the cross elasticity from product i to product j is not the same as the cross elasticity from product j to product i [20]. As an example, if a camera is bought batteries may frequently be bought as well, but if batteries are bought a camera will not be bought as frequently.

Interestingly, these effects depend on the costumer's willingness to make impulse purchases

or replace their preferred products. In situations where the costumer is loyal to a product (more frequently in cases of premium products), shelf space management will have a lower influence on the purchase.

3.2 Optimization methods

Most of the complexity in the Shelf Space Allocation Problem comes from the demand function, which is hard to estimate and non-linear by nature. The literature presents a great variety of models, mostly differing on their demand functions, which incorporate different estimates of (some of) the factors that may influence costumers purchases. Additionally, these models also differ in the level of detail of the decisions, ranging from facings calculation to complete planogram descriptions. Models that can be adapted to reality are particularly difficult to find as they usually lack key practical constraints, such as product grouping. In general, space elasticity is the most commonly used effect with the demand rate of a product defined as a function of the space allocated to it and location decisions are usually disregarded. Finally, the Shelf Space Allocation Problem is usually addressed together with other related retail problems, further increasing the complexity of the demand functions. Assortment and inventory problems are the most frequent ones due to their intrinsic relation. The most relevant models will be revised in chronological order in section 3.2.1, with great focus on the demand function. The original notation of each paper is used, for simplification purposes.

As far as solution approaches are concerned, they are usually applied to simplistic versions of the problem and focus mainly on (meta)heuristics. In particular, Genetic Algorithms are one of the metaheuristics successfully applied to the problem. The work related to solution approaches will be revised in section 3.2.2.

3.2.1 Mathematical Models

Corstjens and Doyle (1981)

The model developed by Corstjens and Doyle in [21] was one of the first models for shelf space allocation and influenced most of the future literature. The authors were the first to incorporate both space and cross-space elasticities between products using multiplicative polynomial forces. The demand function for an individual product was defined in the following way:

$$q_i = \alpha_i S_i^{\beta_i} \prod_{\substack{j=1 \\ j \neq 1}}^{K} S_j^{\delta_{ij}} \tag{3.1}$$

In this equation, α_i represents a scale factor, β_i corresponds to the space elasticity of product i with respect to a unit in shelf space S_i, δ_{ij} represents the cross space elasticity between products i and j while K stands for the number of products. δ_{ij} can be positive or negative depending upon whether i and j are complementary or substitute products, and δ_{ij} is not necessarily equal to δ_{ji}.

For modeling space elasticity, Corstjens and Doyle used a diminishing return polynomial function. With this function, the demand of a product is continuously increasing with the number of facings, but the increasing rate slows down until the demand reaches a steady point ("s" shape) (see Figure 3.2). 3.2For the cross elasticity, the authors also used a polynomial function to model the increasing sales with the complementarity level or the decreasing sales with the substitution level between two products (see Figure 3.2 that shows on the left the demand in case of complementarity and on the right in case of substitution).

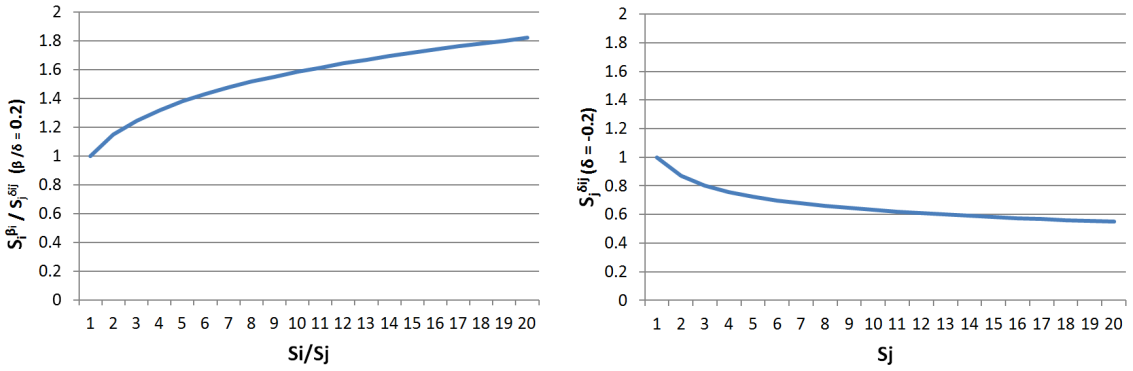

Figure 3.2: An illustration of the space and cross elasticity components of the demand function of a product for $\beta = 0.2$, $\delta = 0.2$ or -0.2.

The model is following:

$$\text{Maximize} \quad \sum_{i=1}^{K} (w_i q_i - C_i(S_i)) \tag{3.2}$$

$$\text{Subject to:} \quad \sum_{i=1}^{K} S_i \leq S^* \tag{3.3}$$

$$\alpha_i S_i^{\beta_i} \prod_{\substack{j=1 \\ j \neq 1}}^{K} S_j^{\delta_{ij}} \leq Q_i^* \ , \ \ i = 1, ..., K \tag{3.4}$$

$$S_i^L \neq S_i \neq S_i^U \ , \ \ i = 1, ..., K \tag{3.5}$$

$$S_i \geq 0 \ , \ \ i = 1, ..., K \tag{3.6}$$

where w_i is the margin rate for product i, C_i corresponds to the aggregate product costs for the store and K is the number of products. The constraints presented are the following:

- Store Capacity Constraint: space allocated to all products cannot exceed total available shelf space, S^*;
- Availability Constraint: for any product i, sales are limited by production or availability, Q_i^*;
- Control Constraint: S_i^L and S_i^U are, respectively, the lower and upper bounds for space allocated to product i;

- Non-negativity Constraint: S_i must not be negative, in order to obtain reasonable solution values.

Two related approaches are worth referring. Irion et al., in [22], changed the cost structure of the objective function and approximated the non-convex function by piecewise linear functions. In this way, the authors relaxed the model into an approximating linear Mixed Integer Programming Model (MIP).

Bai in [8], on the other hand, refers that the high number of parameters is difficult to estimate and drops the cross-space elasticity component of the demand plus the cross structure. He then proposes the following objective function:

$$P = \sum_{i=1}^{n} p_i \alpha_i x_i^{\beta}{}_i \tag{3.7}$$

In which α_i represents the scale factor, β_i corresponds to the space elasticity of item i, x_i is the number of facings of item i, p_i is the profit of item i and n is the total number of items.

Yang and Chen (1999)

Yang and Chen, in [16], point out some drawbacks of the Corstjens and Doyle model. Among them is the fact that the geometric model does not consider the number of products to be integer and neglects to consider location effects found in [10]. Due to these reasons, authors propose two new approaches - a comprehensive model and an alternate form. The comprehensive model has a demand function similar to Cortsjens and Doyle's, but considering the amount of facings of each product i by shelf k (x_{ik}), instead of the total space assigned to the product (S_i). They also change the elasticity parameter β_i to β_{ik}, to vary with the location of the shelves. However, their most important contribution was the simplistic alternate model, where they assumed linear profit within a constrained number of facings [1]. The simplified reformulated demand function is presented below.

$$\text{Maximize} \quad P = \sum_{i=1}^{n} \sum_{k=1}^{m} p_{ik} x_{ik} \tag{3.8}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} \alpha_i x_{ik} \leq T_k \ , \ \ k = 1, ..., m \tag{3.9}$$

$$L_i \leq \sum_{k=1}^{m} x_{ik} \leq U_i \ , \ \ i = 1, ..., n \tag{3.10}$$

$$x_{ik} \in N \cup \{0\} \ , \ i = 1, ..., n, k = 1, ..., m \tag{3.11}$$

In which n corresponds to the number of products, m is the number of shelves, p_{ik} refers to the per facing profit of product i in shelf k and x_{ik} corresponds to the number of facings of product i in shelf k. The three sets of constraints correspond to:

- Capacity constraint T_k of shelf k;

- Lower and Upper bounds of facings of product i (L_i and U_i, respectively);
- Decision variables x_{ik} are non-negative integers.

Both [2] and [8], pointed limitations of this demand function including its contradiction with the studies that report demand as a S-shaped function and low values for space elasticities of products. In spite of this, considering that retailers might want to operate in the (close to) linear part of the curve by setting boundaries to the number of facings such a demand function can be regarded as valid.

Yang and Chen also identified their alternate model as an extension to the multi-knapsack problem, proving that even the simplistic version of the problem is NP-hard.

Hwang et al. (2005)

In [23], Hwang et al. present an approach that differs from Yang and Chen's by integrating location effects into the Corstjens and Doyle model. To do this authors incorporated a location effect multiplier $\alpha_i \geq 1$ in the demand function which became the following:

$$D_i = (main\,space\,effect) \times (cross\,space) \times (location\,effect) = d_i X_i^{\beta_i} \left[\prod_{k \neq i}^{N} X_k^{\beta_{ik}} \right] \alpha_i \tag{3.12}$$

$$\text{With}: \quad X_i = \sum_{j=1}^{M} X_{ij} \tag{3.13}$$

$$\text{and}: \quad \alpha_i = \frac{\sum_{j=1}^{M} X_{ij} \alpha_j}{X_i} \tag{3.14}$$

where, X_i corresponds to the total number of item i displayed on shelves, X_{ij} is the number of item i displayed on shelf j, d_i represents the scale parameter of demand function for item i (d_i>0), β_i is the space elasticity for item i, β_{ik} is the cross-space elasticity between item i and k, α_j corresponds to the scale parameter that reflects the increase of demand rate with respect to the level of shelf when items are displayed on shelf j, α_i is the weighted average of α_j when item i is displayed on more than one shelf, M is the number of shelves in a particular categorized area of the store and N is the number of brands of items.

The above demand function is integrated in an inventory control problem with the objective of maximizing the retailers profit. In this approach authors additionally determine the ordering quantities for the products. Constraints considered in this model are similar to the previous models, including capacity, control and non-negativity constraints.

Russell and Urban (2010)

In [13], Russel and Urban were two of the first authors to explicitly consider the products as part of a family, that can be based on a variety of characteristics, such as brand, flavour, price set,

among others. Products of these families should be kept together and, for aesthetic reasons, in uniform and rectangular shapes.

Authors developed a formulation to determine the optimal shelf location for the products considering both space and location elasticities. They based their demand function on a previous work by Drèze et al. who noted that sales tend to be quadratic in the horizontal dimension and cubic in the vertical one. As for space elasticity, authors chose to use a quadratic formulation, not only for consistency and tractability, but also because it reflected diminishing returns.

The expected demand for each product i, ξ_i, is then expressed as:

$$xi_i = \beta_{0i} + \beta_{1i}X_i + \beta_{2i}X_i^2 + \sum_k \left[(\beta_{3i}(h_k Y_{ik}) + \beta_{4i}(h_k Y_{ik})^2 + \beta_{5i}(h_k Y_{ik})^3) + \beta_{6i}Z_{ik} + \beta_{7i}Z_{ik}^2\right] \quad (3.15)$$

where, X_i is a continuous variable representing the horizontal location of item i on a shelf, Y_{ik} is a binary variable equal to one if item i is located on shelf k, Z_{ik} is an integer variable representing the number of facing of item i placed on shelf k and $\beta_{\bullet i}$ are appropriate coefficients for the specific implementation. Since Y_{ik} is a binary variable, the demand formulation can be expressed as the following quadratic function:

$$xi_i = \beta_{0i} + \beta_{1i}X_i + \beta_{2i}X_i^2 + \sum_k \left[(\beta_{3i}h_k + \beta_{4i}h_k^2 + \beta_{5i}h_k^3)Y_{ik} + \beta_{6i}Z_{ik} + \beta_{7i}Z_{ik}^2\right] \quad (3.16)$$

To formulate the model, it is considered that there are a number of families (f=1,2,...,F) each of which comprises one or more items/products/SKUs (i,j = 1,2,...,N). Each item as a contribution margin, ϕ_i, a facing length i, and a maximum and minimum number of facings, z_i^+ and z_i^-, respectively. There is also a number of shelves (k=1,2,...,M) with shelf length, l_k, and height, h_k. In case a product family occupies more than one shelf, items of the family must form an uniform column, allowing for a deviation no larger than υ from one shelf to the next, where υ is measured as a distance. The objective function aims to maximize profit and is the following:

$$\text{Maximize} \quad \pi = \sum_i \phi_i \xi_i \quad (3.17)$$

$$\text{Subject to:} \quad X_i \geq \frac{d_i}{2} Z_{ik}, \ \forall i,k \quad (3.18)$$

$$X_i \leq l_k - \frac{d_i}{2} Z_{ik}, \ \forall i,k \quad (3.19)$$

$$\sum_k Y_{ik} = 1, \ \forall i \quad (3.20)$$

$$\sum_k Z_{ik} \geq z_i^-, \ \forall i \quad (3.21)$$

$$\sum_k Z_{ik} \leq z_i^+ Y_{ik}, \ \forall i,k \quad (3.22)$$

Constraints 3.18 and 3.19 guarantee that the product facing will not extend beyond the end of shelves. Constraint 3.20 ensures that each item is located on only one shelf. Constraints 3.21 and 3.22 limit the number of facings to a minimum and a maximum value.

Constraints 3.23 and 3.24 are included in the model, so that physical overlap of the items does not occur. P_{ij} is defined as a binary variable equal to one if item i is located to the left of item j.

$$X_i - \frac{d_i}{2}Z_{ik} \geq X_j + \frac{d_j}{2}Z_{jk} - l_k P_{ij} - l_k(2 - Y_{ik} - Y_{jk}),\ \forall i, j(i), k \tag{3.23}$$

$$P_{ij} + P_{ji} = 1,\ \forall i, j(i < j) \tag{3.24}$$

In order to maintain product integrity, the following variables are defined:

- i(f) - subset of items that compose family f;
- W_{fk} - binary variable equal to one if family f is located on shelf k;
- $A_{fk}(B_{fk})$ - continuous variables representing the left (right) coordinate of family f on shelf k;
- $R_f(S_f)$ - integer variables representing the top (bottom) shelf on which family f is located.

$$A_{fk} \leq X_{i(f)} - \frac{d_{i(f)}}{2}Z_{i(f),k} + l_k(1 - Y_{ik}),\ \forall i(f), k \tag{3.25}$$

$$B_{fk} \geq X_{i(f)} + \frac{d_{i(f)}}{2}Z_{i(f),k} - l_k(1 - Y_{ik}),\ \forall i(f), k \tag{3.26}$$

$$B_{fk} - A_{fk} = \sum_{i(f)} d_{i(f)} Z_{i(f),k},\ \forall f, k \tag{3.27}$$

$$W_{jk} \leq \sum_{i(f)} Y_{i(f),k},\ \forall f, k \tag{3.28}$$

$$W_{jk} \geq Y_{i(f),k},\ \forall i(f), k \tag{3.29}$$

$$R_f \geq k W_{fk},\ \forall f, k \tag{3.30}$$

$$S_f \leq M - (M - k)W_{fk},\ \forall f, k \tag{3.31}$$

$$R_f - S_f = \sum_k W_{fk} - 1,\ \forall f \tag{3.32}$$

$$R_f \geq S_f,\ \forall f \tag{3.33}$$

Constraints 3.25 and 3.26 establish the left and right coordinates of family f on each shelf. Constraint 3.27 guarantees that items of the same family are adjacent. Constraints 3.28 and 3.29 define W_{fk} variables. Top and bottom shelves on which family f is located are established by constraints 3.30 and 3.31. Constraint 3.32 keeps the family on adjacent shelves and constraint 3.33 ensures that the top shelf of family f is not below the bottom shelf of the family.

$$A_{fk} - A_{f,k+1} \leq \upsilon + l_k(2 - W_{fk} - W_{f,k+1},\ \forall f, k(k \leq M - 1)) \tag{3.34}$$

$$A_{f,k+1} - A_{fk} \leq \upsilon + l_k(2 - W_{fk} - W_{f,k+1},\ \forall f, k(k \leq M - 1)) \tag{3.35}$$

$$B_{fk} - B_{f,k+1} \leq \upsilon + l_k(2 - W_{fk} - W_{f,k+1},\ \forall f, k(k \leq M - 1)) \tag{3.36}$$

$$B_{f,k+1} - B_{fk} \leq \upsilon + l_k(2 - W_{fk} - W_{f,k+1},\ \forall f, k(k \leq M - 1)) \tag{3.37}$$

Constraints 3.34 and 3.35 guarantee that left coordinates of a family are kept within υ units from one shelf to the next. The same is done to the right coordinates by constraints 3.36 and 3.37.

Hansen et al. (2010)

Hansen et al., in [24], propose that retailers can use a linear integer programming formulation of the non-linear profit function by extending the Yang and Chen's simplified alternate objective function. They discretized the number of facings, turning the variable into binary. This way, the space elasticity values could be determined beforehand for each product i, in shelf k and number of facings h. Following the same approach, they extended the model to consider horizontal impact and the cross-elasticity effect. The resulting model is the following:

$$\text{Maximize} \quad Z = \sum_{j=1}^{N}\sum_{k=1}^{S}\sum_{h=1}^{T_k}\sum_{f=1}^{T_k} P_{jkhf} \times x_{jkhf} + \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{V_{ij}}{2} \times e_{ij} \tag{3.38}$$

$$\text{Subject to:} \quad \sum_{j=1}^{T_k}\sum_{f=1}^{T_k} x_{jkhf} \leq 1 \; \forall j = 1,...,N; k = 1,...,S \tag{3.39}$$

$$\sum_{j=1}^{N}\sum_{f=1}^{h}\sum_{q=h-f*a_j}^{h} x_{jkqf} \leq 1 \; \forall h = 1,2,...,T_k; k = 1,...,S \tag{3.40}$$

$$L_j \leq \sum_{k=1}^{S}\sum_{h=1}^{T_k}\sum_{f=1}^{T_k} f * x_{jkhf} \leq U_j \; \forall j = 1,2,...,N \tag{3.41}$$

$$b_j \leq \sum_{k=1}^{S}\sum_{h=1}^{T_k}\sum_{f=1}^{T_k} f * a_j * x_{jkhf} \; \forall j = 1,...,N \tag{3.42}$$

$$V_{ij} \leq b_i, V_{ij} \leq b_j, V_{ij} \geq 0, b_j \geq 0 \; \forall j = 1,2,...,N \tag{3.43}$$

$$x_{jkhf} = 0 \; \forall j = 1,...,N; k = 1,...,S; h = 1,...,T_k; f > (T_k - h)/a_j \tag{3.44}$$

$$x_{jkhf} \in 0,1 \; \forall j = 1,...,N; k = 1,...,S; h = 1,...,T_k; f = 1,...,T_k \tag{3.45}$$

Where, P_{jkhf} is the profit of product j on shelf k of length T at horizontal level h for face-length f and x_{jkhj} represents the allocation decision for product j on shelf k starting at horizontal level h for face-length f (=1 if true). N corresponds to the total number of products and S is the total number of shelves. The lower and upper bounds of x_{jk} are L_j and U_j, respectively. Constraint 3.39 guarantees that product j is not allocated more than once for its f number of facings on a shelf. Constraint 3.40 avoids physical overlap of products. Constraints 3.41 and ensure that product j is allocated between its limits. Constraints 3.42 and 3.43 and the second part of the objective function correspond to cross-elasticity effects. Cross-elasticity is defined as an N x N matrix, E, where each value of $e_{ij} \in E$ represents the incremental profit or loss due to the cross-product effects. Cross-product elasticity is represented as $min(b_i, b_j) \times e_{ij}$ where b_i and b_j are the total length of products i and j, respectively on the shelf. V_{ij} and constraint 3.43 are used to avoid a nonlinear objective function, and is defined as $V_{ij} = min(b_i, b_j)$.

Models presented in literature have a number of drawbacks that include the difficulty to estimate parameters, such as space and cross-space elasticities of products and the fact that most of them present non-linear demand functions. Another limitation found in most models is that they do not incorporate all of retailers preferences regarding shelf space. One of these preferences is the need to consider family products, preferably forming regular shapes for aesthetic purposes. Lim et al. [11] and Russell and Urban[13] already began to address this problem, although practical applicability of their models is still limited.

3.2.2 Solution Approaches

In order to tackle the Shelf Space Allocation Problem, different approaches can be found in literature. Reviewed approaches include models that incorporate space and cross-space elastic demand, of which Corstjens and Doyle's model [21] is an example. This geometric programming model was solved using a branch and bound algorithm. Another approach found in literature considers space allocation models with space elastic demand. One of the most important models included in this approach is Yang and Chen's [16] alternate model. Following this stream other modeling approaches have been developed, such as Yang's [25], Lim et al.'s [11], Hansen et al.'s [24], Gaijar and Adil's [26] and Castelli and Vanneschi's [27].

In [25], Yang proposed an algorithm to solve the problem, based on the algorithm used for solving a knapsack problem. In the heuristic developed, shelf space is allocated item by item according to the descending order of sales profit for each item per display area or length. The steps followed in the heuristic include: (1) A preparatory phase, where it is verified whether the shelf space available is enough to satisfy the minimum requirement of facings. The weight given to the profit of each product is also sorted in this phase; (2) an allocation phase, where an initial solution for the shelf space allocation of products is calculated;(3) an adjustment phase and finally (3) a termination phase, where the total profit of the final solution is calculated.

Finding limitations in the heuristic proposed by Yang [25], Lim et al. [11] optimized it by resorting to metaheuristics, proposing an hybrid solution strategy, five-phase Squeaky Wheel Optimization with Local (SWOL) search. Bai simplified the comprehensive model by Yang and Chen [16] and proposed greedy heuristics, simulated annealing heuristics and hyper-heuristics for the problem. Gajar and Adil [26], based on the alternate model by Yang and Chen [16], developed heuristics based on a new initial construct and a neighborhood search strategy to solve the problem. Hansen et al. [24] and Castelli and Vanneschi [27] also based on the alternate model by Yang and Chen [16], use genetic algorithms to tackle the problem. Hwang et al. [23] solve the problem of shelf space allocation and inventory control by using a gradient search heuristic and a genetic algorithm, integrating location effects in the Cortjens and Doyle model. These last three approaches, which resort to genetic algorithms will be further reviewed in section 3.3.2.

Knapsack-like models used to tackle the Shelf Space Allocation Problem present drawbacks, as they only include non-negativity, integer and capacity constraints, but ignore other policy constraints.

3.3 Genetic Algorithms

3.3.1 Introduction

Characteristics of optimization problems can vary widely. Depending on the complexity of a problem, the size of input instances or the required search time to solve a given problem, an exact approach might not be the most adequate method to tackle the problem. Metaheuristics are an alternative that allows to solve large-size problem instances in a reasonable time, obtaining satisfactory solutions. As a downside, finding global optimal solutions or even bounded solutions is not guaranteed [3]. Genetic Algorithms (GAs) are a well known and widely used metaheuristic which was first introduced by John Holland in 1975 [28]. They belong to the very popular class of Evolutionary Algorithms (EAs) and may be classified as a nature inspired, population based metaheuristic. GAs apply the concept of survival of the fittest and evolve a population of individuals to find the optimal or near optimal solution to combinatorial problems.

In Figure 3.3, a representation of the process followed by GAs is presented. The encoded solution is known as a chromosome, whereas the decision variables within a chromosome are referred to as genes. A set of chromosomes constitutes a population which is evolved for a number of iterations (known as generations), simulating the evolution of species. Usually, the initial population is generated randomly. Every individual (chromosome in GAs) is evaluated through an objective function. This objective function corresponds to a fitness value that indicates the individual's suitability to the problem. In each generation, the individuals with the better fitness are selected with a higher probability. Afterward, the selected individuals are reproduced through variation operators, such as mutation and crossover to generate new offspring. The process is repeated until a chosen stopping criteria is met [3]. In Figure 3.4, a template of an evolutionary algorithm is represented.

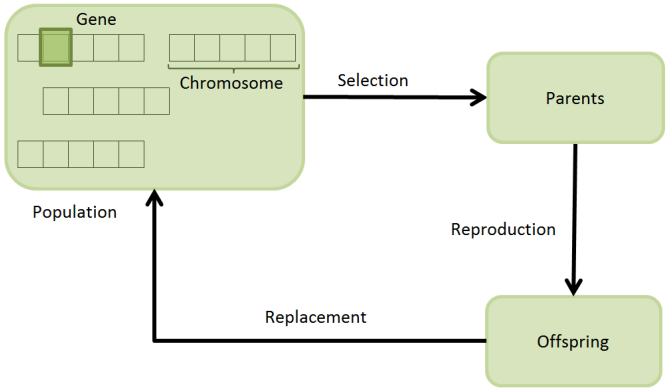

Figure 3.3: A generation in a genetic algorithm

This dissertation focuses on the use of a Biased Random-key Genetic Algorithm to tackle the Shelf Space Allocation Problem. A BRKGA corresponds to a particular type of genetic algorithms that will be explained in the following chapter.

```
Generate(P(0)) ; /* Initial population */
t = 0 ;
While not Termination_Criterion(P(t)) Do
   Evaluate(P(t)) ;
   P'(t)      = Selection(P(t)) ;
   P'(t)      = Reproduction(P'(t)); Evaluate(P'(t)) ;
   P(t + 1)   = Replace(P(t), P'(t)) ;
   t = t + 1 ;
End While
Output Best individual or best population found.
```

Figure 3.4: Template of an evolutionary algorithm [3]

3.3.2 Genetic Algorithms for the Shelf Space Allocation Problem

Different heuristics and metaheuristics have been used, throughout literature, to tackle the Shelf Space Allocation Problem. Due to its relevance for this dissertation, the use of genetic algorithms for this problem will be revised. The most relevant papers will be presented in chronological order.

Hwang et al. (2004)

In [23], Hwang et al. propose a gradient search heuristic and a genetic algorithm to solve a Shelf Space Allocation and Inventory Control problem. For the purpose of this dissertation, only the genetic algorithm developed will be studied. The problem faced by the authors, consists of having a retailer who displays various brands of items within a category to multi-level shelves that have limited space. The demand function considered is an extension of Corstjen and Doyle's, in which location effects are incorporated. Constraints include capacity, control and non-negativity constraints.

Solution Encoding	Chromosome keys are represented as a number in the interval]0,1[. Each chromosome key corresponds to the total number of item i displayed on shelves.
initialization	Genes from the initial populations are randomly generated in the interval $[X_i^{min}, X_i^{max}]$.
Crossover and mutation	The "one-cut-point" method is used, in which one "cut-point" is chosen and exchanges the right part of two parents. Probability of crossover is considered to be 0.25. On the other hand, mutation is used to alter one or more genes with probability 0.01.
Constraints	To incorporate constraints in the algorithm, penalties are imposed according to the degree of violation of constraints.
Fitness Function	The fitness function considered to evaluate chromosomes is the total profit of each solution. The aim is to maximize the fitness function. This function is scaled to avoid premature convergence and to diversify the population.

Population update New populations are selected among the off-spring obtained through crossover and mutation. The probability of selection of a certain off-spring chromosome is proportional to its scaled fitness value and is obtained through a roulette wheel technique.

This algorithm was tested for a maximum of 4 items and 6 shelves.

Hansen et al. (2010)

Hansen et al. [24] propose a metaheuristic approach to solve the Shelf Space Allocation Problem over a heuristic approach, claiming that the proposed metaheuristic requires less computational effort and managers can directly apply the solution to the retail shelf-space.

Solution Encoding A two-dimensional array is defined, consisting of five rows and S× N columns, to represent a chromosome, where S is the number of shelves and N is the number of products. The first two bottom rows represent the products and shelves, respectively. The third and fourth rows from the bottom represent the face-length of a product in each shelf. The number in these rows are random integers, generated so that the total face-length of a product j in row three is equal to the lower bound (L_j) and the fourth row values are between 0 and ($U_j - L_j$). Finally, the fifth row from the bottom represents the relative horizontal position of a product on a shelf using random keys or a uniform random number in the range of (0,1).

Crossover and mutation A single point crossover with mating probability p_c is used. The single point crossover chooses a random point in each of two strings to form two sub strings one to the left of the point and one to the right. The left part of the string of one parent and the right part of the string of the other parent are spliced together. Concerning mutation, two different types are considered: uniform and swapping. Mutation is used to avoid being stuck on local optimal solutions.

Constraints To allocate products on shelves, the algorithm starts by using the minimum display requirement in row three. Afterward, it is repeated using row four. The results are evaluated with the fitness function.The order by which products are placed on shelves is given by the increasing order of random keys or uniform random keys.

Fitness Function The fitness function evaluates the total profit generated for the allocation of products on different shelves.

Population update Binary tournament selection is used to randomly generate chromosomes. Binary tournament selection consists of randomly choose two chromosomes from the population and the most profitable is selected for the next generation.

This algorithm is applied for a maximum of 100 products and 10 shelves.

Castelli and Vanneschi (2014)

In [27], Castelli and Vanneschi propose a hybrid algorithm that combines a Genetic Algorithm and a Variable Neighborhood Search (VNS) algorithm for the Shelf Space Allocation Problem. The model followed by Castelli and Vanneschi is based on Yang and Chen's alternate model.

Solution Encoding	A possible solution is represented as a string s of length equal to n.m. Where n is the number of products and m is the number of shelves. Each position $s_{i,k}$ of the chromosome corresponds to the number of facings of product i on shelf k.
Initialization	Initialization is performed randomly, in which chromosomes keys have values in the range $[[\frac{L_i}{m}];U_i]$. The procedure ends when all chromosomes from the population satisfy the hard constraints.
Crossover and mutation	To perform crossover, two individuals p_1 and p_2 are chosen as parents, based on the mating selection policy. Then, a crossover point is chosen then the resulting parts of the chromosome are exchanged, as long as hard constraints are not violated. Regarding mutation, a locus $p_1[l]$ is selected with probability $p \leq p_m$, where p_m is the mutation probability. With uniform probability, it is considered that $p_1[l] = p_1[l] \pm 1$. If hard constraints are not violated, the new solution is accepted. Otherwise, the performed mutation is discarded.
Fitness Function	The fitness function under consideration aims to maximize profit.
VNS	A Variable Neighborhood Search is used to explore the possibility of improving the exploration/exploitation ratio compared to only using a Genetic Algorithm.

This algorithm is applied for a maximum of 100 products and 10 shelves.

3.3.3 Biased Random-key Genetic Algorithm

Biased Random-key Genetic Algorithms (BRKGAs) are a particular type of Random-key Genetic Algorithms (RKGA) that were first introduced by José Fernando Gonçalves and Mauricio Resende in 2009 [4]. They have successfully been used to solve different optimization problems (e.g. [29] [4] [30] [17] [5]).

Genetic Algorithms with random-keys were first introduced by Bean, in 1994 [31] for solving sequencing problems. In RKGAs chromosomes are represented as randomly generated keys in the interval [0,1]. Unlike traditional Genetic Algorithms, RKGAs move all the feasibility issues into the objective evaluation procedure and guarantees that all offspring formed during the reproduction phase are feasible. Moreover, they use a single parametrized uniform crossover and keep the best fitness value solutions from one generation to the other (elite), resulting in a monotonically evolved

heuristic. As for mutation, RKGA generates a set of random individuals that add to the population. A decoder, that corresponds to a deterministic algorithm, associates a chromosome with a solution of the optimization problem. This decoder is the only problem dependent part of the algorithm.

BRKGAs differ from RKGAs in the way parents are selected for mating. On one hand, in RKGA both parents are randomly chosen among the individuals of the population. On the other hand, in BRKGA a random parent is chosen from the elite group of the current population and the other is chosen from the rest of the population. Moreover, in BRKGA, the offspring inherits more characteristics from the elite parent.

The flowchart of the BRKGA algorithm is presented in Figure 3.5. The algorithm starts with an initial population of p vectors and r random-keys. After the population is evaluated, individuals are divided into pe elite individuals (best fitness values) and p-pe non-elite individuals. The elite individuals of generation g are copied to the population of generation g+1. A small number of mutants (pm) is also introduced in the next generation's population. To complete the population of generation g+1, p-pe-pm individuals still need to be generated. These individuals are obtained through mating or crossover, in which a random parent is chosen from the elite group and the other from the rest of the population (Figure 3.6). Repetition is allowed in the selection of parents. As such, each individual can generate more than one offspring in the same generation.

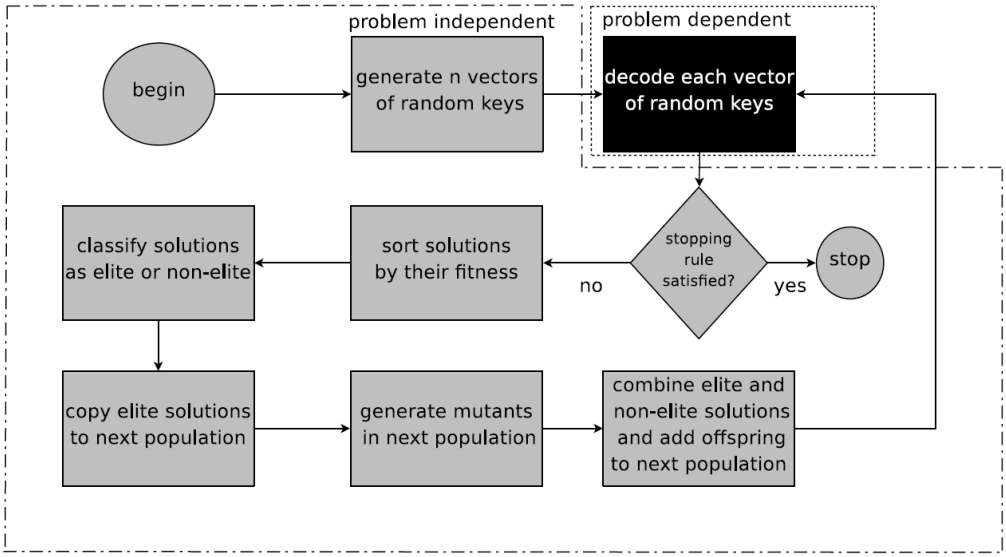

Figure 3.5: Flowchart of the BRKGA algorithm [4]

Mating is implemented through a parametrized uniform crossover. As seen in Figure 3.7, a crossover parameter (ρe) defines the probability that the offspring inherits a component from the elite parent. If the chromosome key value is higher than ρe, then the parent chromosome 2 key (elite) is chosen. Otherwise, the parent chromosome 1 key (non elite) is selected. In the BRKGA, the crossover parameters should be higher than 0.5. In case all vectors of random-keys correspond to feasible solutions, the offspring that results from mating will also correspond to feasible solutions.

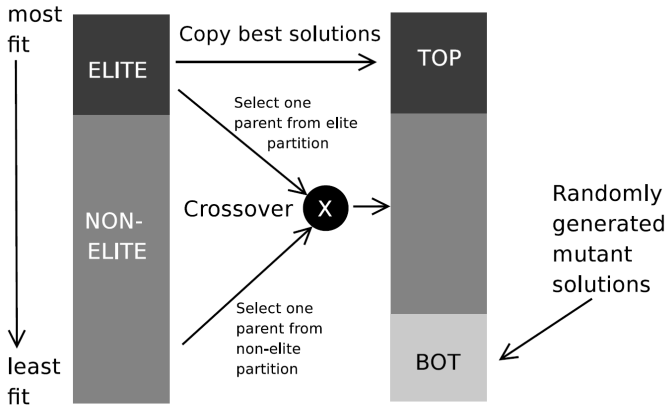

Figure 3.6: Transition from generation g to generation g+1 [4]

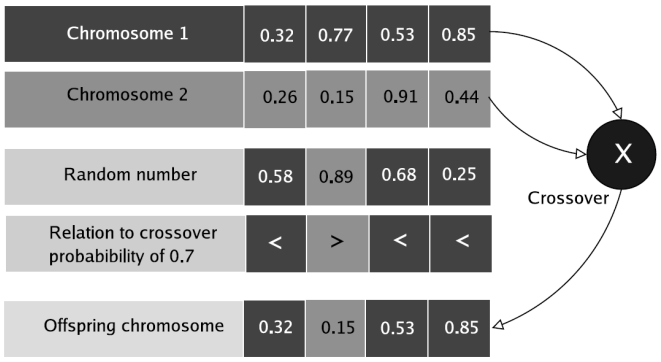

Figure 3.7: Parameterized uniform crossover [4]

After the population is complete, the new fitness values are calculated. The population is once again divided into elite and non-elite to start a new generation.

The problem dependent elements of the algorithm that must be specified include the way solutions are coded and decoded and the way their fitness values are calculated.

A BRKGA API has been developed [32]. In this API, only the problem dependent part of the algorithm has to be implemented by the user. Other parameters have to be adjusted. Although these parameters do not follow a specific criteria, in [4], the suggestions presented in Table 3.1 are made.

Table 3.1: Recommended parameter value settings.

parameter	description	recommended value
p	size of population	p=an, where $1 \leq a \in \Re$ is a constant and n is the length of the chromosome
p_e	size of elite population	$0.10p \leq p_e \leq 0.25p$
p_m	size of mutant population	$0.10p \leq p_m \leq 0.30p$
ρ_e	elite alele inheritance probability	$0.5 \leq \rho_e \leq 0.8$

3.3.4 Biased Random-key Genetic Algorithms for Cutting and Packing Problems

As discussed in chapter 2, Cutting and Packing Problems bare a resemblance to the Shelf Space Allocation Problem. Although Biased Random-key Genetic Algorithms have not been used to tackle the Shelf Space Allocation Problem, they have been used to successfully solve Cutting and Packing Problems [17] [5].

Among the examples found in literature, the Two-dimensional Packing Problem and the Two-dimensional Non-guillotine Orthogonal Cutting Problem share a greater similarity with the Shelf Space Allocation Problem. Both these problems use two dimensions and have small rectangles that have to be either packed into or cut from a larger rectangle. The same happens in the Shelf Space Allocation Problem where retailers want to place smaller rectangles (products) into larger rectangles (shelves). In spite of this, the objective function differs between the problems. Due to this reason, the review of Biased Random-key Genetic Algorithms for Cutting and Packing Problems will focus only on the encoding and decoding of solutions and not on their evaluation through the fitness function.

In [5], considering M to represent the number of smaller rectangles, each chromosome is divided as follows:

$$Chromosome = (\underbrace{gene_1 ... gene_M}_{Rectangle\ Packing\ Sequence} , \underbrace{gene_{M+1} ... gene_{2M}}_{Vector\ of\ Placement\ Procedures})$$

Figure 3.8 illustrates the sequence of steps applied to each chromosome for its decoding into a packing or cutting solution and corresponding fitness value.

Firstly, the component of the chromosome that corresponds to the Rectangle Packing Sequence is decoded to provide the sequence according to which smaller rectangles will be packed into or cut from the larger rectangle. To do this, the chromosome keys are sorted in ascending order.

Secondly, the second component of the chromosome is used to choose the placement procedure that will be used to pack or cut smaller rectangles. The placement procedure can be either Left-Bottom (LB) or Bottom-Left (BL) according to the following expression:

$$\begin{cases} BL,\ if\ gene(M+1) \leq \frac{1}{2}; \\ LB,\ if\ gene(M+1) > \frac{1}{2}. \end{cases}$$

In the Left-Bottom procedure, if the small rectangle fits an empty space, it is placed as far to the left of the larger rectangle as possible and than as close to the bottom as possible. Similarly, in the Bottom-Left procedure, if the small rectangle fits an empty space, it will be placed as close to the bottom as possible and than as far to the left as possible. Using only a Bottom-Left placement procedure would not allow to obtain all possible optimal solutions [33].

To determine the spaces where small rectangles can be placed the Difference Process (DP) developed in [6] is used. For a Three-dimensional problem, the Difference Process is obtained as follows:

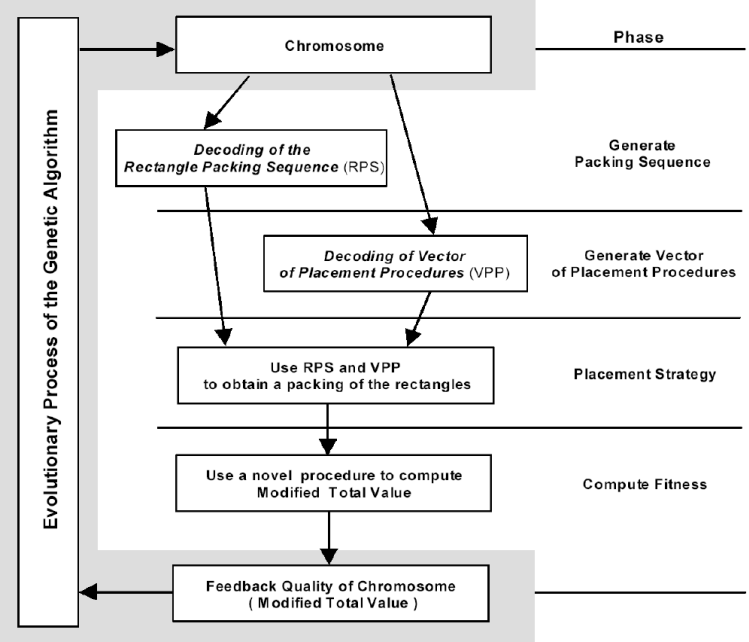

Figure 3.8: Architecture of the heuristic [5]

Consider a small hypothetical box whose bottom left and upper right coordinates are (x3,y3) and (x4,y4), respectively. This box will be placed on a larger box with bottom left coordinates (x1,y1) and upper right coordinates (x2, y2). It is assumed that x1≤x3≤x4≤x2 and y1≤y3≤y4≤y2.

After the small box is placed on the larger box, the empty rectangles generated are calculated in the following manner:

$$[(x1,y1),(x2,y2)] - [(x3,y3),(x4,y4)] =$$

$$[(x1,y1),(x3,y2)]$$

$$[(x4,y1),(x2,y2)]$$

$$[(x1,y1),(x2,y3)]$$

$$[(x1,y4),(x2,y2)]$$

The problem is illustrated in Figure 3.9.

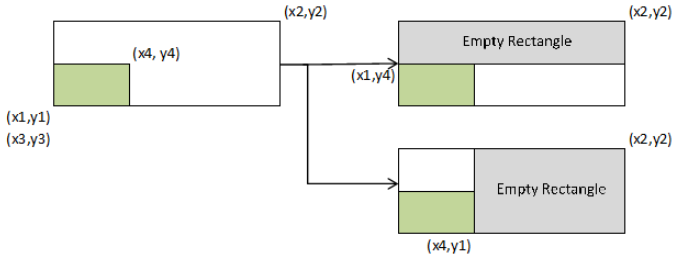

Figure 3.9: Difference process in interval generation. [6]

After the Difference Process, an elimination has to take place to remove space intervals that are inscribed in others and space intervals that have infinite thinness. The process is repeated after each rectangle is placed.

The placement procedure is represented in Figure 3.10, in which ERS corresponds to the empty rectangular space.

```
procedure PLACEMENT
1   for i = 1,...,M do
2       if PP_i = BL then
3           Let ERS_i^* be the ERS, in the list of available ERSs,
·           in which rectangle r_i is placed when the Bottom-Left
·           placement heuristic is applied;
4       end if
5       else if PP_i = LB then
6           Let ERS_i^* be the ERS, in the list of available ERSs,
·           in which rectangle r_i is placed when the Left-Bottom
·           placement heuristic is applied;
7       end if
8       if ERS_i ≠ ∅ then
9           Place r_i at bottom left corner of ERS_i^*;
10          Update list of available ERSs using the DP process
·           of Lai and Chan (1997b);
11      end if
end PLACEMENT;
```

Figure 3.10: Pseudo-code of the placement procedure. [5]

Afterward, the solution is evaluated through the objective function which differs according to the problem.

Chapter 4

Biased Random-key Genetic Algorithm for the Shelf Space Allocation Problem

The Biased Random-key Genetic Algorithm has been used to successfully solve Cutting and Packing Problems, which bare a resemblance to the Shelf Space Allocation Problem (see section 2.4 and 3.3.3). Aiming to prove its suitability to tackle the Shelf Space Allocation Problem, two novel approaches were developed which will be described in this chapter. In Appendix A, an example for each of the approaches developed can be found.

4.1 Problem Definition

As explained in previous chapters, there is not a single definition or model for the Shelf Space Allocation Problem. In fact, although some of the constraints are hard, others vary according to retailers preferences. The Shelf Space Allocation Problem tackled in this work is defined according to the reality of the case study. Bearing in mind the different types of models present in the literature, this problem is most related with the one tackled by Russel and Urban in [13].

Decision Variables

The aim is to obtain fully defined planograms and, therefore, the problem considers the overall spectrum of decisions, namely the number of facings, allocation (which shelf) and location (which horizontal position in the shelf). This level of detail is rare in the literature as most approaches focus only on facings and allocation decisions.

Objective Function

For determining the number of facings, the case study gives more importance to the inventory level of the products than their space elasticities. This way they can manage stock more efficiently, integrating downstream decisions in the problem. Therefore, instead of using a demand function to

evaluate the number of facings for the products, we consider the need to balance products rotation days (i.e. the number of days until the products become out-of-stock).

The target number of facings for each product is determined beforehand and given as an input to the algorithms, whose goal is trying to accomplish those values while taking into consideration space constraints. One advantage of this approach is the possibility of changing at any time the way facings are calculated without further changes in the algorithm. It also allows to consider more theoretic approaches for facings calculation.

We additionally aim to maximize sales bearing in mind the vertical location of the products. For that purpose, the curve from Figure 4.1 is considered. This curve gives more emphasis to the shelves at eye and hand-levels. This curve combines experimental findings of marketing literature with the company's knowledge concerning the importance of vertical location.

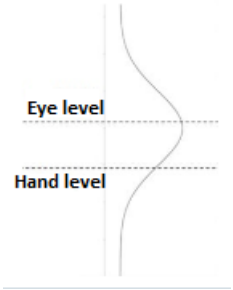

Figure 4.1: Impact of vertical location on sales (the vertical axis presents the height of the shelf and the horizontal axis the impact).

Constraints

The case study constraints were described in chapter 2.3.3. Concerning the family block constraints, the retailer works with more than one level of family blocks (i.e. products might be grouped, for instance, into brand and then further divided into size and flavor). Furthermore, the second level might be different for each brand, given rise to a network of blocks never seen in the literature.

To better understand this network of blocks, a block diagram is exemplified in Figure 4.2. In this example, products 7-13 are beverages that should be allocated to 4 shelves, following a given merchandising guideline. This guideline first divides the products into two brands, corresponding to blocks 1 and 2. Afterwards, the products from each brand are further divided into flavors. Block 1 contains flavors 3 and 4 whereas Block 2 contains flavors 5 and 6. Figure 4.3 shows a possible solution for this problem.

4.2 Biased Random-key Genetic Algorithms

Two different Biased Random-key Genetic Algorithms were developed to tackle the Shelf Space Allocation Problem. The main difference between these algorithms concerns the way products are

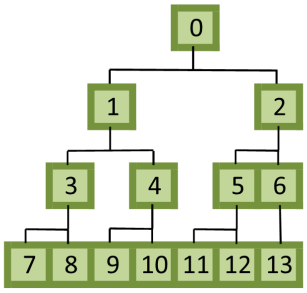

Figure 4.2: Block diagram for the example.

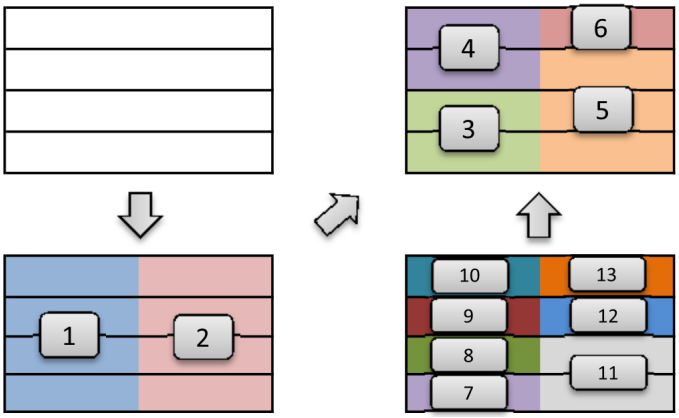

Figure 4.3: Allocation of capacities, brands and products.

allocated on shelves. The second algorithm developed goes further into the case study and includes an additional practical constraint. This constraint states that family blocks should be placed either vertically or horizontally. If they are placed vertically, they will occupy the entire height of the corresponding upstream block. On the other hand, if family blocks are placed horizontally, they will occupy the upstream block in width. Moreover, all the blocks belonging to the same upstream block should have the same direction. Looking again into Figure 4.3, one can say that blocks 1 and 2 are vertically placed and blocks 3-6 horizontally.

4.2.1 First BRKGA for the SSAP

The approach developed is inspired by the BRKGA for the Constrained Two-Dimensional Non-Guillotine Orthogonal Cutting Problem [5]. A description of the algorithm will be presented in this chapter. Furthermore, an example of its application can be found in Appendix A.

4.2.1.1 Algorithm Architecture

In the Biased Random-key Algorithm, chromosome keys are evolved and decoded, until the best solution is found. The algorithm is stopped if all products are allocated and the fitness function

does not improve after a given number of generations or a time limit is reached. The different phases of the decoder are presented in Figure 4.4 and described in detail in the following sections.

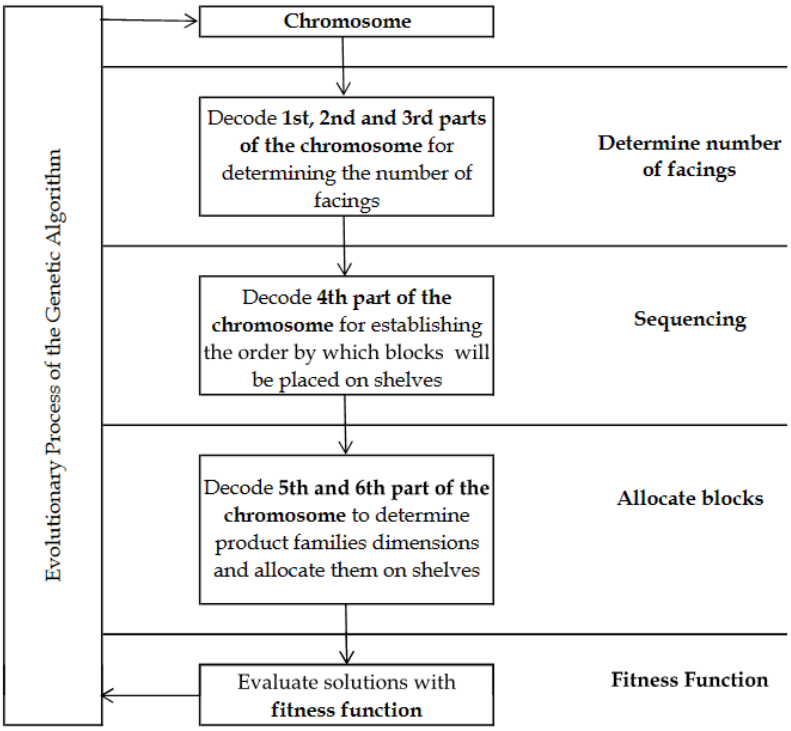

Figure 4.4: Architecture of the algorithm.

(1) *Determine number of facings:* The first phase decodes part of the chromosome into the number of facings that should be allocated to each product. Although a target number is given beforehand, the family block constraints turn unlikely that the targets will be reached for all the products. Facings calculation is carried out in three stages, each one having a component of the chromosome: remove facings, determine the amount of space that will be left empty on the shelves after products are placed and add facings to products until the amount of space that will be left empty is fulfilled or no more facings can be added.

(2) *Sequencing:* The second phase uses a fourth component of the chromosome to determine the sequence in which the blocks (family and products) are placed on the shelves.

(3) *Allocate Blocks:* The third phase allocates the blocks on the shelves. For that purpose, it starts by determining the dimensions of the family blocks (height and width), by using a fifth component of the chromosome. Afterwards, a sixth and last component of the chromosome is used to choose between two placement strategies: Left-Bottom and Bottom-Left.

(4) *Fitness Function:* An original fitness function is used to evaluate the quality of the solution.

4.2.1.2 Chromosome Representation

For this algorithm, the following chromosome was considered:

$$Chromosome = (\underbrace{gene_1 \ldots gene_i}_{(1)Remove\ Facings}, \underbrace{gene_{i+1}}_{(2)Empty\ Space}, \underbrace{gene_{i+2} \ldots gene_{2i+1}}_{(3)Add\ Facings},$$

$$\underbrace{gene_{2i+2} \ldots gene_{2i+n+1}}_{(4)Sequencing}, \underbrace{gene_{2i+n+2} \ldots gene_{2i+n+u+1}}_{(5)Product\ Families\ Dimensions}, \underbrace{gene_{2i+n+u+2} \ldots gene_{2i+2n+u+1}}_{(6)Placement})$$

In which, i corresponds to the number of products, u represents the number of product families and n is the total number of blocks (the amount of products and product families combined).

4.2.1.3 Chromosome Decoding

Determine number of facings

Although the target number of facings is obtained taking into consideration the overall capacity of the planogram, there is the need to consider changes in those values. This phase determines the number of facings for each product based on three chromosome components:

- *Remove Facings*: An integer number between 1 and the number of target facings of a product is chosen, based on the component (1) from the chromosome. This number is then removed from the target number of facings. In order to give a higher probability to smaller deviations from the target number of facings, a negative exponential distribution is used as shown in Figure 4.5.
- *Empty Space* - Provides information concerning the amount of space that will be left empty on shelves. This amount of empty space is chosen, based on a gene (component (2)), between 0 and 15% of all the shelf space available. The value chosen for the highest percentage of empty space was based on the tests conducted. The decoding process is straighforward;
- *Add Facings* - Product facings are added to the products. To do this, chromosome genes from component (3) are sorted in increasing order as shown in Figure 4.6. The amount of empty space up until this point is then multiplied by the sorted chromosome keys and divided by the product's width. This results in adding more facings to products. The process is repeated until either the amount of sorted empty space to be left on shelves is fulfilled or until it is not possible to add more facings to products. The chromosome keys are sorted in increasing order so that all products have an equal chance to receive extra facings. This would not happen if new facings where always added to products in the same order, as the last products might not receive any new facings.

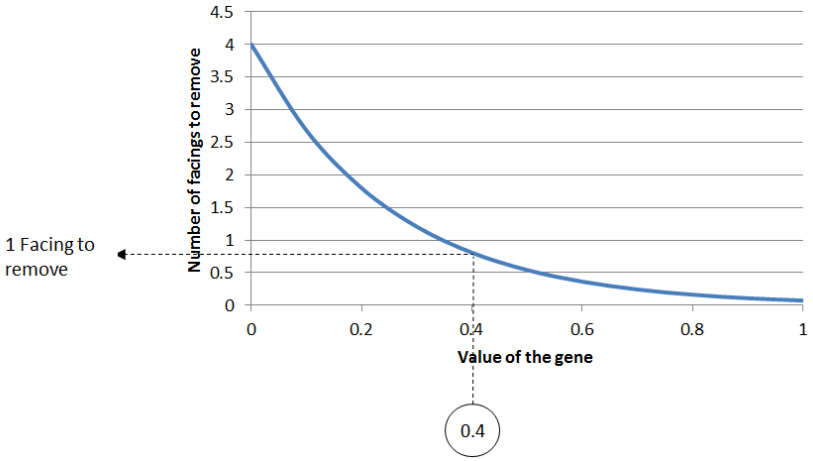

Figure 4.5: Decoding of the facings removal.

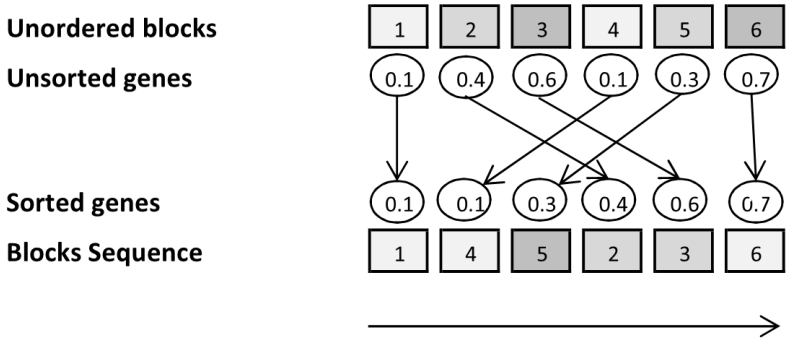

Figure 4.6: Decoding of the block sequence.

Sequencing

This phase consists of finding the sequence in which the blocks (family and products) are placed on the shelves. For that, component (4) from the chromosome is sorted in increasing order, as shown in Figure 4.6.

Note that the blocks sequence is determined separately for each level of the blocks diagram. The blocks are then placed following a top-down approach: the sequence starts with the top block and proceeds until the end of the diagram before placing the next block of the same level. For the example of Figure 4.7, the placement of blocks would follow the sequence: 1-4-9-8-5-10-2-6-11-3-7-12.

Allocate blocks

This last phase consists of allocating products on the shelves. Before that, additional information is necessary: the dimensions of the blocks and the placement strategy.

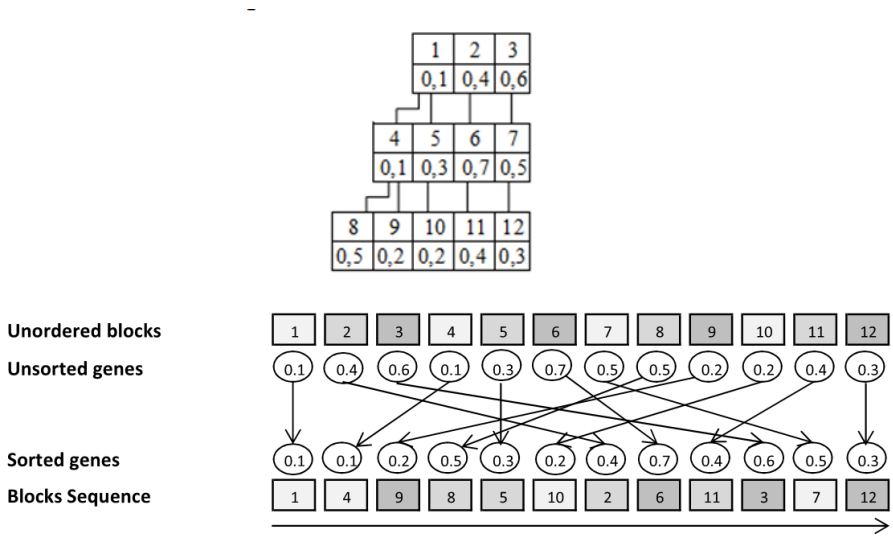

Figure 4.7: Example of blocks sequencing across levels

1st step: Determine family blocks dimensions

Starting from the first family block until the corresponding products, the following calculations are made:

- Knowing the number of products within each family blocks, the areas of all the family blocks are calculated.
- The lowest possible height of the product family (hmin) is obtained by dividing its area for the largest width allowed (given by the largest empty space within the product family that contains it);
- The highest possible height of the product family (hmax) is calculated through the division of its area and the width of the largest product belonging to the product family. In case this height exceeds the maximum height (corresponding to the height of the highest empty rectangle), then the maximum height of the product family will be equal to the maximum height allowed.

The height (h) of each family block is obtained by sorting a value between the lowest and the highest heights, based on component (5) from the cromosome, by using the formula:

$$h = (-1 + ((hmax - hmin) + 1) * chromosome\,key);$$

The value for the height is then rounded up until the next shelf. The corresponding width (w) is calculated by dividing the family block area by the randomly chosen height.

2nd step: Decide the strategy for placement

The next step consists in using component (6) from the chromosome to give information about the way family blocks are allocated. In case the chromosome key is lower or equal to 0.5, the

method used is Bottom-Left, otherwise, the method used is Left-Bottom. These methods have been explained in the previous chapter.

The allocation process starts by searching for an empty space to allocate the first family block, using a bottom-left or left-bottom approach, depending on the solution from the chromosome. If an empty space where the family block fits is found, it will be allocated in that space. Afterwards, using the Difference Process developed by Lai and Chan and explained in the previous chapter, empty rectangular spaces are updated. If it is not possible to allocate the block (no empty space where it fitted was found), it will be placed in an impossible position, so that the blocks it contains will not fit as well. All product families are allocated in the same manner, following the established sequence until products level.

Similarly to what happens with family blocks, the height of the products is calculated. In order to do that, the valid shape that has the lowest possible height is chosen for products. Valid shapes are based on the factors that can divide the number of facings (which must be integer). When this process is finished, the height and width of products' facings is defined.

The steps to allocate products are similar the ones followed for product families and include:

- Search for empty space to allocate products (Bottom-Left and Left-Bottom);
- If products fit any empty space:

  Allocate products;

  Update rectangular empty spaces through difference process.
- If products do not fit any empty space:

  Allocate products in invalid position, indicating that they were not allocated.

4.2.1.4 Fitness function

The objective function considered consists of 4 components, each having a weight higher than the next. The two first components correspond to soft constraints whose violation is penalized in the objective function. Therefore, their weight is higher than the two last components, that correspond to the real objectives of the problem.

1. Minimization of the number of products that are not allocated;
2. Minimization of the number of facings below the minimum;
3. Minimization of the deviation from the target number of facings;
4. Maximization of the impact on sales of vertical location, based on the previously described impact curve.

$$minFitness = \underbrace{\beta^4 \sum_{i=1}^{N} A_i}_{Products\,not\,allocated} + \underbrace{\beta^3 \sum_{i=1}^{N} sales_i \times M_i}_{Minimum\,number\,of\,facings} +$$

$$+\underbrace{\beta^2 \sum_{i=1}^{N} sales_i \times |targets_i - newFacings_i|}_{Deviation\ from\ targets} - \underbrace{\sum_{i=1}^{N}\sum_{k=1}^{K} sales_i \times ImpactShelf_k \times Y_{ik}}_{Impact\ of\ vertical\ location} \quad (4.1)$$

Where:

- β corresponds to the sum of target facings of all products;
- $sales_i$ represents sales of each product i on stores;
- M_i = 1 if the new facings calculated for product i are below the minimum number of facings;
- A_i = 1 if product i has not been allocated;
- $ImpactShelf_k$ corresponds to the impact on shelves of vertical and horizontal location;
- Y_{ik} =1 if product i is placed on shelf k.

4.2.2 Second BRKGA for the SSAP

This approach differs from the previous one in the phase of blocks allocation. In this algorithm, product families can only be placed either vertically or horizontally. Another difference results from the fact that after the dimensions of product families are defined, the number of facings is recalculated based on the rotation days of products, in order to fit its container family. An example of the application of this algorithm is presented in Appendix A.

4.2.2.1 Algorithm Architecture

The different phases of the decoder are presented in Figure 4.8. In the Biased Random-key Algorithm, chromosome keys are evolved and decoded, until the best solution is found. The algorithm is stopped if all products are allocated and the fitness function does not improve after a given number of generations or a time limit is reached.

The steps applied to the chromosome are the following:

(1) *Determine number of facings*

(2) *Sequencing*

(3) *Allocate Blocks:* This phase aims to allocate the blocks on the shelves, either horizontally or vertically. For that purpose, it starts by establishing whether family blocks will be placed horizontally or vertically, by using the fifth component of the chromosome. Afterwards, a sixth part of the chromosome is used to choose between Left-Bottom and Bottom-Left methodologies to place products on shelves.

(4) *Fitness Function*.

4.2.2.2 Chromosome Representation

The chromosome considered is the following:

$$Chromosome = (\underbrace{gene_1 \ldots gene_i}_{(1)Remove\ Facings}, \underbrace{gene_{i+1}}_{(2)Empty\ Space}, \underbrace{gene_{i+2} \ldots gene_{2i+1}}_{(3)Add\ Facings},$$

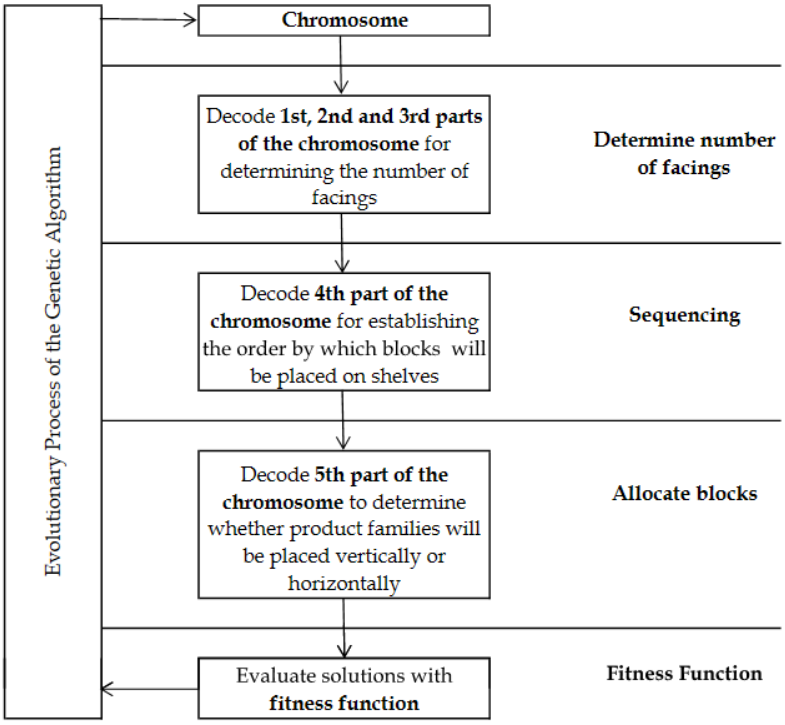

Figure 4.8: Architecture of the algorithm.

$$Chromosome = (\underbrace{gene_{2i+2}...gene_{2i+n+1}}_{(4)Sequencing}, \underbrace{gene_{2i+n+2}...gene_{2i+u+n+1}}_{(5)Orientation}, \underbrace{gene_{2i+u+n+2}...gene_{3i+u+n+1}}_{(6)Placement}$$

Where, i corresponds to number of products, n corresponds to the total number of products and u corresponds to the number of product families, with the exception of the level of product families immediately before products.

4.2.2.3 Chromosome Decoding

Since all other steps are similar to the first BRKGA decoder, only the allocation of blocks will be described.

Allocate blocks

To allocate product families on shelves, components (5) and (6) from the chromosome are used. The steps followed are the following:

- Similarly to the previous decoder, the area of product families is calculated based on the already defined product facings;
- Each product family will then be placed either vertically or horizontally within the product family that contains it. To establish the product family orientation, component (5) from

the chromosome is used. Each chromosome key corresponds to a product family (with the exception of the level of product families immediately before products).

If chromosome key < 0.5: product families will be placed vertically, and therefore, the width of the family that contains them is proportionally distributed by their areas;

If chromosome key > 0.5: product families will be placed horizontally. If the product family occupies more than one shelf, it will occupy the entire width of the family in which it is contained. On the other hand, if the family only needs one shelf, more than one family can be placed on the same shelf.

A problem that might arise results from the fact that heights are integer numbers. Due to this reason, when distributed by families, the sum of heights might exceed its maximum value. In this circumstance, if the height distributed to the last product family (according to sequencing) exceeds the maximum, it will be limited to the permitted value.

Product allocation is done in the same manner as explained in the previous decoder. This is done, because although products should preferably be placed horizontally, there are circumstances in which they can only fit by occupying more than one shelf. The strategy previously explained that uses the Bottom-Left and Left-Bottom methods, as well as the Difference Process, manages to cover all possible situations.

Chapter 5

Tests and Results

In this section, computational tests were performed in order to validate the algorithms developed. Both algorithms were implemented in C++, using the BRKGA API developed by Toso and Resende [32]. All the computational experiments were conducted on Intel @ 2.40 gigahertz processing units limited to 4 gigabytes of random access memory and using the Linux operating system. Only one thread was used for testing. The algorithms stopped if after 100 generations the fitness function had not improved and all products were allocated or a time limit of 600 seconds was reached. Instances tested are based on data from the case study whose key information was hidden for confidential purposes. The algorithms developed are tailored to the case study preferences, containing a different problem definition from those in the literature. This fact prevented their comparison with former solution alternatives. Additionally, this chapter includes a sensitivity analysis to the BRKGA parameters.

5.1 Case Study Instances

A set of real world instances was used for testing the Shelf Space Allocation Problem algorithms. These instances belong to 18 different categories and vary in size. In most cases, within each category, more than one planogram was solved, summing up to 51. In Table 5.1 the four major characteristics of the test instances are presented, namely: number of products, number of family blocks, number of hierarchical levels and number of shelves. Instances are significantly bigger than those found in literature, with up to 160 products, 216 families, 8 shelves and 5 family levels.

5.2 Parameter Sensitivity Analysis

In order to evaluate the influence of BRKGA parameters in the final results, different values for the parameters were tested. Those values were chosen within the ranges proposed by Gonçalves and Resende [4] based on their past experience with genetic algorithms using the same evolutionary strategy. The ranges and the values tested in this dissertation are presented in Table 5.2. The instances were tested with all the combinations of those values.

Table 5.1: Details of the Case Study Instances

Planogram	Products	Families	Family Levels	Shelves	Planogram	Products	Families	Family Levels	Shelves
YW1	103	11	2	5	MOV21	53	60	4	5
YW2	73	4	2	5	MOV22	78	30	4	6
S1	49	20	4	8	MOV23	21	17	4	5
Sh1	24	69	5	6	BP1	96	19	3	7
Sh2	55	142	5	6	BP2	92	23	3	5
Sh3	98	216	5	6	C1	32	16	3	7
V1	60	50	4	8	C2	25	12	3	5
V2	19	25	4	6	C3	22	15	3	5
V3	7	14	4	6	C4	19	11	3	5
V4	28	40	4	6	C5	16	16	3	5
V5	42	57	4	7	DV1	26	34	4	7
TP1	38	31	5	6	CHF1	37	44	4	6
AF1	28	25	4	7	CHF2	41	33	4	8
AF2	160	59	4	7	CHF3	40	34	4	8
OO1	32	28	3	5	CHF4	47	31	4	7
OO2	25	13	3	5	J1	31	22	4	6
OO3	10	13	3	5	J2	46	22	4	6
MO1	14	21	3	5	J3	32	20	4	6
MO2	19	24	3	5	J4	38	23	4	6
MO3	20	20	3	5	J5	19	14	4	6
CF1	36	14	4	6	CP1	12	31	4	5
CF2	19	33	4	6	CP2	8	16	4	6
CF3	113	25	4	7	CP3	9	14	4	6
CF4	83	32	4	4	CP4	39	25	4	6
VN1	45	33	3	7	CP5	47	52	4	8
F1	16	20	4	6					

Table 5.2: Values tested for BRKGA parameters

Parameters	Suggested Interval	Tested Values
Population size - p/n	≥ 1 ; $\in \Re$	1; 1.25; 1.5; 1.75; 2
Size of elite partition (p_e)	[0.10;0.25]	0.1; 0.15; 0.2
Size of mutant set (p_m)	[0.10;0.30]	0.1; 0.15; 0.2; 0.25
Child inheritance probability (ρ_e)	[0.5;0.8]	0.5; 0.6; 0.7; 0.8

Gonçalves and Resende [4] suggest and obtained good results when indexing the size of the population to the size of the problem. By doing this, small populations are obtained for small instances and larger populations for larger instances. Therefore, the parameter tested was the ratio between the population (p) and chromosome (n) size.

To analyze the sensitivity of the algorithms to these parameters, the deviation from the best objective function was calculated for each instance. This way, it became possible to compare all instances tested. Since the objective function is formed by 4 different components with different weights, the deviations in the objective function can vary widely. In particular, when the algorithm is not able to allocate all products, the value of the objective function is much higher, as this is the component with the highest weight.

The results obtained for the four main parameters are presented in Figure 5.1 - 5.8.

Population Size

Regarding the population size, by analyzing Figure 5.1 it can be seen that the variation of the population size does not have a great influence on the algorithms performance in terms of deviation from the best objective function. On the other hand, Figure 5.2 shows more detail, allowing to conclude that, although the differences are not very significant, a value of p/n=2 is the one that provides the best results. Moreover, as the value of p/n decreases, results obtained are progressively worse, until p/n=1.

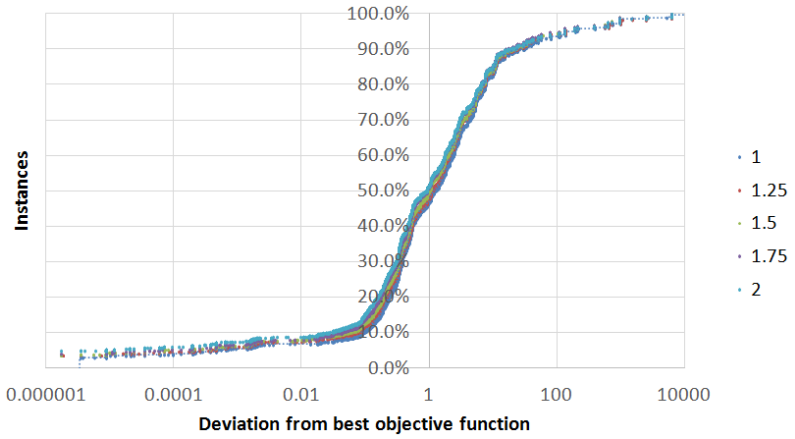

Figure 5.1: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different p/n values (logarithmic horizontal scale).

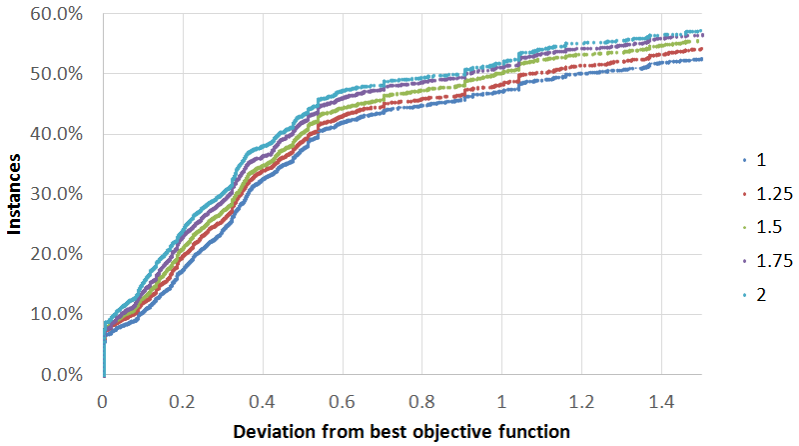

Figure 5.2: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different p/n values (normal scale).

Size of elite partition

The observation of Figure 5.3 and Figure 5.4 shows that the size of the elite partition does not influence results significantly as well. In spite of this, it is also noticeable that, as the value of p_e increases from 0.1 to 0.2, results become better.

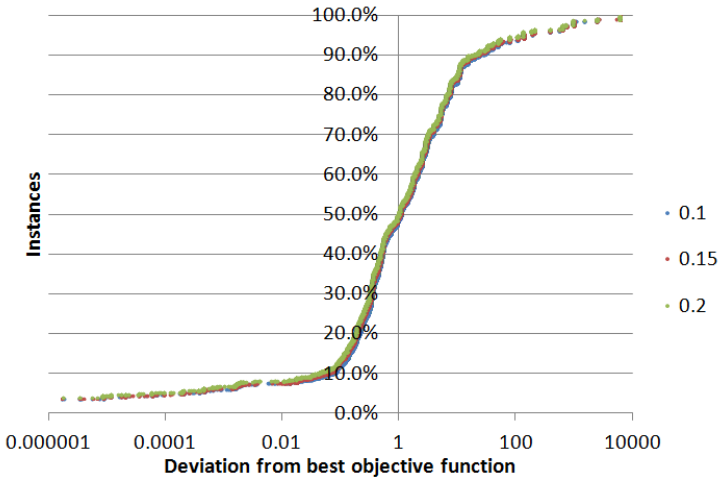

Figure 5.3: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different p_e values (logarithmic horizontal scale).

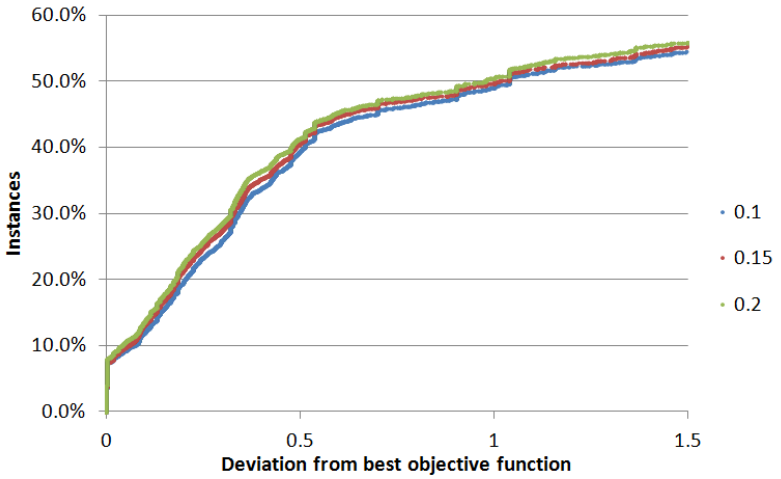

Figure 5.4: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different p_e values (normal scale).

Size of mutant set

In Figure 5.5 and Figure 5.6, the values tested for the size of the mutant set present very similar results, although a p_m value of 0.1 achieves slightly worse results. It is not possible to claim which is the best value for p_m since for smaller deviations from the best objective function p_m=0.25 seems to perform better, whereas for higher deviations $p_m = 0.2$ outperforms the previous value.

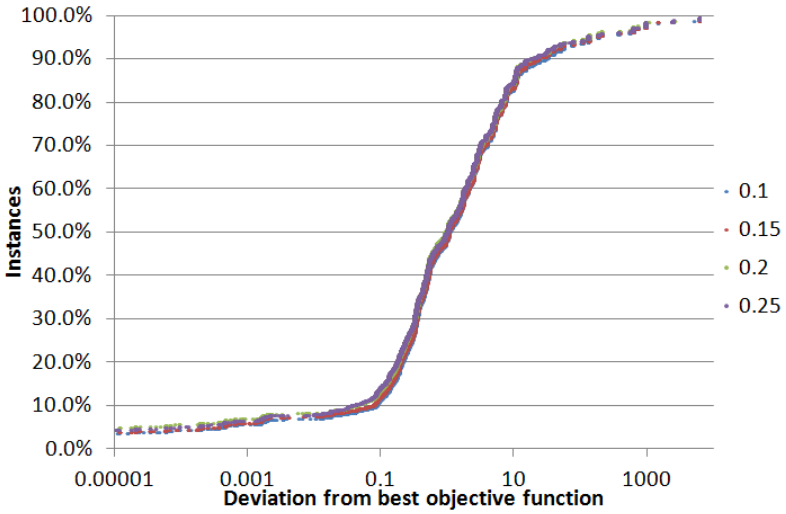

Figure 5.5: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different p_m values (logarithmic horizontal scale)

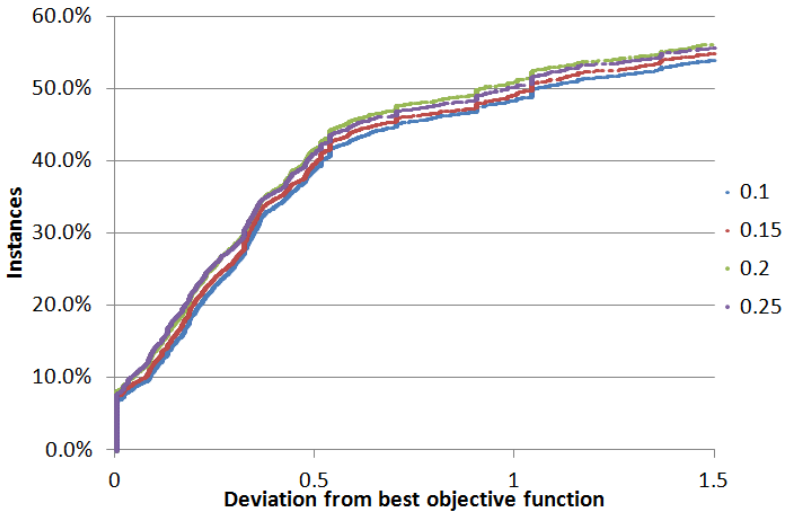

Figure 5.6: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different p_m values (normal scale).

Child inheritance probability

Analyzing Figure 5.7 and Figure 5.8, similarly to other parameters, it can be observed that the differences in results are not significant. Nevertheless, setting ρ_e to the value of 0.8 shows the worst results and results improve as the value for ρ_e decreases.

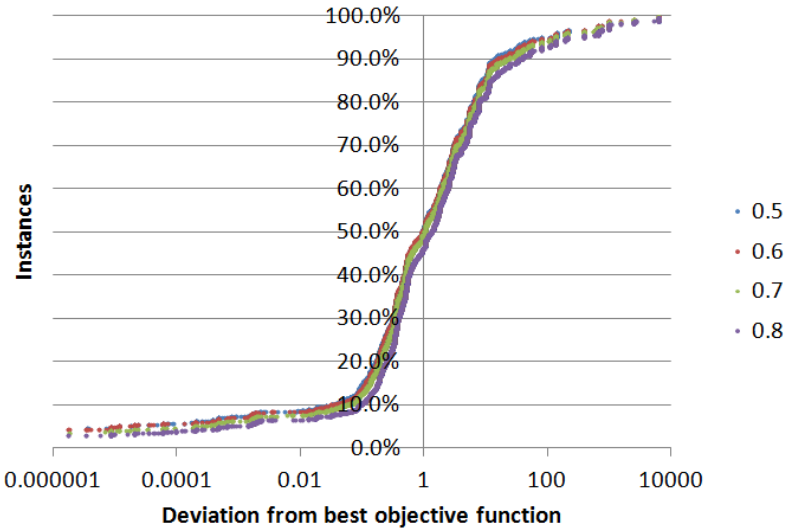

Figure 5.7: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different ρ_e values (logarithmic horizontal scale).

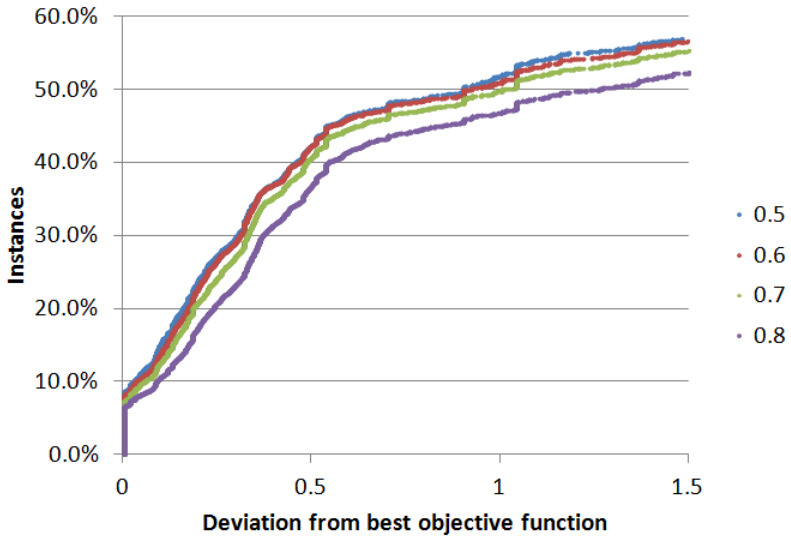

Figure 5.8: Evolution of the percentage of instances solved in function of the deviation from the best objective function of each instance, for different ρ_e values (normal scale).

Overall from the sensitivity analysis of the BRKGA parameters we can conclude that the algorithms developed are robust. The values tested for the parameters did not have a significant impact on final results, although some of the values provided slightly better results. The tests

also indicate that the child inheritance probability shows the higher impact on the algorithms performance. In the future, parameters outside the range considered could be tried, as there is a possibility of improvement.

5.3 Influence of the Planogram Complexity on the Execution Time

In the extremely competitive retail market, minimizing the time spent generating planograms may bring an important advantage to retailers.The average time the algorithms developed needed to solve each instance tested was around 90 seconds. This shows that instances were solved within a short amount of time that meets the needs of retailers in reality. To analyze the influence of the planogram complexity on the time needed to solve the problem, different analysis were made. Particularly:

- Influence of the number of families of each instance on the execution time;
- Influence of the number of products of each instance on the execution time;
- Influence of the total number of blocks of each instance on the execution time;
- Influence of shelf space available per product on the execution time;
- Influence of shelf space on the execution time.

The parameters analyzed were separated in classes, each within a certain range. Results showed that the number of products was the factor, among the parameters tested, that most influenced the time needed to solve each instance, with $R^2 = 0.98$. In Figure 5.9, products were divided in classes with a step of 25 products.

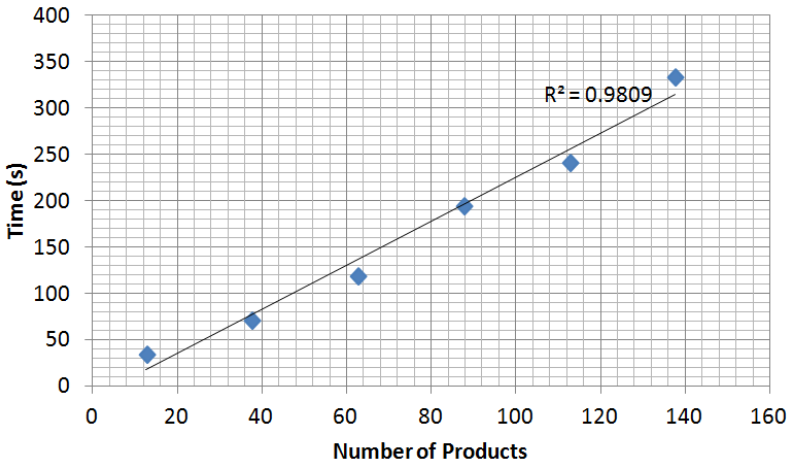

Figure 5.9: Influence of the number of products on the time needed to solve instances

5.4 Decoders Comparison

In Figure 5.10, a comparison of both algorithms (decoder 1 and 2), can be found with the results from all the instances tested. The average fitness function and the average execution time for each instance can be found in Table 5.3.

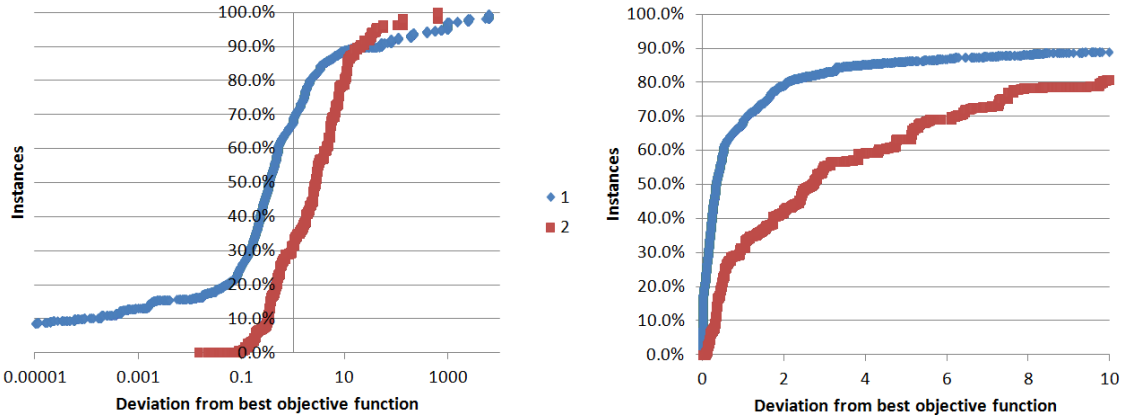

Figure 5.10: Evolution of the percentage of instances solved with the deviation from the best objective function of each instance, for decoders 1 and 2 (logarithmic horizontal scale).

The observation of Figure 5.10, shows that for smaller deviations of objective function (less than 10%), decoder 1 is able to solve a larger percentage of instances than decoder 2. This results from the fact that decoder 2 includes additional practical constraints compared to decoder 1. Surprisingly, the decoder 2 outperforms for higher deviations of the best objective function.

From Table 5.3 it can be observed that for some instances decoder 1 provides better results, whereas for others decoder 2 has better average values of fitness in shorter execution time. These differences result from the characteristics of the planograms (number of products, number of product families, shelf dimensions, among others), which may favor either decoder 1 or decoder 2. There are circumstances where the difference in the average of the fitness function is very significant between both decoders. This can be explained by the fact that the fitness function has 4 components with different weights. Therefore, cases where not all products are allocated on shelves and/or the minimum number of facings are not respected are heavily penalized resulting in significant differences in the fitness function.

Table 5.3: Average fitness and time for each decoder in the test instances

	Decoder 1		Decoder 2			Decoder 1		Decoder 2	
Planogram	Avg Fitness	Avg Time(s)	Avg Fitness	Avg Time(s)	Planogram	Avg Fitness	Avg Time(s)	Avg Fitness	Avg Time(s)
YW1	-339.52	76.17	-302.64	608.17	MOV21	0.04	10.75	0.88	7.71
YW2	0.01	22.46	2.92	598.26	MOV22	712.69	463.54	0.32	23.89
S1	402.67	332.26	5.84	14.97	MOV23	2.94	7.92	6.15	1.01
Sh1	0.16	8.80	0.40	4.41	BP1	-106.48	68.25	-90.38	218.89
Sh2	200.19	173.80	0.24	24.69	BP2	**-2718.71**	88.70	**-1915.76**	549.17
Sh3	65.32	244.91	-1.56	120.75	C1	837.79	544.02	6.35	2.16
V1	5.90	27.11	6.64	61.78	C2	6.38	2.24	12.30	0.83
V2	128.74	75.94	183.06	2.16	C3	3.17	1.29	6.63	0.73
V3	63.36	13.08	513.24	361.81	C4	13.98	40.26	15.30	0.49
V4	328.19	305.20	76.58	3.14	C5	7.73	1.39	1012.52	600.01
V5	3.16	10.34	8.90	7.55	DV1	164.57	120.80	4.83	3.05
TP1	574.37	409.13	43.54	8.38	CHF1	26.45	23.02	2.51	10.79
AF1	21.04	18.35	0.78	2.27	CHF2	25.66	28.77	3.81	10.61
AF2	-185.28	232.45	-160.12	436.16	CHF3	6.08	16.13	7.21	5.49
OO1	0.18	4.49	1.52	3.63	CHF4	317.52	261.06	12.17	13.49
OO2	-176.76	6.21	-134.09	43.13	J1	350.54	236.92	22.32	6.09
OO3	11.12	2.09	19.53	0.67	J2	4.11	9.96	74.33	43.20
MO1	1.58	9.29	1.99	0.56	J3	5.69	5.07	26.85	6.30
MO2	0.51	1.33	1.07	0.90	J4	10.48	7.67	36.96	6.09
MO3	1.07	23.82	0.67	0.86	J5	5.69	1.02	65.57	2.33
CF1	249.23	236.69	12.73	6.84	CP1	0.06	20.38	0.31	1.10
CF2	365.92	269.44	12.80	1.84	CP2	14.53	18.42	32.89	0.74
CF3	-181.00	106.00	-177.95	178.26	CP3	1.03	0.84	1.20	**0.41**
CF4	0.87	51.00	-8.87	41.80	CP4	-55.14	15.52	-44.74	28.83
VN1	529.13	354.73	51.02	6.48	CP5	-65.16	34.61	-59.35	26.64
F1	2.12	**0.65**	10.63	0.63					

Chapter 6

Conclusion and Future Work

In this chapter, the outcome of the work developed is analyzed. Suggestions of further research are also presented.

6.1 Shelf Space Considerations

Competition in retail industry is fierce and every advantage even if small can make a difference for retailers. One of the most important managerial challenges in a retail store is shelf space management. The reason for this lies in the fact that shelf space has a limited capacity and, therefore, product allocation has to be carefully managed. Moreover, experimental studies show that there are space related factors that can influence sales, particularly in circumstances of impulse purchases. Among these factors is the number of facings, the location of products on shelves (vertical or horizontal) and product adjacencies. Models from the literature, generally, do not include all constraints preferred by retailers and include parameters that are hard to estimate. Therefore, they still require improvements to become adequate for real life application. On the other hand, the software applications available for shelf space management require significant human interaction.

This dissertation is a contribution to bridge this gap between theory and practice of shelf space allocation. Inspired by the case of a major Portuguese supermarket chain, two algorithms for the Shelf Space Allocation Problem are presented.

6.2 Algorithms proposed

Two original Biased Random-key Genetic Algorithms have been developed and tested with real case study instances. Biased Random-key Genetic Algorithms have previously been successfully used to solve problems (Cutting and Packing Problems) that share similarities with the problem studied. In spite of this, BRKGAs had not yet been used to tackle the Shelf Space Allocation Problem.

The algorithms developed take into consideration preferences from a real case study, namely:

- The number of facings of a product should balance its rotation days (number of days until the product becomes out-of-stock);
- Products are grouped into multi-level families;
- Facings of the same product should be preferably placed horizontally on shelves;
- Families should be kept together in vertical or horizontal rectangular shapes on shelves;
- The minimum number of facings should be respected;
- The impact of vertical positioning on sales is considered.

The first original algorithm developed was based on the algorithm developed in [5], for the Constrained Two-Dimensional Non-Guillotine Orthogonal Cutting Problem. The decoder of the algorithm has 4 main steps: (1) Determine number of facings, (2) Sequencing, (3) Allocate Blocks and (4) Fitness Function.

The second algorithm developed is similar to the previous one, but with changes made concerning the allocation of family blocks on shelves (phase 3). In this case, product families can only be placed either vertically or horizontally.

Both algorithms were tested with real case study instances. First and foremost, results prove the applicability of the algorithms on practice. Instances were solved within an average time of 90 seconds meeting the needs of retailers in reality. In addition, novel practical constraints never considered before were included in the algorithms, such as the multi-level family grouping and the direction of implementation.

Furthermore, from the sensitivity analysis of the BRKGA parameters, we concluded that the algorithms developed are robust as the values tested for the parameters did not have a significant impact on final results. In spite of this, when observed in detail, there are parameters that perform slightly better.

We were also able to understand how the execution times are influenced by the complexity of the instances. Results show that, among the parameters tested, the execution time presents a stronger correlation with the number of products in the instance.

By analyzing the results for the decoders, we concluded that decoder 2, although with additional practical constraints, was still able to outperform decoder 1 for some of the instances.

Overall, we can argue that, concerning real applications, these algorithms are state of the art approaches that bring closer together the possibility of using analytical methods in practice.

6.3 Future Work

In spite of the algorithms developed there is still margin for improvement in the future. Two possible directions can be chosen: extending the problem definition to include additional features or improving the performance of the algorithms.

In the first direction, one possible improvement would be to consider products heights, although it is usually not necessary, since shelves are generally adjustable. The depth of facings is also not included in the algorithms due to the fact that it is not directly seen. By including it in the algorithms, the problem would become three-dimensional.

Furthermore, other additional features in the algorithms might include discontinuities in shelves and stacking of products.

To improve the algorithm's applicability in reality, changes can be made to allow retailers to choose whether they prefer product families to be placed vertically or horizontally. Other retailer preferences can also be included. Furthermore, the algorithms developed were only tested for food retail instances placed in regular shelves. The algorithm could potentially be improved in order to be able to allocate other types of products and to place products on different types of shelves.

Finally, it is considered that the choice of products to be displayed (assortment problem) has already been made. It would be interesting to incorporate the assortment problem in the algorithm as this problem is interrelated with shelf space planning.

Considering the second direction of future work, one possible approach would be to integrate the target facings calculation in the algorithm, eventually incorporating space and cross-space elasticities. Additionally, more extensive sensitivity analysis could be conducted and results could be compared to manually generated planograms.

Appendix A

Examples of Application of Algorithms Developed

In this appendix, for clarification purposes, one example of application for each of the two BRKGA decoders described in chapter 4 is presented.

A.1 Example of Application of the First BRKGA for the SSAP

To better explain the first algorithm presented in section 4.2.1, a fictional example will be described. For this example, 5 shelves are considered, each measuring 50 centimeters as presented in Figure A.1. The block diagram for this case is presented in Figure A.2 and the information related to the products can be found on Table A.1.

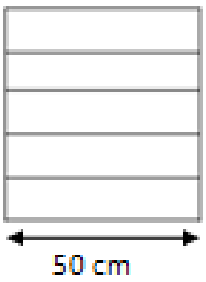

Figure A.1: Shelves considered in the example.

Table A.1: Product data considered in the example

Product	8	9	10	11	12
Width (cm)	10	10	12	12	10
Target Facings	7	2	5	9	4
Minimum Facings	1	1	1	1	1

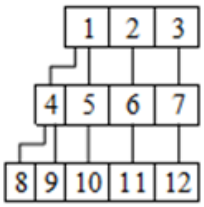

Figure A.2: Block diagram considered in the example.

For this example, an hypothetical chromosome will be considered and decoded, following the previously explained steps.

A.1.1 Determine number of facings

Facing Removal

The number of integer facings to remove from targets will be calculated according to the formula presented below.

$$Facings\ removed_i = -0.5 + (target\ facings_i - 1) \times e^{-1 \times (target\ facings_i - 1) \times chromosome\ key_i}$$

The number of facings removed is then rounded up. Chromosome keys considered belong to component (1). Resulting facings after removal are presented in Table A.2.

Table A.2: Facings of products after removal.

Product	8	9	10	11	12
Facings	7	1	4	9	4

Calculate Empty Space

The amount of space that will be left empty is randomly chosen between 0% and 15%, based on component (2) from the chromosome. It will be assumed that in this example the percentage of space that will be left empty on shelves is 1.6%.

Facing Addition

Facings are added to products, based on component (3) from the chromosome, until the percentage of space left empty is met or until it is not possible to add more facings. This component is then sorted in increasing order. This order establishes the order by which products will receive new

Table A.3: Facings of products after additions.

Product	8	9	10	11	12
Facings	8	2	4	4	5

facings. For this example, in particular, calculations are the following:

$$Space\,occupied\,by\,facings = \sum_{i=1}^{N} Target_i Width_i = 216\,cm^2$$

Where N corresponds to the number of products, $Target_i$ represents the number of target facings of product i and $Width_i$ is the width of the product facing.

$$Shelf\,space = 250\,cm^2$$

$$Space\,difference = shelf\,space - space\,occupied\,by\,facings = 34\,cm^2$$

So for each product, following the order established by component (3), the following formula is applied:

$$Facings\,added_i = Chromosome\,key_i \times Space\,difference/Width_i$$

After this process, the number of facings obtained for each product is presented in Table A.3.

A.1.2 Sequencing

After this process, products will be sequenced and allocated on shelves. Component (4) from the chromosome is used for sequencing and, for this example, it is represented in Figure A.3.

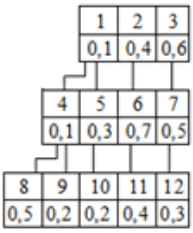

Figure A.3: Component (2) of the chromosome, used for sequencing.

According to it, the sequence that will be followed for allocation is 1-4-9-8-5-10-2-6-11-3-7-12.

A.1.3 Allocate Blocks

Calculate Areas

Table A.4: Areas of products.

Product	8	9	10	11	12
Area (cm^2)	20	80	48	48	50

The first step is to calculate the area for each block, knowing the number of facings of products that will be placed within it. For that, the height of each shelf and product is considered to be equal to 1 centimeter.

The results obtained for the areas of blocks are presented in Table A.5 and Table A.4.

Determine Block Dimensions and Allocate Blocks

Allocation begins for block 1. Next, the minimum and maximum values for the height are calculated. These values are given by the following formulas:

$$hmax = (Block\,Area)/(Largest\,Width\,of\,Empty\,Spaces\,Within\,Previous\,Block)$$

$$hmin = (Block\,Area)/(Largest\,Width\,of\,Facing\,Belonging\,to\,the\,Block)$$

If the maximum height is higher than the limit, then it will be restricted to the highest value permitted. For block 1:

- hmin=148/50=3 shelves
- hmax=148/10=14.8=5 shelves, since 15 shelves is higher than the number of shelves in the example.

To calculate the corresponding widths, the area of the block is divided by the height. Both cases of highest and lowest height are represented in Figure A.4.

In the situation of minimum height, represented on the right of Figure A.4, since the space between the block and the end of the shelf is not enough to fit any product and it is inferior to the space left empty on shelves after all block are allocated, it is considered that block 1 occupies the entire width of the shelf.

Afterwards, using component (5) from the chromosome, suppose height=3.

The resulting allocation on shelves is represented in Figure A.5.

The first block to be placed (following sequencing) is placed at the left-bottom corner of shelves. Other blocks are placed according to Bottom-Left and Left-Bottom methodologies, which are chosen based on component (6) from the chromosome. To update empty rectangular spaces,

Table A.5: Areas of product families.

Product	1	2	3	4	5	6	7
Area (cm^2)	148	48	50	100	48	48	50

Figure A.4: Example of maximum and mininum heights for block 1.

the Difference Process by Lai and Chan is used. The process is similar for all product families. To allocate products, on the other hand, it is given a preference for horizontal allocations. Due to this reason the minimum height that can fit an empty space within a block is chosen for the product. It is also taken into consideration the fact that facings are integer and cannot be divided to fit shelves. Left-Bottom, Bottom-Left and the Difference Process methodology are also used in product allocation. The result after allocating all blocks is represented in Figure A.6.

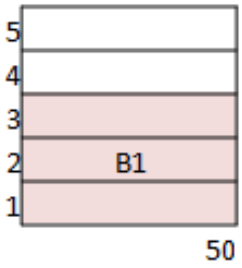

Figure A.5: Allocation of block 1.

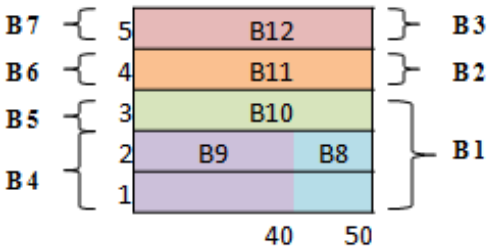

Figure A.6: Allocation of all blocks.

A.1.4 Fitness Function

The solution is then evaluated according to the fitness function previously presented.

This algorithm resembles the algorithm used in [5]. Among the most significant differences, is the fact that in the Shelf Space Allocation Problem the dimensions of rectangles (blocks) are unknown and depend on the number of facings of each product and on their arrangement on shelves. Due to this, the first thing done in the algorithm is to establish blocks dimensions which originates multiple problems of packing rectangles within other rectangles.

A.2 Example of Application of the Second BRKGA for the SSAP

To better illustrate the algorithm a simple example will be presented. This example is not based on reality and its only purpose is to provide further clarity.

For this example, there are 5 shelves with a width of 50 centimeters, as presented in Figure A.1. The block diagram under consideration is presented in Figure A.7. Furthermore, information concerning products is present in Table A.6.

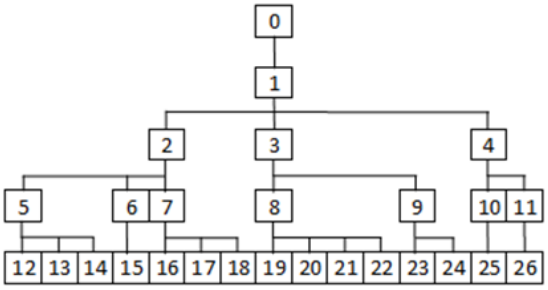

Figure A.7: Block diagram for the example.

Table A.6: Product data

Product	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Width (cm)	17.5	17.5	2.5	20	2.5	2.5	15	10	10	10	10	10	10	5	5
Target Facings	1	1	2	2	1	1	1	1	1	1	1	3	1	4	5
Minimum facings	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

A.2.1 Determine number of facings

The first step is to determine the number of facings. Since this process is the same as in the first decoder, only the results will be presented, on Table A.7.

Table A.7: New facings obtained for products

Products	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Facings	1	1	2	1	2	2	2	1	1	1	1	2	1	5	5

A.2.2 Sequencing

For this example, chromosome keys for sequencing (component (4)) are represented in Figure A.8.

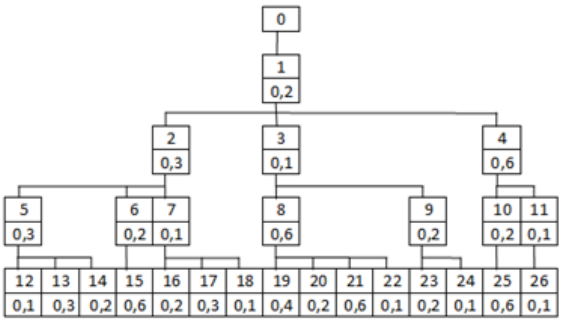

Figure A.8: Chromosome keys for sequencing

A.2.3 Allocate Blocks

Calculate Areas

First, the area of blocks is calculated. For this, it is considered that each shelf and product has an height of 1 centimeter. The areas of blocks are presented in Table A.8, Table A.9 and Table A.10

Table A.8: Areas of blocks and products within block 2

Blocks	12	13	14	5	15	6	16	17	18	7	2
Area (cm^2)	17.5	17.5	5	40	20	20	5	5	30	40	100

Table A.9: Areas of blocks and products within block 3

Blocks	19	20	21	22	8	23	24	9	3
Area (cm^2)	10	10	10	10	40	40	20	60	100

Table A.10: Areas of blocks and products within block 4

Blocks	25	10	26	11	4
Area (cm^2)	25	25	25	25	50

The area of block 1 corresponds to the areas of blocks 2, 3 and 4 combined and therefore is equal to 250 cm^2.

Orientation and Allocation of Blocks

Chromosome keys for orientation (component (5)) are represented in Figure A.9.

Block 0: chromosome key (component (5)) = 0.3 < 0.5 => block 1 will be placed vertically on shelves.

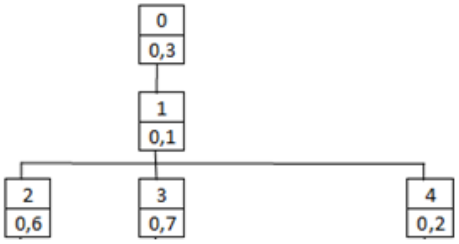

Figure A.9: Chromosome keys for block orientation

Total area of block 1 is 250 cm^2, which means it will occupy the entirety of the shelf's width (50 cm) and height (5 cm). Representation of the placement of block 1 on shelves if presented in Figure A.10.

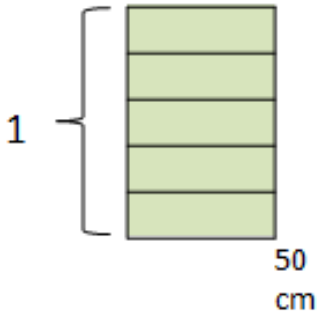

Figure A.10: Placement of block 1 on shelves.

Block 1: chromosome key (component (5)) = 0.1 <0.5 => blocks 2,3 and 4 will be placed vertically on shelves.

Total area of blocks 2,3 and 4 combined is also 250 cm^2 (100+100+50). For each block, the proportion of space it will occupy is the following:

Block 2: 100/250=0.4

Block 3: 100/250=0.4

Block 4: 50/250=0.2

Since these blocks will be placed vertically, their measurements will be:

Width block 2=0.4*(width previous block)=0.4*50=20 cm

Height block 2= height previous block=5

Width block 3=0.4*(width previous block)=0.4*50=20 cm

Height block 3= height previous block=5

Width block 4=0.2*(width previous block)=0.2*50=10 cm

Height block 4= height previous block=5

Blocks will be placed according to the order given by component (4) from the chromosome. In this case, it will be 3,2,4. The placement of these blocks on shelves is represented in Figure A.11.

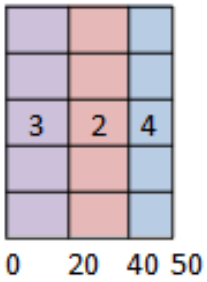

Figure A.11: Placement of blocks 2,3 and 4 on shelves.

Block 2: chromosome key (component (5)) = 0.6 $\geq$ 0.5 => blocks 2,3 and 4 will be placed horizontally on shelves.

The total area of blocks 5, 6 and 7 is 100 cm^2 (40+20+40). Accordingly, the proportions of space they occupy are the following:

Block 5: 40/100=0.4

Block 6: 20/100=0.2

Block 7: 40/100=0.4

Considering they will be placed horizontally their measurements are the following:

Height block 5=0.4*(height previous block)=0.4*5=2

Width block 5= width previous block=20 cm

Height block 6=0.2*(height previous block)=0.2*5=1

Width block 6= width previous block=20 cm

Height block 7=0.4*(height previous block)=0.4*5=2

Width block 7= width previous block=20 cm

It is necessary to verify whether the sum of heights does not exceed the maximum height value. In case this happens, the last blocks to be placed will have reductions in their height to meet requirements. Also, in this case, due to the dimensions of product families, only one can be placed per shelf.

The sequencing of the placement of blocks, according to component (4) of the chromosome is: 7,6,5, as represented in Figure A.12.

The process is repeated for all product families, until the level of products. To place products, the methods followed are similar to the ones presented in the previous decoder. One difference is that after determining the dimensions of product families, facings of products are recalculated based on the rotation days, to fit the available space. Doing this, helps minimize the deviation from target facings. Product placement, for this example is represented in Figure A.13.

A.2.4 Fitness Function

Afterwards, the solution is evaluated by the fitness function previously explained.

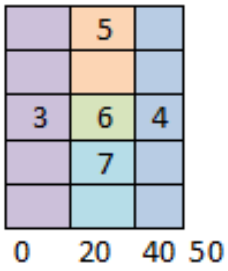

Figure A.12: Placement of blocks 5,6 and 7 on shelves.

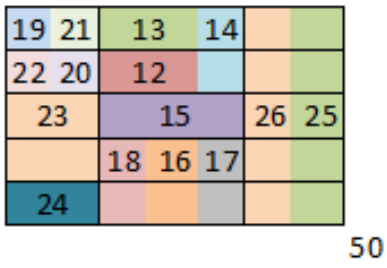

Figure A.13: Placement of products on shelves.

References

[1] Alexander H Hübner and Heinrich Kuhn. Retail category management: State-of-the-art review of quantitative research and software applications in assortment and shelf space management. *Omega*, 40(2):199–209, 2012.

[2] Teresa Bianchi de Aguiar. The retail shelf space allocation problem: New optimization methods applied to a supermarket chain (PhD proposal). *Faculty of Engineering, University of Porto*, 2012.

[3] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, volume 74. John Wiley & Sons, 2009.

[4] José Fernando Gonçalves and Mauricio GC Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011.

[5] JF Gonçalves and MGC Resende. A hybrid heuristic for the constrained two-dimensional non-guillotine orthogonal cutting problem. *AT&T Labs Research Technical report TD-&UNQN6*, 2006.

[6] KK Lai and Jimmy WM Chan. Developing a simulated annealing algorithm for the cutting stock problem. *Computers & industrial engineering*, 32(1):115–127, 1997.

[7] Pierre Chandon, J Wesley Hutchinson, Eric T Bradlow, and Scott H Young. Does in-store marketing work? Effects of the number and position of shelf facings on brand attention and evaluation at the point of purchase. *Journal of Marketing*, 73(6):1–17, 2009.

[8] Ruibin Bai. *An investigation of novel approaches for optimising retail shelf space allocation*. PhD thesis, University of Nottingham, 2005.

[9] Associação Portuguesa de Empresas de Distribuição. Ranking APED 2011, 2012. Available in http://www.aped.pt/Media/content/348_1_G.pdf, last access May 10th, 2014.

[10] Xavier Dreze, Stephen J Hoch, and Mary E Purk. Shelf management and space elasticity. *Journal of Retailing*, 70(4):301–326, 1995.

[11] Andrew Lim, Brian Rodrigues, and Xingwen Zhang. Metaheuristics with local search techniques for retail shelf-space optimization. *Management Science*, 50(1):117–131, 2004.

[12] Mümin Kurtulus and Alper Nakkas. Retail assortment planning under category captainship. *Manufacturing & Service Operations Management*, 13(1):124–142, 2011.

[13] Robert A Russell and Timothy L Urban. The location and allocation of products and product families on retail shelves. *Annals of Operations Research*, 179(1):131–147, 2010.

[14] Chase C Murray, Debabrata Talukdar, and Abhijit Gosavi. Joint optimization of product price, display orientation and shelf-space allocation in retail category management. *Journal of Retailing*, 86(2):125–136, 2010.

[15] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.

[16] Ming-Hsien Yang and Wen-Cher Chen. A study on shelf space allocation and management. *International journal of production economics*, 60:309–317, 1999.

[17] José Fernando Gonçalves and Mauricio GC Resende. A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500–510, 2013.

[18] Ronald C Curhan. The relationship between shelf space and unit sales in supermarkets. *Journal of Marketing Research*, pages 406–412, 1972.

[19] Ronald C Curhan. Shelf space allocation and profit maximization in mass retailing. *Journal of Marketing*, 37(3), 1973.

[20] Mehmet E Coskun. Shelf space allocation: A critical review and a model with price changes and adjustable shelf heights. 2012.

[21] Marcel Corstjens and Peter Doyle. A model for optimizing retail space allocations. *Management Science*, 27(7):822–833, 1981.

[22] Jens Irion, Faiz Al-Khayyal, and Jye-Chyi Lu. A piecewise linearization framework for retail shelf space management models. *Retrieved May*, 6:2006, 2004.

[23] Hark Hwang, Bum Choi, and Min-Jin Lee. A model for shelf space allocation and inventory control considering location and inventory level effects on demand. *International Journal of Production Economics*, 97(2):185–195, 2005.

[24] Jared M Hansen, Sumit Raut, and Sanjeev Swami. Retail shelf allocation: A comparative analysis of heuristic and meta-heuristic approaches. *Journal of Retailing*, 86(1):94–105, 2010.

[25] Ming-Hsien Yang. An efficient algorithm to allocate shelf space. *European journal of operational research*, 131(1):107–118, 2001.

[26] Hasmukh K Gajjar and Gajendra K Adil. Heuristics for retail shelf space allocation problem with linear profit function. *International Journal of Retail & Distribution Management*, 39(2):144–155, 2011.

[27] Mauro Castelli and Leonardo Vanneschi. Genetic algorithm with variable neighborhood search for the optimal allocation of goods in shop shelves. *Operations Research Letters*, 2014.

[28] John H Holland. *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[29] José Fernando Gonçalves, Mauricio GC Resende, and Miguel Dias Costa. A biased random-key genetic algorithm for the minimization of open stacks problem. 2013.

[30] José Fernando Gonçalves, Mauricio GC Resende, and Jorge JM Mendes. A biased random-key genetic algorithm with forward-backward improvement for the resource constrained project scheduling problem. *Journal of Heuristics*, 17(5):467–486, 2011.

[31] James C Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160, 1994.

[32] Rodrigo F. Toso and Mauricio GC Resende. A c++ application programming interface for biased random-key genetic algorithms. Technical report, Technical report, AT&T Labs Research, Florham Park, NJ, 2012.

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