Optimizing Shelf Space Allocation under Merchandising Rules

Luís Fernando Moreira Souto

Master's Dissertation

Supervisor: Prof. José Fernando Oliveira

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mestrado Integrado em Engenharia e Gestão Industrial

2018-07-25

Abstract

The allocation of products to shelves is a challenging problem that retailers are faced with on a regular basis. With retail being an industry known for its strong competitiveness and low profit margins, a proper placement of products becomes of paramount importance, as it may constitute a source of competitive advantage.

In retail environments, it is common for products to be grouped into product families (blocks), according to a particular criterion. Additionally, a set of merchandising rules specifies how the different product families should be arranged on the shelves, which gives rise to a complex multi-level hierarchical structure that reflects the retailer's preferences. The purpose of these rules is to improve in-store experience by trying to reproduce the way consumers search for products while shopping, thus enhancing the shelves' attractiveness.

These rules lead to a highly constrained Shelf Space Allocation Problem (SSAP), as the location of a particular product on the shelf is not independent from the location of other products belonging to the same product family. This dissertation proposes a heuristic approach to this problem, consisting of a constructive and an improvement stage.

The constructive procedure is comprised of problem-specific heuristics that explore the problem's inherent hierarchical nature, with the actual allocation of products only taking place after all product families have been assigned a specific place on the existing set of shelves. The initial solution is then improved by applying the Iterated Local Search (ILS) metaheuristic, which combines local search techniques with perturbation mechanisms. While the former explore a specific region of the solution space, the latter are employed in order to escape from locally optimal solutions.

The developed algorithms were tested on a set of benchmark instances from a European grocery retailer. When compared with state-of-the-art methods, this approach led to an increase in the number of feasible solutions and shorter computation times. The methods proposed in this thesis could be used by retailers to increase both the efficiency and the effectiveness of their planogram generation processes.

Resumo

A alocação de produtos nas prateleiras é um problema complexo com o qual os retalhistas são regularmente confrontados. Atendendo a que o retalho é um mercado extremamente competitivo e com baixas margens de lucro, um posicionamento adequado dos produtos torna-se ainda mais importante, uma vez que poderá constituir uma vantagem competitiva.

Em ambientes de retalho, é frequente agrupar os diferentes produtos em famílias de produtos (blocos), de acordo com um determinado critério. Além disso, existe um conjunto de regras de merchandising que especifica como é que as diferentes familias de produtos deverão estar dispostas nas prateleiras, o que dá origem a uma estrutura multi-nível complexa que reflete as preferências do retalhista. Estas regras visam melhorar a experiência do consumidor, tentando reproduzir o modo como este procura por um determinado produto enquanto faz as suas compras, e desta forma tornar as prateleiras mais atrativas.

Estas regras conduzem a um Problema de Alocação dos Produtos nas Prateleiras (SSAP) altamente restringido, uma vez que a localização de um determinado produto na prateleira não é independente da localização dos outros produtos pertencentes à mesma família. Esta dissertação propõe uma abordagem heurística a este problema, com uma fase construtiva e uma fase de melhoramento.

A fase construtiva é composta de heurísticas específicas ao problema em causa, que exploram a natureza hierárquica que lhe é inerente. A alocação dos produtos só é efetuada depois de atribuir a cada família de produtos um espaço específico no conjunto de prateleiras disponíveis. A solução inicial é depois melhorada através da utilização da meta-heurística *Iterated Local Search* (ILS), que conjuga métodos de busca local (*Local Search*) com mecanismos de perturbação. Enquanto que os primeiros exploram uma região específica do espaço de soluções, os segundos são usados com o intuito de escapar de ótimos locais.

Os algoritmos desenvolvidos foram testados num conjunto de instâncias de referência de um retalhista alimentar Europeu. Quando comparada com os métodos mais avançados na área, esta abordagem conduziu a um aumento do número de soluções admissíveis e tempos de computação mais reduzidos. Os métodos propostos nesta tese podem ser utilizados por retalhistas por forma a aumentar a eficiência e eficácia dos seus processos de geração de planogramas.

Acknowledgments

I would like to start by thanking my supervisor Prof. José Fernando Oliveira, whose guidance and insights greatly improved the quality of this dissertation.

I also want to thank Dr. Teresa Bianchi de Aguiar, who introduced me to the topic and closely followed the progress of this work. Not only were her suggestions always very pertinent, but also her constant enthusiasm and encouragement were definitely a source of inspiration for me, for which I cannot thank enough.

I am extremely grateful for the opportunity of working at LTPlabs. It is for sure an amazing company, made up of amazing people, to whom I owe most of my personal and professional development over the last few months.

To all my lifelong friends from my hometown, and to the incredible friends I made during these five years of university, thank you for your everlasting support and for all the great moments that we shared together and that I so fondly remember.

Finally, I owe a very special "thank you" to my family, particularly to my parents and my brother. Your invaluable advice and infinite patience are completely irreplaceable. Thank you for believing in my capabilities and for always pushing me to do my best. This thesis would have never been possible without you.

"Man is not infallible. He searches for truth – that is, for the most adequate comprehension of reality as far as the structure of his mind and reason makes it accessible to him. Man can never become omniscient. He can never be absolutely certain that his inquiries were not misled and that what he considers as certain truth is not error. All that man can do is to submit all his theories again and again to the most critical reexamination."

Ludwig von Mises

Contents

Acronyms and Symbols

BAP	Block Allocation Problem
BKP	Bounded Knapsack Problem
DAG	Directed Ayclic Graph
DP	Dynamic Programming
ILS	Iterated Local Search
ILS-B	Iterated Local Search using the *Better* acceptance criterion
ILS-RW	Iterated Local Search using the *Random walk* acceptance criterion
ILS-LSMC	Iterated Local Search using the *LSMC* acceptance criterion
KP	Knapsack Problem
MINLP	Mixed Integer Nonlinear Programming
SA	Simulated Annealing
SKU	Stock Keeping Unit
SSAP	Shelf Space Allocation Problem
SSRP	Shelf Space Replication Problem
SWO	Squeaky-Wheel Optimization
TS	Tabu Search

List of Figures

List of Tables

Chapter 1

Introduction

The present chapter serves as the starting point for this dissertation and is divided into four sections. Section 1.1 sets up the motivation for this work, Section 1.2 defines the scope of the problem under analysis, and Section 1.3 presents an overview of the methodology that was followed to tackle the problem. Finally, Section 1.4 provides a brief description of each chapter of the present dissertation.

1.1 Motivation

The retail industry, whose total global sales are expected to amount to $27.7 trillion (USD) in 2020 (eMarketer, 2018), is well-known for its fierce competitive environment and constantly changing dynamics. As a result, retailers are always seeking for new ways of staying ahead of their direct competitors and thus maximize their profits. The recent years' trend towards the use of analytical techniques to support both strategic and operational decision-making stemmed precisely from this tireless effort of obtaining new sources of competitive advantage (Fisher, 2009).

The allocation of products to shelves is a challenging problem that retailers face on a regular basis. With shelf space being a scarce resource in retail stores and given the trend towards the increase in the number of products included in the assortment (Hübner and Kuhn, 2012), the importance of making appropriate allocation decisions becomes clearer. Furthermore, a clever arrangement of products may also increase demand, as customers' purchase decisions are frequently influenced by in-store factors (Bianchi-Aguiar, 2015). This is particularly the case in situations where products are low-priced and impulse purchases are frequent (e.g. grocery stores). In those cases, assigning the right amount of space and finding the best location for each product on the shelf is of crucial importance, since the impact of these decisions may constitute a differentiating factor for choosing one particular product over another. Finally, shelf space planning has a strong impact on in-store operations and, in particular, on the efficiency of shelf replenishment processes, as explained by Hübner and Schaal (2017c). Hence, good shelf space management practices, assisted by decision support systems and software applications, can enhance the store's overall profitability.

1.2 Scope

In retail environments, it is common to differentiate between two different levels of planning: the store (*macro*) level, where each product category is assigned a particular space in the store (long-term planning horizon); and the category (*micro*) level, in which individual products are allocated on the shelves (mid-term planning horizon) (Bianchi-Aguiar, 2015). The present dissertation focuses exclusively on micro space planning. Hence, within each product category, our goal is to find the optimal allocation of a set of products on a set of shelves. This implies deciding the appropriate shelf space to assign to each product, as well as its horizontal and vertical location. In the literature, this problem is normally referred to as the Shelf Space Allocation Problem (SSAP).

1.3 Methodology overview

In this thesis, a particular formulation of the SSAP that was introduced by Bianchi-Aguiar et al. (2018b) is tackled. In their approach, the arrangement of products on the shelves is contingent on a set of merchandising rules that specify that certain groups of products must be placed together on the shelves while forming rectangular shapes, which can either be vertically or horizontally arranged.

To first address the problem, a problem-specific constructive heuristic was designed and implemented. As it will be shown in the following chapters, this is a highly constrained problem, which stresses the importance of obtaining high quality initial solutions. To improve the initial solution, an Iterated Local Search (ILS) framework was applied, combining local search techniques with perturbation mechanisms that are employed to escape from locally optimal solutions. By using this procedure, we are able to stop its execution at any time and obtain a solution which is, in the vast majority of the cases, of satisfactory quality.

Finally, our method was tested on a set of benchmark instances and the results were compared with the ones obtained using state-of-the-art methods.

1.4 Dissertation structure

This dissertation is divided into seven chapters, the first of which being the present chapter, which presented an introduction to the topic of shelf space allocation. The remainder can be outlined as follows.

Chapter 2 provides a contextualization of the challenges that arise within the context of shelf space management in retail environments, as well as a brief high-level perspective on the Shelf Space Allocation Problem, including some important definitions that will be used throughout this thesis.

In Chapter 3, a literature review is conducted. First, some empirical and qualitative approaches to shelf space management are presented. Then, a clear focus is given to works that make use of

optimization methods to solve the SSAP, including a few number of studies that integrate shelf space allocation decisions with other planning problems that frequently emerge in retail.

Chapter 4 presents the formulation of the problem under analysis and constitutes the starting point for the heuristic approach.

Chapter 5 provides a comprehensive description of the heuristic methods that were implemented to solve the SSAP, including both the algorithms that make up the constructive procedure and the ILS framework.

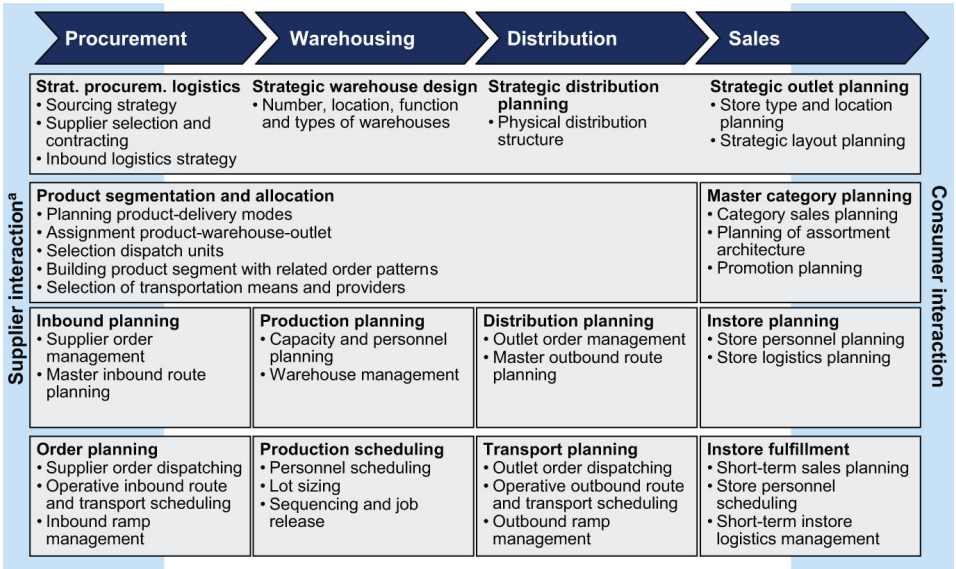
Chapter 6 contains the set of computational experiments that were performed in order to assess the performance of the methods described in the previous chapter.

Finally, Chapter 7 unveils the main conclusions that can be drawn from the present dissertation and indicates opportunities for improvement in future works that deal with the SSAP.

Chapter 2

Retail Shelf Space Management

Decision-making processes within the retail industry can be highly complex, given the strong interdependencies that exist between decisions that are made at different hierarchical levels of planning. Despite this, however, most businesses have adopted a functionally-oriented perspective, dividing the company into the so-called "functional silos" (Ensor, 1988), each of which being typically associated with a particular business function. Hübner et al. (2013) indicate the pitfalls of such an approach and propose a comprehensive framework for integrating the different planning challenges within retail, which is presented in Figure 2.1.



Note: [a]including supplier-relationship management and collaboration

Figure 2.1: Retail demand and supply chain planning framework proposed by Hübner et al. (2013)

As indicated by the authors, the problem of allocating products to shelves falls into the category of *Master category planning*, which is sales-oriented and generally considers a mid-term planning horizon. Category management has become an increasingly relevant topic since the late 1980s, when retailers started to group similar products into categories that were managed as separate business units (Kurtuluş and Nakkas, 2011). Hübner and Schaal (2017a) emphasize the close

relationship between shelf space planning and other category planning processes, namely, assortment planning (where the decision of which products should be displayed on the shelves is made) and in-store replenishment planning (which determines the appropriate shelf refilling policies).

Provided this initial contextualization, Section 2.1 will present the current practices of shelf space management, while Section 2.2 provides an overview of the Shelf Space Allocation Problem and the challenges that it presents to retailers.

2.1 Current practices in Shelf Space Management

In the retail environment, shelf space allocation is usually preceded by the assortment planning. As previously mentioned, there are strong interdependencies between these two decisions, as providing a broader assortment may hinder product availability due to the limited space that can be attributed to each item. Besides determining how much space should be allocated to each of the products in the assortment, retailers must also decide their vertical and horizontal location on the shelves. It should be stressed that these decisions are far from being trivial, for the reasons that will be detailed next.

At first, it could appear reasonable for retailers to allocate shelf space according to the sales contribution of each item (i.e. the items with the highest amount of sales would be assigned the largest space, and vice versa). However, as noted by Within (1957) over half a century ago, this is a simplistic approach that neglects the fact that inventory (in this case, the inventory that is displayed to the customer) and sales are not independent from each other. In the decades that followed, several authors presented empirical evidence that supported Within's statement, showing that an increase in the amount of shelf space that is allocated to a particular product (which consequently causes an increase in the displayed inventory) could stimulate consumer demand for that product (e.g. Wolfe (1968), Curhan (1972), Drèze et al. (1994), Desmetab and Renaudinb (1998), Koschat (2008), and Eisend (2014)). Consistent with this idea, Larson and DeMarais (1990) introduced the concept of "psychic stock" as being the display inventory that is specifically carried for the purpose of stimulating demand. To account for diminishing returns, Drèze et al. (1994) used an S-shaped curve when modeling demand as a function of shelf space, as shown in Figure 2.2.

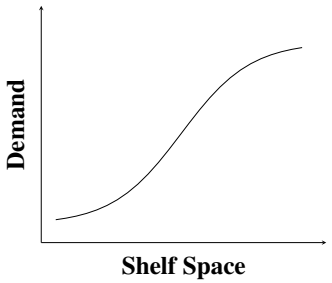


Figure 2.2: Effect of shelf space on product demand

In addition to shelf space, the location of a product may also influence its demand (Bai, 2005), as shelves that are located at the eye level are more easily noticeable by customers. In fact, Drèze

et al. (1994) concluded that the vertical location of an item had a significantly higher impact on its demand than the attributed space. Despite also having studied the impact of the horizontal location, the authors reach no consensus as to which location is the best for capturing the customers' attention.

Apart from the aforementioned two-way relationship between shelf space allocation and demand, there is an additional factor that increases the complexity of the problem. As Bianchi-Aguiar (2015) mentions, the arrangement of products on shelves is not arbitrary, as it is common practice for retailers to define merchandising rules that specify that certain groups of products should be placed together on shelves. These groups are typically composed of products that share one or more attributes (such as color, brand, or type). These rules can also assign specific locations to some products or establish a given sequence. Ebster and Garaus (2011) described visual merchandising as being the "language of the store", since it serves as a means for communicating with the customer through product presentation. Grouping products according to a particular criterion facilitates the search for an item on the shelf and may even boost the store's total profit, either by encouraging the purchase of complementary products or by displaying premium brands in more attractive locations (Varley, 2006).

Planograms are an indispensable tool for an effective shelf space planning. They consist in a visual representation of the products' placement on the shelves. Figure 2.3 shows an example of a planogram, with the products highlighted according to their type.

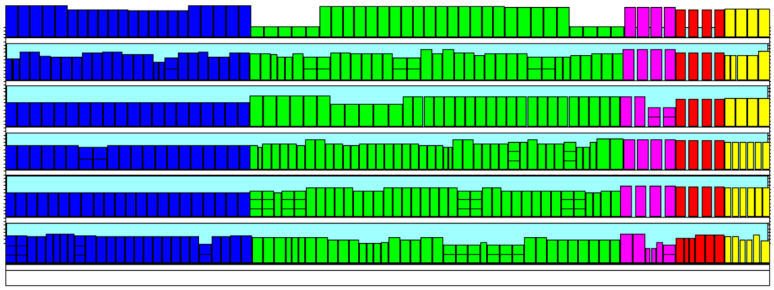


Figure 2.3: A planogram with the products highlighted by type

For large retailers, generating planograms for each store may be an overwhelming task, as variations in assortment and available shelf space between different stores are always to be expected. In a collaboration with the leading Portuguese food retailer, Bianchi-Aguiar et al. (2016) developed a methodology to handle the inherent complexity of generating store-specific planograms. In their approach, stores were aggregated into clusters according to their similarities in terms of assortment, sales, and space patterns. Then, *role* planograms were automatically generated for each cluster of stores and then validated by the space managers, that could perform small manual adjustments if necessary. Finally, these role planograms were *replicated* into store-specific planograms, that attended to each store's specific characteristics while following the products' arrangement dictated by the corresponding role planogram. This process of creating two different "layers" for the planogram generation process can prove itself useful for large retailers to enhance compliance with the designed planograms.

Regarding software applications that support shelf space management, Hübner and Kuhn (2012) reported a fair degree of IT adoption among retailers, with only 30% of respondents stating that they did not possess any IT support, nor were they planning to invest in that area in a near future. Among the advantages of these applications, the authors refer the creation of realistic planograms within reasonable computation times. Notwithstanding the efficiency of the planogram generation processes, commercial software solutions also have their limitations. As some authors point out (e.g. Hansen et al. (2010), Hübner and Kuhn (2012), Irion et al. (2012)), these solutions do not serve as optimization tools, as they mainly rely on simplistic heuristics in order to generate planograms, thus leading to suboptimal solutions in terms of shelf space allocation. Hence, there is still room for improvement in this field, particularly with respect to bridging the gap between shelf space theory and retail practices.

2.2 The Shelf Space Allocation Problem

The Shelf Space Allocation Problem consists in finding the optimal placement of a set of products on a set of shelves. It is a combinatorial optimization problem which is, in general, NP-Hard (Bianchi-Aguiar, 2015). It bears a close relationship with other combinatorial problem that are frequently addressed in the literature, such as the knapsack problem and the bin packing problem (Bai, 2005).

2.2.1 Key definitions

In the present section, some concepts that are related with shelf space allocation problems will be introduced. These definitions are mainly based on the works of Janeiro (2014) and Bianchi-Aguiar (2015) and are summarized in Table 2.1.

Table 2.1: Important concepts in shelf space allocation problems

Stock Keeping Unit (SKU)	Unique identifier that refers to a given product which is kept on stock. The terms SKU, *product*, and *item* will be used interchangeably throughout the present dissertation
Product family	Group of products that share a particular attribute
Merchandising rules	Set of rules that products must follow when placed on the shelves for visual merchandising purposes. These rules may define, for instance, specific product groupings, sequencing, or special display locations
Facing	Unit of a given product which is placed on the shelf. The multiplication of the number of facings *wide*, *high*, and *deep* yields the total inventory that is displayed on the shelf, as suggested by Figure 2.4
Shelf level	Horizontal surface where the items are placed. The terms *shelf level* and *shelf* will be used interchangeably throughout the present dissertation
Space elasticity	For a given product, measures the corresponding sales sensitivity to a change in the allocated shelf space
Location elasticity	For a given product, measures the corresponding sales sensitivity to a change in its placement (horizontal and/or vertical) on the shelves
Cross-space elasticity	For a given product, measures the corresponding sales sensitivity to a change in the space allocated to adjacent product(s)
Master planogram	Planogram that applies to a cluster of stores that share strong similarities in terms of assortment, sales, and space patterns
Store-specific planogram	Planogram that applies only to a specific store, considering its assortment, sales, and available shelf space

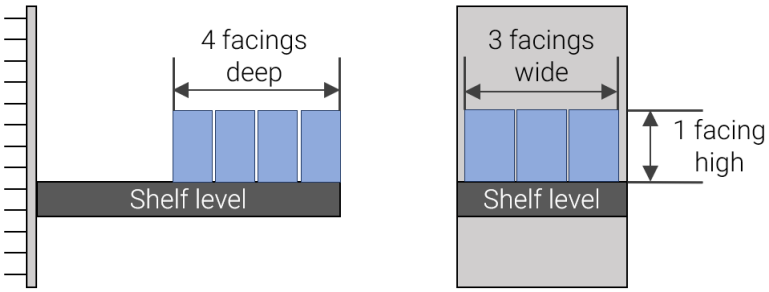


Figure 2.4: Number of facings wide, high, and deep on a planogram

2.2.2 Decisions

In the SSAP, the most important decisions correspond to the number of facings (wide) to attribute to each product and its vertical and horizontal location on the shelf (Bianchi-Aguiar, 2015). The horizontal location can be modeled as a continuous variable, whereas the vertical location is generally limited to a fixed (discrete) number of shelf levels. However, if there is the flexibility to adjust the fixtures' vertical location, an additional decision regarding the vertical location of each fixture can be made (see, for instance, Tongsari (1995)). The number of facings deep is generally not treated as a relevant decision in the SSAP, since it has no impact on the inventory that can be directly seen by the customer.

2.2.3 Objective

A profit maximization objective is often chosen for the SSAP, since the problem is normally seen from the retailer's perspective, that tries to make the most out of the available shelf space. The profit function usually takes into account the elasticity considerations described in Section 2.2.1. Notwithstanding that a cost minimization approach may also be followed, Bai (2005) points out that it may lead to a decline in profit due to conflicting goals.

2.2.4 Constraints

The SSAP is typically subject to a variety of constraints, which can vary according to the context of the problem and the specificities of the retailer. Janeiro (2014) and Bianchi-Aguiar (2015) divided the most common constraints into four different groups, as presented in Table 2.2.

Table 2.2: Typical constraints in shelf space allocation problems

Integrality Constraints	The number of facings to attribute to each product must be an integer value
Physical Constraints	The capacity of each shelf (in terms of length and, if considered, height) must not be exceeded
Control Constraints	It is common for retailers to define lower and upper bounds to the number of facings, so as to ensure limits for the level of exposure that a product may have
Family Constraints	Merchandising rules specify which products should be placed together on the shelves, according to their corresponding product families

Chapter 3

Literature Review

This chapter provides a literature review on the subject of shelf space allocation. Section 3.1 presents empirical and qualitative approaches to shelf space management, while Section 3.2 focuses solely on the application of optimization methods to solve the SSAP.

3.1 Empirical and qualitative approaches

Effectively managing shelf space is crucial for every retailer. In a qualitative study, Buttle (1984) presents a holistic perspective on space allocation in retail stores. The author highlights five different aspects that managers should pay attention to: (1) fixture location in the store; (2) product category location; (3) product allocation within each category; (4) off-shelf displays; and (5) point-of-sale promotional support. Regarding product allocation, a number of pertinent managerial insights are provided. The sequencing of items on the shelves should attend to the direction of traffic flow, by placing products in such a way that the most profitable items are the first to be found by customers. Furthermore, the notion of product families is implicitly addressed, by stating that retailers may choose to group different product brands vertically or horizontally – consumers generally prefer the former, as it facilitates price comparisons between competing brands. The use of commercial software applications is also addressed in this paper, with the author mentioning the rise in popularity of this type of applications among retailers, which allocate shelf space on the basis of simple rules.

As mentioned in Section 2.1, there is a wide range of studies that attempt to empirically measure the relationship between shelf space and sales. Eisend (2014) presents a meta-analysis that considers over 1,200 estimates of shelf space elasticities, which allowed for some interesting conclusions to be drawn. These conclusions were then translated into a list of practical recommendations for retailers. The author reports an average shelf space elasticity of 0.17, but the values vary considerably across product categories. As one would expect, commodities have a lower than average shelf space elasticity and, conversely, impulse buys have higher elasticity coefficients. Another valuable insight, particularly for large supermarket chains, is the fact that elasticity estimates may be sensitive to the size of the store. Additionally, the sales increments that are due to shelf

space increases appear to be less significant for large shelf space variations, which validates the idea of diminishing returns (that are translated into an S-shaped demand curve, as in Figure 2.2).

In an important study, Drèze et al. (1994) measure the impact of the products' location and number of facings on sales. They find that, after a certain threshold is reached, benefits from an increased number of facings are practically non-existent. Interestingly, they also conclude that the impact of location on sales is far stronger than the impact of the number of facings, with the vertical location having a more noticeable effect on sales than the horizontal location. In addition, although it is clear that the best vertical location lies approximately at the eye level (except for the refrigerated section), the best horizontal location varies substantially across product categories.

Experimental studies on cross-space elasticities are scarce in the literature. In fact, these values are very difficult to estimate and are often disregarded in shelf space allocations (Bianchi-Aguiar, 2015). Schaal and Hübner (2017) combine an empirical approach with the application of a stochastic shelf space optimization model to investigate whether it is economically justified to develop and implement shelf space allocation models with cross-space elasticity considerations. They conclude that, under normal circumstances, the effect of cross-space elasticity is practically meaningless and adds little value to the shelf space optimization models, since its impact on the retailer's profit is very limited. As empirically measuring this effect may be particularly costly, it is probably more advantageous for retailers to direct their efforts towards other, potentially more meaningful, investigations.

3.2 Optimization methods

Bianchi-Aguiar et al. (2018a) review the major streams of research that have emerged within shelf space optimization since the problem first became relevant for the Operations Research community. Their work is the first to systematize published research in the area, also including a classification framework and a proposal of a unique modeling approach for the Shelf Space Allocation Problem. Their study focuses exclusively on papers that make use of quantitative models, which is also the focus of the present section. Subsection 3.2.1 covers papers that focus exclusively on shelf space-related decisions, while Subsection 3.2.2 provides a brief overview of works that integrate other planning areas in these decisions, namely, assortment selection, in-store replenishment, and pricing.

3.2.1 Shelf Space Allocation problems

Corstjens and Doyle (1981) were among the first authors to propose a mathematical model to optimize shelf space allocation. Their paper can certainly be regarded as influential, as it was a major reference for most of the research works that followed. The demand for product i, q_i, is modeled in terms of a multiplicative function, which includes both space and cross-space elasticities, as presented in Equation (3.1).

$$q_i = \alpha_i \, s_i^{\beta_i} \prod_{\substack{j=1 \\ j \neq i}}^{K} s_j^{\delta_{ij}} \tag{3.1}$$

In this demand function, the factor $s_i^{\beta_i}$ estimates the impact of shelf space on the demand of product i, with s_i corresponding to a unit of shelf space and β_i being the space elasticity of product i, which assumes values between 0 and 1, in order to account for diminishing returns. It is important to note that there is no relationship between a unit of shelf space and the number of facings of the corresponding product, as integrality constraints are not imposed, which is a clear limitation of the model. Moreover, the factor $\prod_{\substack{j=1 \\ j \neq i}}^{K} s_j^{\delta_{ij}}$ measures how the demand of product i is affected by the space that is attributed to the other products (indexed by j). Here, K denotes the number of items in the assortment, while δ_{ij} refers to the cross-space elasticity between products i and j, which may be positive or negative, depending on whether the products are complementary or substitute, respectively. Finally, α_i is used as a scale parameter.

The only decision variables that are present in the mathematical model correspond to the space that is attributed to each product (s_i), with the objective being the maximization of the total profit. Hence, for each product i, demand (q_i) is first multiplied by the corresponding margin, w_i. Then, the associated costs are deducted using a cost function that depends on the shelf space that is attributed to the product, $C_i(s_i)$. Concerning the model's constraints, it is possible to group them into four categories, as suggested by Janeiro (2014), which are presented in Table 3.1.

Table 3.1: Constraints of the Shelf Space Allocation model of Corstjens and Doyle (1981)

Capacity	The total space that is attributed to the products cannot exceed the store's capacity in terms of available shelf space
Availability	Demand fulfillment is subject to a production (or availability) limit
Control	Lower and upper bounds for the space that is allocated to each product must be respected
Non-negativity	The space allocated to each product must be a non-negative value

The authors tested their model using the data that were provided by a retailer, with the goal of optimizing space allocation of 5 product categories. Their approach led to profit increases of around 19.8% for small stores and 3.7% for large stores. The fact that they were able to show that the use of analytical methods could substantially outperform allocations based on rules of thumb set the stage for further research in this field.

One of the most evident drawbacks of Corstjens and Doyle's approach is the fact that it disregards the products' location on the shelves, as the only decision that is made is the space that ought to be allocated to each product. Yang and Chen (1999) took into consideration the conclusions presented by Drèze et al. (1994), which pointed towards the location decision having a markedly higher impact on demand than the allocated shelf space, and were the first authors to introduce location decisions into shelf space allocation models. The decision variables, x_{ik}, denote the number of facings of product i that should be allocated to shelf k. These variables are required to be integer, leading to a much more precise space allocation. Similarly to Corstjens and Doyle

(1981), they propose a multiplicative function for the purpose of demand modeling, with the possibility of adding new marketing variables that may be relevant for the problem, and incorporate the constraints that were mentioned in Table 3.1. To avoid having to solve a nonlinear programming problem, the authors propose an alternative model, with a linear objective function, as displayed in Equation (3.2). By assuming that the retailer has an effective logistic system, they also neglect the availability constraints that had been previously considered.

$$\text{Maximize } P = \sum_{i=1}^{n} \sum_{k=1}^{m} p_{ik} x_{ik} \tag{3.2}$$

Regarding the notation, n and m denote the number of products in the assortment and the number of shelves, respectively, while p_{ik} corresponds to the profit per facing of product i on shelf k. By using this model, the authors are implicitly assuming that the profit associated with each product varies linearly with the number of facings, provided that the number of facings is contained in the interval defined by the control constraints. It is important to note that only the vertical location of items is considered in the objective function, with the horizontal location being disregarded.

Acknowledging the strong similarities between multi-constraint knapsack problems and the SSAP, Yang (2001) developed a heuristic algorithm that adapts a commonly used procedure in knapsack problems to solve the SSAP. First, a constructive procedure is implemented. It consists of three main stages: (1) preparatory phase, that checks whether the problem is feasible; (2) allocation phase, that allocates the available space to the products according to a priority ranking; and (3) termination phase, where the solution is evaluated according to the objective function presented in Equation (3.2). Despite this constructive procedure being sufficient to obtain satisfactory results, three options are considered for an improvement phase. Not only did the improved heuristic provide excellent results, it also proved out to be extremely efficient. In order to have a benchmark for evaluating the obtained results, the model of Yang and Chen (1999) (without availability constraints, which was the basis for the development of this heuristic approach) was solved to optimality, using complete enumeration. One of the improvement options yielded an average ratio of the heuristic solution to the optimal solution of 99.6%.

Despite its promising results, Lim et al. (2004) point out some limitations of this approach. To begin with, the data set that was used to test the heuristic may not have been the most appropriate, as there was enough shelf space to allocate all products even when the number of facings that was introduced for each item coincided with the upper bound. Hence, the upper bound constraint did not have any real impact on the solution that was obtained. Additionally, some problems are identified in the heuristic itself. Firstly, in the constructive procedure, each of the considered products is first allocated so as to satisfy its minimum display requirement, and only then is the remaining shelf space filled, which may lead to suboptimal allocations. Secondly, in the improvement stage, the designed neighborhood structures severely restrict the search space, thus not ensuring a proper search over the entire solution space. Lim et al. (2004) address all these issues in their paper. First, they implement three new neighborhood structures, in an attempt to diversify the search after an initial solution is generated. Then, they present a method that adapts the SSAP into a network flow

model. Finally, two metaheuristics, Tabu Search (TS) and Squeaky-Wheel Optimization (SWO), are also implemented. In order to test the different algorithms and compare them with the heuristics presented by Yang (2001), they consider small and large instances, placing more emphasis on the large instances, as they could provide a more accurate depiction of a real-world scenario. They conclude that the new implementations outperformed Yang's heuristic approaches, with the network flow model and the SWO algorithm yielding the best results for large instances.

In the works that have been presented so far, the choice of the products' horizontal location has not been included in the mathematical formulation. In fact, this decision has been neglected in the vast majority of mathematical models than can be found in the literature (Bianchi-Aguiar et al., 2018a). In an innovative paper, van Nierop et al. (2008) propose a two-fold approach to shelf space optimization, considering the horizontal location of the items as a relevant decision. They start by providing a statistical model that attempts to measure the impact of shelf layout on sales, recurring to Bayesian hierarchical modeling. Then, by applying Simulated Annealing (SA), they optimize shelf space allocation, using the sales model that was built in the previous stage. Hwang et al. (2009) also tackle the decision of determining the horizontal location of the products on the corresponding shelves. They present a novel approach, viewing the problem in a geometric fashion, that tries to determine how to optimally partition a rectangular shelf space into smaller rectangles (that correspond to the space that should be allocated to each product) using genetic algorithms. However, their test set is very limited, consisting of only 4 products (which are referred to as "brands of items" in the article). As a result, the conclusions that can be drawn from this work are fairly vague. A geometric perspective is also the one that inspired the two-dimensional study of shelf space allocation of Geismar et al. (2015). Their formulation attempts to maximize the total weighed revenue of a given two-dimensional display, considering that each vertical location has a specific degree of effectiveness at generating revenue, and may be applied to areas other than shelf space allocation, such as retail flyer design and web page advertising layouts. The problem is decomposed into two smaller sub-problems, each of which is solved using a method that combines integer programming with an algorithm for the maximum-weight independent set problem.

The fact that retailers, due to aesthetic considerations, try to "allocate shelf space to product items or brands as uniform and complete columns" (Zufryden, 1986) had remained practically absent from the literature until the work of Russell and Urban (2010) was published. Despite Lim et al. (2004) having proposed an extension to the profit function that favors the placement of certain types of items on adjacent locations, the notion of product families (as defined in Subsection 2.2.1) was only introduced by Russell and Urban (2010). In their case, product families may be based on a variety of characteristics, even though they are normally determined by the product brand. In their formulation of the SSAP, the goal is to find the optimal number of facings and location (vertical and horizontal), while maintaining the integrity of product families (i.e. products belonging to same family are placed together). The objective function that is used is the total profit, as in Equation (3.2). The expression used to calculate the expected demand for each product is based on the research of Drèze et al. (1994), who indicated that demand tended to vary according

to a quadratic function in the horizontal dimension and to a cubic function in the vertical direction. As for the impact of the number of facings on the demand, a quadratic formulation is chosen both for the sake of consistency and to reflect diminishing returns. Two different mathematical models are developed. The first one, best suited towards smaller-sized instances, treats each product's location as a continuous variable and therefore enables the determination of the precise location of the items on the shelves, while ensuring the integrity of product families. The second one, developed to handle larger problems, despite still making sure that each family is kept on adjacent shelves, does not allow for the determination of the exact location of each item. Nevertheless, a post-processing procedure was developed in order to convert the generated solution into one that indicates the precise shelf location of the items. This second model has the additional drawback of not guaranteeing the vertical alignment of product families (at least not to the extent to which this constraint is enforced in first model).

Bianchi-Aguiar et al. (2018b) build on and extend the notion of product families to present their formulation of the SSAP. By organizing the different products into families, retailers can then establish an appropriate set of merchandising rules (as defined in Subsection 2.2.1), which aim to reproduce the way customers look for products while shopping. These rules determine how the different product families should be arranged on the shelves. In order to attract customers' attention and interest, retailers normally want product families to be organized in rectangular shapes. However, as aforementioned above, most of the shelf space-related literature completely disregards this crucial aspect of shelf space management. It is this misalignment between theory and practice that sets the motivation for their novel approach to the SSAP. They propose a mixed integer programming formulation for the SSAP, using single commodity flow constraints to model product sequencing. The decision variables include the number of facings to attribute to each product and their horizontal and vertical location on the shelves. To solve the problem, the authors developed a mathematical programming-based heuristic (matheuristic) which makes use of the existing hierarchy of product families to decompose the problem into smaller sub-problems. Procedures for improving the matheuristic's efficiency and for recovering from infeasible solutions are also implemented. A series of computational experiments were then conducted to assess the performance of the exact mathematical formulation and the matheuristic, and promising results were obtained. This work served as the starting point for the development of the heuristic that is presented in Chapter 5. Taking into consideration the specificities of the problem that were presented in the described paper, that provide a more accurate depiction of real-world retail enviornments, a new heuristic method for the SSAP will be proposed. Given the central role of this paper in the motivation of this dissertation, the details of its formulation will be more thoroughly presented in Chapter 4, where a formal mathematical description of the problem is introduced.

3.2.2 Related planning problems

In the shelf space planning literature, it is common to find integrative approaches that combine shelf space allocation with other planning problems. Borin et al. (1994) present one of such approaches, proposing a joint optimization of shelf space allocation and assortment planning. To

solve the problem, they apply a Simulated Annealing algorithm and compare its performance with a commonly used rule of thumb, which allocated space to an SKU in direct proportion to its historical sales share. They conclude that SA leads to a substantially better performance, even when the proportional shelf allocation rule is applied to the optimal assortment. In a recent paper, Hübner and Schaal (2017b) propose an integrated assortment and shelf space optimization model that considers stochastic and space-elastic demand, out-of-assortment and out-of-stock substitution effects. By applying a specialized heuristic on two real data sets containing perishable and non-perishable items, they report profit increases of up to 25%.

Space management decisions also have a clear impact on inventory control and in-store replenishment processes. Hübner and Schaal (2017c) explain that "keeping more shelf stock of an item increases the demand for it due to higher visibility, permits decreased replenishment frequencies and increases inventory holding costs". Nonetheless, as shelf space is limited, it will necessarily trigger the opposite effect for other products in the assortment that will be assigned a smaller space. In a case study in collaboration with a German grocery retailer, they present an optimization model for profit maximization that determines the number of facings that each product should have on the shelves, the corresponding display orientation, and the order frequencies, while considering space-elasticity effects and limited shelf and backroom capacity at the stores. They report a significant profit potential of about 29%.

The work of Bianchi-Aguiar et al. (2015) also describes a shelf space management approach that is inventory-oriented. Here, they introduce the notion of planogram replication, which consists in translating a master planogram into a store-specific planogram (the corresponding definitions can be found in Subsection 2.2.1), accounting for the particular characteristics of each store. As they assume that the maximization of space effectiveness was already considered when generating the master planogram, the creation of store-specific planograms is based on a mathematical model that has the goal of balancing the products' inventory levels (more precisely, the days-supply indicator) in order to favor joint in-store replenishments. This problem is defined as the Shelf Space Replication Problem (SSRP).

Regarding the incorporation of other decisions when allocating products to shelves, the article presented by Murray et al. (2010) is particularly noteworthy, proposing an innovative approach that optimizes not only the location and space assigned to each product, but also the display orientation and the price that is charged to the customer. The problem is formulated as a mixed integer nonlinear problem (MINLP) and solved using a branch-and-bound algorithm.

Chapter 4

Problem formulation

In the present chapter, the formulation of the SSAP is provided. In Section 4.1, the problem is described from a formal point of view. Given that the formulation of the problem is based on the work of Bianchi-Aguiar et al. (2018b), their mathematical optimization model, which proposes a novel network flow approach, is broadly presented in Section 4.2. For the purpose of consistency, the same notation is used.

4.1 Problem description

In the Shelf Space Allocation Problem, the goal is to find the optimal allocation of a set of N products (indexed by i, $j \in \mathcal{N}$) on a set of K shelves (indexed by $k \in \mathcal{K}$). As indicated in Figure 4.1, each product i, of width a_i and height b_i, must be placed on a particular shelf k, which has a total width of w_k and a height of h_k. The total number of facings of product i must be contained in the interval $[l_i, u_i]$, which denote the lower and upper bounds, respectively.

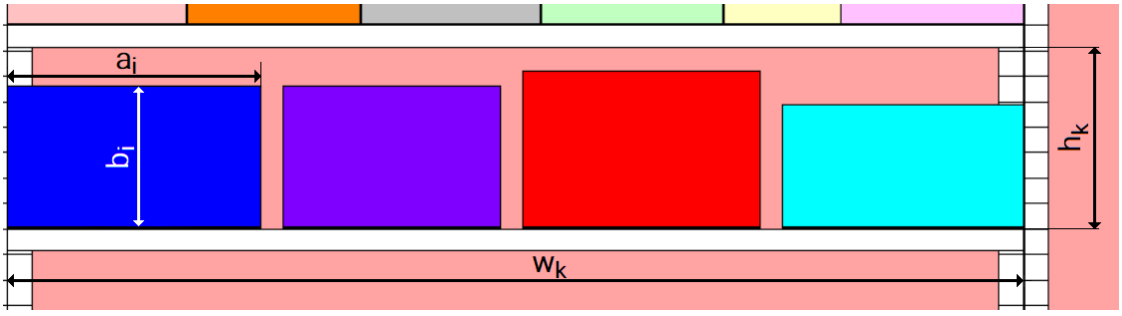


Figure 4.1: Product and shelf widths and heights

Product grouping considerations are also integrated in the formulation of the problem under analysis. As introduced in Section 2.1, retailers typically specify a set of merchandising rules that determine the arrangement of product families on the shelves. We consider a set of M product families (indexed by $u \in \mathcal{M}$), each with its corresponding set of products, $\mathcal{N}_u$. For modeling purposes, merchandising rules can be seen as a set of hierarchical relationships between different product families. While shopping, consumers make their purchase decisions according to the

products' attributes. For a given product category, some attributes have a stronger influence on the customers' decisions than others. Retailers try to understand the relative impact of each attribute in order to know how to group products into families and organize them on the shelves accordingly. In this way, retailers attempt to capture the behavior of their customers and enhance their in-store experience and overall satisfaction, as less time is spent searching for specific products on the shelves.

The planograms that are shown in Figure 4.2 are representative of the multi-level hierarchical structure of product families that stems from the retailer's set of merchandising rules. Firstly, as seen in the planogram on the left, products are divided into two families, A and B, according to their type. Then, within each of these two families, the products are further differentiated with respect to their brand, as depicted in the planogram on the right. This leads to families A1, A2, and A3 (which *belong to* family A) and families B1, B2, and B3 (which *belong to* family B). As illustrated by both planograms, products are placed on the shelves in such a way that their corresponding families form rectangular and uniform shapes. For this reason, product families will also be referred to as *blocks*, and both terms will be used interchangeably in the remainder of this dissertation.

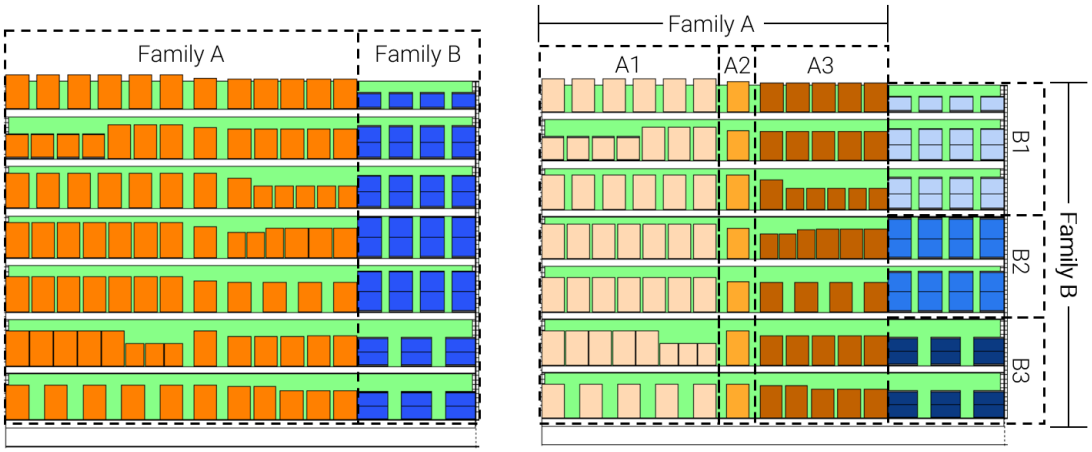


Figure 4.2: Example of a planogram highlighting the arrangement of product families on the shelves

Bianchi-Aguiar et al. (2018b) show that this hierarchy of product families can be translated into a directed acyclic graph (DAG). Figure 4.3 contains the DAG that represents the hierarchical structure of product families from the previous example. A dummy node (equivalent to a product family containing all the products in the planogram) connects with the product families from the first level, A and B. Moving downstream, each of the product families from the first level connects with the corresponding second level families. Even when considering only these two levels, the graph could still be extended such that each of the last level blocks would be connected with its corresponding products, with each product being represented by a node. Such an extension would allow for the full representation of the problem, but was omitted for the sake of simplicity. Hereafter, when considering any given block, we will refer to the immediately upstream block to which

it is connected as its *parent* block, while the immediately downstream blocks to which it connects will be referred to as its *child* blocks. This conceptualization of the problem is particularly useful for the mathematical formulation of the problem, which will be detailed in Section 4.2.

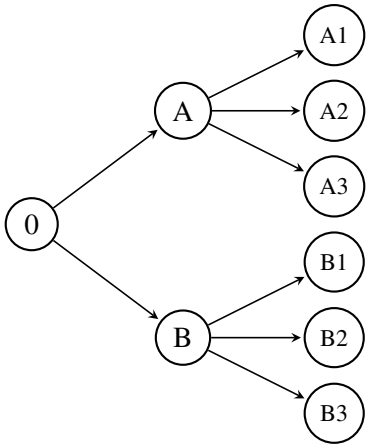


Figure 4.3: Two-level hierarchical structure of product families

Another important aspect that derives from the specification of merchandising rules and was emphasized in Figure 4.2 is the display orientation (direction) of product families. In the right planogram of the referred figure, it is clearly noticeable that blocks A1, A2, and A3 are arranged vertically, whereas blocks B1, B2, and B3 are displayed in a horizontal fashion. It is a requirement of the problem that blocks that share the same parent block must be displayed in the same direction. As aforementioned, all blocks must form rectangular shapes (hence, no L shapes are allowed). As a result, vertically implemented blocks must occupy all shelf levels of the corresponding parent block. On the other hand, notwithstanding that horizontally oriented blocks tend to occupy the full width of the parent block, this need not always be the case. In fact, they may share the same shelf level with other horizontal blocks, insofar as each of these other blocks are also restricted to the referred shelf level, so that the rectangular shapes of all blocks are preserved.

Provided this description of the problem, there are three types of decisions to be made: (1) the number of facings to allocate to each product; (2) the shelf level on which each facing should be placed; and (3) the horizontal location of each facing within the shelf.

4.2 Mathematical formulation

With regard to the objective function, and similarly to Yang and Chen (1999), a linear profit function is used. A small nuance exists in the present formulation, nonetheless, which consists in expressing p_{ik} (the profit per facing of product i on shelf k) as the product of two factors: the profit per facing of product i (p_i) and the attractiveness of shelf k (γ_k), with the latter being an estimate of how effective shelf k is at stimulating consumer demand. Generally speaking, eye level shelves have the highest attractiveness coefficients. Equation (4.1) presents the objective function of the mathematical model, with W_{ik} representing the number of facings of product i on shelf k. Note

that the objective function does not incorporate any horizontal elasticity effects.

$$\text{Maximize } Z = \sum_{i \in \mathcal{N}} \sum_{k \in \mathcal{K}} p_i \cdot \gamma_k \cdot W_{ik} \tag{4.1}$$

The profit function is then subject to a series of constraints that will now be briefly presented. Constraints (4.2) impose a minimum and maximum number of facings for each product, whilst the set of constraints in (4.3) prevents the shelf capacity from being exceeded.

$$l_i \leq \sum_{k \in \mathcal{K}} W_{ik} \leq u_i, \quad \forall\, i \in \mathcal{N} \tag{4.2}$$

$$\sum_{i \in \mathcal{N}} a_i \cdot W_{ik} \leq w_k, \quad \forall\, k \in \mathcal{K} \tag{4.3}$$

To determine the sequencing of the products on the shelves, Bianchi-Aguiar et al. (2018b) explored the hierarchical nature of the problem, rather than directly defining the sequence of products on the shelves. For each shelf, blocks that share the same parent block are sequenced, in a process that is performed one hierarchical level at a time. By continuing to move downstream until the product level is reached, all products can then be placed on the shelf according to a specific order, which provides an overall product sequence (which was the initial goal), while ensuring that products that belong to the same block are placed in consecutive locations of the shelf.

For modeling purposes, a network is considered for each parent block, with the nodes representing the corresponding blocks from the downstream level. For this purpose, individual products can be seen as blocks containing only one item. Before presenting the next sets of constraints, some additional notation must first be introduced. The set $\mathcal{V}_u$ contains the child blocks of block $u \in \mathcal{M}$ and the set $\mathcal{V} = \mathcal{M} \cup \mathcal{N}$ aggregates all of the blocks (including the blocks which contain only one product). Finally, in each of the referred networks, a dummy node (0) associated with the source (initial) and sink (final) nodes is considered. As a consequence of this network formulation, two new sets of binary decision variables are introduced: T_{mnk}, which takes the value of 1 if block m is displayed immediately after block n on shelf k, and Y_{mk}, which assumes the value of 1 when block m is present on shelf k.

Constraints (4.4) to (4.7) guarantee that when a block is present on a shelf, a sequence is defined for the corresponding child blocks, starting and ending at the source and sink nodes, respectively. In addition, constraints (4.8) ensure that a block is only present on a particular shelf if its parent block is also present on the same shelf, while constraints (4.9) provide the necessary connection with the allocation part of the problem, stating that a given product may only be placed on a shelf on which the blocks that contain that specific product are present.

$$\sum_{m \in \mathcal{V}_u} T_{0mk} = Y_{uk}, \quad \forall\, u \in \mathcal{M}, k \in \mathcal{K} \tag{4.4}$$

$$\sum_{m \in \mathcal{V}_u} T_{m0k} = Y_{uk}, \quad \forall\, u \in \mathcal{M}, k \in \mathcal{K} \tag{4.5}$$

$$\sum_{n \in \mathcal{V}_u \cup \{0\}} T_{nmk} = Y_{mk}, \quad \forall\, u \in \mathcal{M}, m \in \mathcal{V}_u, k \in \mathcal{K} \tag{4.6}$$

$$\sum_{n \in \mathcal{V}_u \cup \{0\}} T_{mnk} = Y_{mk}, \quad \forall\, u \in \mathcal{M}, m \in \mathcal{V}_u, k \in \mathcal{K} \tag{4.7}$$

$$Y_{mk} \leq Y_{uk}, \quad \forall\, u \in \mathcal{M}, m \in \mathcal{V}_u, k \in \mathcal{K} \tag{4.8}$$

$$W_{ik} \geq Y_{ik}, \quad \forall\, i \in \mathcal{N}, k \in \mathcal{K} \tag{4.9}$$

In order to guarantee that the sequencing decisions are fully completed, single commodity flow constraints are defined, with the commodity being associated with the length of the corresponding upstream block. This leads to the addition of two new sets of decision variables to the model: F_{mnk}, which corresponds to the continuous flow (i.e. the cumulative shelf length) from block m to block n on shelf k, and L_{ik}, which represents the shelf length assigned to product i on shelf k.

Constraints (4.10) impose a null initial flow, constraints (4.11) state that the total flow must be equal to the length of the upstream block on the shelf, and constraints (4.12) ensure flow balance. The flow is required to be null between blocks that are not connected, as indicated in (4.13). Finally, the sum of all shelf lengths L_{ik} assigned to products on shelf k must be equal to the total shelf width w_k (4.14), and the length assigned to each product i on a particular shelf k, L_{ik}, must be sufficient to accommodate all facings of that product on that specific shelf (4.15).

$$\sum_{m \in \mathcal{V}_u} F_{0mk} = 0, \quad \forall\, u \in \mathcal{M}, k \in \mathcal{K} \tag{4.10}$$

$$\sum_{m \in \mathcal{V}_u} F_{m0k} = \sum_{i \in \mathcal{N}_u} L_{ik}, \quad \forall\, u \in \mathcal{M}, k \in \mathcal{K} \tag{4.11}$$

$$\sum_{\substack{n \in \mathcal{V}_u \cup \{0\}: \\ m \neq n}} F_{nmk} + \sum_{i \in \mathcal{N}_m} L_{ik} = \sum_{\substack{n \in \mathcal{V}_u \cup \{0\}: \\ m \neq n}} F_{mnk}, \quad \forall\, u \in \mathcal{M}, m \in \mathcal{V}_u, k \in \mathcal{K} \tag{4.12}$$

$$F_{mnk} \leq w_k \cdot T_{mnk}, \quad \forall\, u \in \mathcal{M}, m, n \in \mathcal{V}_u \cup \{0\} : m \neq n, k \in \mathcal{K} \tag{4.13}$$

$$\sum_{i \in \mathcal{N}} L_{ik} = w_k, \quad \forall\, k \in \mathcal{K} \tag{4.14}$$

$$a_i \cdot W_{ik} \leq L_{ik}, \quad \forall\, i \in \mathcal{N}, k \in \mathcal{K} \tag{4.15}$$

With the constraints that have been introduced so far, there is no guarantee that the different blocks will form rectangular shapes when the corresponding products are placed on the shelf, as their precise horizontal location is yet to be determined. For that purpose, four new sets of decision variables are introduced: X^s_m and X^e_m indicate the left and right coordinates of block m, and FL_{mk} and LL_{mk} are binary variables that take the value of 1 if shelf k is the first or last shelf of block m, respectively.

Constraints (4.16) and (4.17) define the left coordinate of each block, whereas constraints (4.18) and (4.19) establish the corresponding right coordinate, allowing for a small horizontal deviation v between the different shelves on which a given block is present. Constraints (4.20) and (4.21) define the first and last shelves of each block, and constraints (4.22) to (4.24) ensure that any given block is kept on adjacent shelves.

$$X_m^s \geq X_u^s + \sum_{\substack{n \in \mathcal{V}_u \cup \{0\}: \\ n \neq m}} F_{nmk}, \quad \forall u \in \mathcal{M}, m \in \mathcal{V}_u, k \in \mathcal{K} \tag{4.16}$$

$$X_m^s \leq X_u^s + \sum_{\substack{n \in \mathcal{V}_u \cup \{0\}: \\ n \neq m}} F_{nmk} + W \cdot (1 - Y_{mk}), \quad \forall u \in \mathcal{M}, m \in \mathcal{V}_u, k \in \mathcal{K} \tag{4.17}$$

$$X_m^e - X_m^s \geq \sum_{i \in \mathcal{N}_m} L_{ik}, \quad \forall m \in \mathcal{V}, k \in \mathcal{K} \tag{4.18}$$

$$X_m^e - X_m^s - v \leq \sum_{i \in \mathcal{N}_m} L_{ik} + W \cdot (1 - Y_{mk}), \quad \forall m \in \mathcal{V}, k \in \mathcal{K} \tag{4.19}$$

$$\sum_{k \in \mathcal{K}} FL_{mk} = 1, \quad \forall m \in \mathcal{V} \tag{4.20}$$

$$\sum_{k \in \mathcal{K}} LL_{mk} = 1, \quad \forall m \in \mathcal{V} \tag{4.21}$$

$$FL_{m,k+1} + Y_{mk} = Y_{m,k+1} + LL_{mk}, \quad \forall m \in \mathcal{V}, k \in \mathcal{K} : k \neq K \tag{4.22}$$

$$FL_{m0} = Y_{m0}, \quad \forall m \in \mathcal{V} \tag{4.23}$$

$$LL_{mK} = Y_{mK}, \quad \forall m \in \mathcal{V} \tag{4.24}$$

Lastly, it is necessary to impose the proper display orientation (horizontal or vertical) to each block. Let $\mathcal{S}^V$ and $\mathcal{S}^H$ denote the sets of product families that should have their child blocks arranged vertically and horizontally, respectively. While vertical blocks must be present on the same shelves of their parent block (4.25), horizontal blocks can only occupy more than one shelf level if the first one is completely occupied by the block (4.26).

$$Y_{mk} = Y_{uk}, \quad \forall u \in \mathcal{S}^V, m \in \mathcal{M}_u, k \tag{4.25}$$

$$\sum_{n \in \mathcal{V}_u : n \neq m} Y_{nk} \leq M \cdot (2 - Y_{mk} - Y_{m,k+1}), \quad \forall u \in \mathcal{S}^H, m \in \mathcal{M}_u, k \in \mathcal{K} : k \neq K \tag{4.26}$$

The full notation of the mathematical model can be consulted in Appendix A. For a complete description of the model and a proposal of a mathematical programming-based solution approach to the SSAP, please refer to Bianchi-Aguiar et al. (2018b).

Chapter 5

A heuristic for optimizing Shelf Space Allocation

The heuristic approach to the SSAP that was followed in the present work is detailed in this chapter. Section 5.1 describes the constructive procedure that allows us to obtain an initial solution. Section 5.2 presents the implementation of the Iterated Local Search metaheuristic in the context of the problem under analysis. The additional notation that will be introduced in this chapter can be consulted in Appendix B.

5.1 Constructive procedure

In this section, the constructive heuristic is presented. The main objective of this procedure is to generate an initial solution to the Shelf Space Allocation Problem. Before proceeding, we need to introduce the concept of *target facings* of a given product $i \in \mathcal{N}$, denoted by TF_i, which corresponds to the number of facings (wide) that would be attributed to the product if no merchandising rules were defined. This notion of target facings was also introduced by Bianchi-Aguiar et al. (2016), but in a different context, in which the goal was to balance in-store replenishment frequencies among the different products.

As depicted in Figure 5.1, the heuristic is comprised of three main steps that are performed sequentially: (1) target facings calculation; (2) block allocation; and (3) product allocation. In the referred figure, arrows pointing to a given stage represent an input of data. Each of the heuristic's stages will be explained separately in Subsections 5.1.1, 5.1.2, and 5.1.3.

5.1.1 Target facings calculation

In this stage, the goal is to obtain the number of target facings of each product $i \in \mathcal{N}$, which will allow for the estimation of the amount of space that should be allocated to each block (further details will be provided in Subsection 5.1.2). If we completely disregard the retailer's set of merchandising rules, the problem's complexity decreases substantially, as constraints (4.4) to (4.26) would no longer be necessary. Hence, one can interpret a model for calculating the target facings

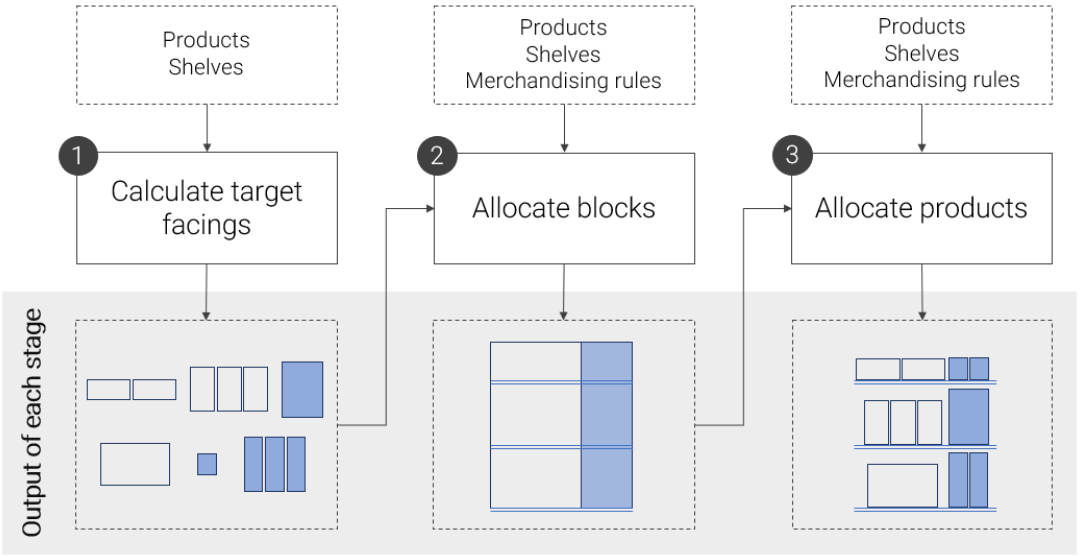


Figure 5.1: Main phases of the constructive heuristic

as a *best case scenario* – ideally, the number of facings to attribute to each product would be equal to the corresponding number of target facings, in order to maximize the total profit.

The target facings model is as follows. Equation (5.1) presents the considered objective function, Z^{TF}, which is very similar to the objective function that was already presented in (4.1) for the SSAP model, where we attempt to maximize a total profit function. In this case, however, since no actual allocation occurs, space and location elasticity considerations can be disregarded. For each $i \in \mathcal{N}$, TF_i corresponds to a decision variable of the model. Since the goal of this stage is to obtain a broad estimate of the number of facings to attribute to each product, there are no location decisions to be made, as these will only be introduced in the next stages of the constructive heuristic. Constraints (5.2) impose lower and upper bounds to the number of target facings, and constraints (5.3) ensure that the total capacity (considering all shelves) is not exceeded. Finally, constraints (5.4) restrict the domain of the decision variables to the set of natural numbers including zero.

$$\text{Maximize} \quad Z^{TF} = \sum_{i \in \mathcal{N}} p_i \cdot TF_i \tag{5.1}$$

$$\text{Subject to:} \quad l_i \leq TF_i \leq u_i, \quad \forall i \in \mathcal{N} \tag{5.2}$$

$$\sum_{i \in \mathcal{N}} a_i \cdot TF_i \leq \sum_{k \in \mathcal{K}} w_k \tag{5.3}$$

$$TF_i \in \mathbb{N}_0, \quad \forall i \in \mathcal{N} \tag{5.4}$$

This formulation is tantamount to a Bounded Knapsack Problem (BKP), with the slight caveat that each product $i \in \mathcal{N}$ must be selected at least l_i times. Hence, transforming our problem into a BKP is straightforward: to each product $i \in \mathcal{N}$, we attribute a number of facings equal to the corresponding lower bound l_i, and then reduce the total knapsack capacity by $\sum_{i \in \mathcal{N}} a_i \cdot l_i$. Considering a similar set of decision variables TF'_i $(= TF_i - l_i, \forall i \in \mathcal{N})$ and replacing constraints

(5.2) and (5.3) with (5.5) and (5.6), we obtain the exact formulation of a BKP.

$$0 \leq TF_i' \leq u_i - l_i, \quad \forall i \in \mathcal{N} \tag{5.5}$$

$$\sum_{i \in \mathcal{N}} a_i \cdot TF_i' \leq \sum_{k \in \mathcal{K}} w_k - \sum_{i \in \mathcal{N}} a_i \cdot l_i \tag{5.6}$$

Additionally, we can convert a BKP into an equivalent 0/1 Knapsack Problem (KP) by creating $u_i - l_i$ copies of each product i and consider each of them as a distinct item to be inserted in the knapsack (Kellerer et al., 2004). To solve the KP, a greedy algorithm will be used. It is clear that such an approach does not necessarily lead to the optimal solution – in fact, in certain instances, it may even lead to poor solutions. Nevertheless, a greedy heuristic suffices our purposes in the present stage for two main reasons. Firstly, as stated previously, we are only interested in obtaining a broad estimate of the number of facings to allocate to each product, and a greedy algorithm provides an efficient way of quickly obtaining those estimates. Secondly, given that the average width of a product is significantly smaller than the total shelf space available, it is more likely that a greedy algorithm will lead to a reasonable solution.

Algorithm 1 provides the pseudocode of the greedy heuristic. We start by attributing to each product its corresponding minimum number of facings (line 3) and update the available shelf space accordingly (line 4). The greedy component of the heuristic relies on the profit-to-width ratio of each product, which we designate as *score* (line 5). Then, we add facings to the products until there are no products that fit in the remaining shelf space or the upper bound has been reached for all products (lines 8–24). When that is the case, the auxiliary variable $flag$ becomes false, which signals that the allocation is complete, as indicated in line 22. The product i^* to which a facing is added is selected based on its corresponding score. Since we are dealing with a pure greedy implementation, the product with the highest score is selected (lines 18 and 19) and we subtract its width from the available space (line 20). To make sure that the obtained solution is feasible, it is ensured that the products that can no longer be assigned any additional facings have a null score (line 12).

5.1.2 Block allocation

After the target facings of each product are computed, the different blocks are allocated. Allocating a block u implies defining its width, the shelf levels on which it is located, and the left coordinate (with the right coordinate being obtained by adding the width of the block to its left coordinate). Considering that the blocks' shapes are required to be rectangular, as discussed in Section 4.1, the width of a block is constant (i.e. does not depend on the particular shelf level we are considering). For any given block u, L_u will denote its width and $\mathcal{K}_u$ will represent the set of shelves on which it is present (with K_u corresponding to the cardinality of the set $\mathcal{K}_u$). Regarding the left and right coordinates of each block, the notation that was introduced in Section 4.2 will be kept: X_m^s (left coordinate) and X_m^e (right coordinate). In addition, we consider a total of C hierarchical block levels (indexed by c), which include an initial level containing a dummy block and a

```
1  emptySpace ← ∑_{k∈K} w_k
2  for i ← 1 to N do
3  |   TF_i ← l_i
4  |   emptySpace ← emptySpace − a_i · l_i
5  |   score_i ← p_i / a_i
6  end
7  allocationComplete ← false
8  while allocationComplete = false do
9  |   flag ← false
10 |   for i ← 1 to N do
11 |   |   if a_i > emptySpace or TF_i = u_i then
12 |   |   |   score_i ← 0
13 |   |   else
14 |   |   |   flag ← true
15 |   |   end
16 |   end
17 |   if flag = true then
18 |   |   i* ← arg max_{i∈N}(score_i)
19 |   |   TF_{i*} ← TF_{i*} + 1
20 |   |   emptySpace ← emptySpace − a_{i*}
21 |   else
22 |   |   allocationComplete ← true
23 |   end
24 end
```

Algorithm 1: Greedy heuristic for calculating the target facings

final level in which each block is associated with a single product. Finally, it is useful to introduce the set $\mathcal{G}_c$, which includes the blocks in hierarchical level c.

The allocation of blocks constitutes a challenging geometrical problem. First, the two dimensions of a planogram are of different nature: while the horizontal location can be treated as a continuous variable, the vertical location is confined to a discrete number of shelf levels. Additionally, product families have different predefined display orientations. Due to the fact that vertical and horizontal block allocations are fundamentally different, they require distinct approaches, which will be explained in detail in 5.1.2.1 and 5.1.2.2, respectively.

Algorithm 2 provides a high-level picture of the block allocation process. We start by allocating the block that aggregates all products (line 1), which is assigned the total width of the planogram and all of the existing shelf levels. Then, for each hierarchical level c (until $C-2$), and for each each block $u \in \mathcal{G}_c$, we check whether its corresponding child blocks are implemented vertically (i.e. $u \in \mathcal{S}^V$) or horizontally (i.e. $u \in \mathcal{S}^H$) and perform the allocation process accordingly (lines 2–10). We end the loop at $C-2$, since the child blocks of the blocks at level $C-1$ correspond to the products themselves. The part regarding the allocation of products will be treated in Subsection 5.1.3.

5.1.2.1 Vertical allocation

When allocating blocks that must be arranged vertically, the set of shelf levels in which each of the blocks will be present is automatically determined *a priori*, since they are required to occupy the same shelf levels as their parent block, whose allocation has already taken place. Moreover,

```
1  Allocate dummy block
2  for c ← 1 to C − 2 do
3  |  foreach u ∈ G_c do
4  |  |  if u ∈ S^V then
5  |  |  |  Allocate blocks vertically(V_u)
6  |  |  else
7  |  |  |  Allocate blocks horizontally(V_u)
8  |  |  end
9  |  end
10 end
```

Algorithm 2: Block allocation overview

given that no horizontal elasticity effects are taken into consideration in the problem under analysis, the sequence of blocks has no impact on the objective function and therefore an arbitrary sequence can be chosen. Notwithstanding that these conditions greatly simplify the problem at hand, we still need to decide on the width that should be assigned to each block, which will have a substantial impact on the subsequent allocation stages.

A straightforward approach to estimate the width of each block could be based on the number of facings computed in the previous stage. Considering that each product i occupies an expected linear space of $a_i \cdot TF_i$, we can obtain the total expected linear space of a block u by calculating the sum $\sum_{i \in u} a_i \cdot TF_i$ (it is worth keeping in mind that the linear space of a block can be larger than its width, provided that it is present in more than one shelf level). Then, each block could be attributed a width in direct proportion to its expected linear space, and considering that the sum of the widths of the considered block must be equal to the width of the parent block. Despite appearing to be a fair approximation, this represents a naive strategy in the sense that it neglects the fact that a product cannot be physically broken into smaller parts whenever the remaining space on a shelf is not sufficient to accommodate an additional facing of that particular product. This can be problematic, especially for narrower blocks, as their assigned space may not be enough to place one facing of each product on the shelves.

Under these conditions, we need, in some way, to ensure a minimum width to each block so that we do not compromise the allocation of products that will take place in the final stage of the heuristic. In order to guarantee a certain flexibility in the algorithm, this minimum width should be specific to each block (attending to its characteristics), rather than just being a fixed percentage of the width of the parent block, as this latter approach could potentially lead to very ineffective allocations. To tackle this issue, we first need to introduce the concept of *tightness* of a block u, denoted by τ_u, which corresponds to the ratio between the total linear space occupied by the products that belong to block u (considering, for each product $i \in u$, a number of facings equal to the lower bound l_i) and the block's attributed linear space, and is calculated according to Equation (5.7). The higher the value of τ_u, the more likely it is that block u will not have enough space to incorporate at least l_i facings of each product $i \in u$ during the product allocation stage. By imposing a maximum value to the block's tightness, τ_u^{max}, we can control its width based on the specific products that belong to it. The values of τ_u^{max} are an input to the constructive heuristic and can be fine-tuned by recurring to computational experiments.

$$\tau_u = \frac{\sum_{i \in u} a_i \cdot l_i}{K_u \cdot L_u}, \quad \forall u \in \mathcal{M} \tag{5.7}$$

Let $p \in \mathcal{M}$ denote the parent block of $u \in \mathcal{V}_p$ and α_u the proportion of the width of p that should be allocated to u. With this notation, we can rewrite $K_u L_u$ as $\alpha_u K_p L_p$. If we replace τ_u with τ_u^{max} in (5.7) and rearrange the terms, we can obtain the minimum width proportion to assign to block u, α_u^{min}, as indicated in Equation (5.8).

$$\alpha_u^{min} = \frac{\sum_{i \in u} a_i \cdot l_i}{\tau_u^{max} \cdot K_p \cdot L_p}, \quad \forall p \in \mathcal{M},\ u \in \mathcal{V}_p \tag{5.8}$$

We will now formulate a very simple goal programming model for adjusting the blocks' widths proportions considering a particular parent block p. Our goal is that α_u is as close as possible to $\alpha_u^{target} = \frac{\sum_{i \in u} a_i \cdot TF_i}{\sum_{j \in p} a_j \cdot TF_j}$, which is the value that best reflects what should be the proportion of the parent block's width to allocate to u, given the previously computed target facings. Hence, the objective function is as expressed in (5.9), with α_u corresponding to the set of decision variables. However, α_u must be at least equal to the minimum width proportion calculated according to (5.8), which gives rise to the set of constraints expressed in (5.10). Constraints (5.11) and (5.12) guarantee that all values of α_u are expressed in percentage terms. Note that if $\sum_{u \in \mathcal{V}_p} \alpha_u^{min} > 1$, no feasible solutions exist.

$$\text{Minimize} \quad \sum_{u \in V_p} |\alpha_u - \alpha_u^{target}| \tag{5.9}$$

$$\text{Subject to:} \quad \alpha_u \geq \alpha_u^{min}, \quad \forall u \in \mathcal{V}_p \tag{5.10}$$

$$\sum_{u \in V_p} \alpha_u = 1 \tag{5.11}$$

$$0 \leq \alpha_u \leq 1, \quad \forall u \in \mathcal{V}_p \tag{5.12}$$

This problem is solved using a heuristic that is presented in Algorithm 3. For each block, we calculate the difference between the target and minimum proportions, Δ_u (line 1). When the sum of those differences is positive or null, the problem is feasible (lines 5 and 6), since we can reduce the width proportion of blocks which have a positive Δ_u and increase the size of blocks with a negative Δ_u in a way that complies with constraint (5.10). That is not the case when $\sum_{u \in \mathcal{V}_p} \Delta_u < 0$ (lines 8 and 9) – in this situation, we reduce the space of blocks with positive Δ_u to a minimum (α_u^{min}) and increase the remaining blocks' width proportionally to how far each block is from its corresponding minimum value. After executing the algorithm, obtaining the actual width of each block u is trivial: $L_u = \alpha_u \cdot L_p, \forall u \in \mathcal{V}_p$.

1 $\Delta_u \leftarrow \alpha_u^{target} - \alpha_u^{min}, \quad \forall u \in \mathcal{V}_p$
2 $addedProp = -\sum_{u \in \mathcal{V}_p : \Delta_u < 0} \Delta_u$
3 $removedProp = \sum_{u \in \mathcal{V}_p : \Delta_u > 0} \Delta_u$
4 **if** $\sum_{u \in \mathcal{V}_p} \Delta_u \geq 0$ **then** `// Problem is feasible`
5 $\quad \alpha_u \leftarrow \alpha_u^{min}, \quad \forall u \in \mathcal{V}_p : \Delta_u \leq 0$
6 $\quad \alpha_u \leftarrow \alpha_u^{target} - \frac{\Delta_u}{removedProp} \cdot addedProp, \quad \forall u \in \mathcal{V}_p : \Delta_u > 0$
7 **else** `// Problem is infeasible`
8 $\quad \alpha_u \leftarrow \alpha_u^{target} - \frac{\Delta_u}{addedProp} \cdot removedProp, \quad \forall u \in \mathcal{V}_p : \Delta_u \leq 0$
9 $\quad \alpha_u \leftarrow \alpha_u^{min}, \quad \forall u \in \mathcal{V}_p : \Delta_u > 0$
10 **end**

Algorithm 3: Width proportions adjustment heuristic

5.1.2.2 Horizontal allocation

Allocating blocks in a horizontal fashion requires a different approach, due to the distinct nature of the problem. In this case, we have to decide on both the widths and the shelf levels on which a given block will be present. These two decisions are interrelated: the width of a block depends on whether or not it shares the same shelf with other block(s). If there is only a single block on a particular shelf level, that block's width will be equal to the width of its parent block. In the present approach, we start by determining the shelf levels on which each of the blocks should be present, and then decide on their widths.

The initial stage of the procedure for assigning horizontal blocks to specific shelf levels is presented in Algorithm 4. We start by calculating the adjusted width proportions of each block $u \in \mathcal{V}_p$ (line 2) using the procedure described in Algorithm 3, which will act as a mere reference point for defining the amount of space that ought to be allocated to each block. It is therefore pertinent to clarify the rationale behind the choice of these adjusted proportions over the target proportions (α_u^{target}), as the latter constitute a seemingly more intuitive reference point. However, note that the target proportion of a given block is based on the target facings of the products that comprise that same block which, in its turn, are calculated using a deterministic greedy heuristic. Hence, if we were to use the target proportions, the horizontal allocation would become highly sensitive to the characteristics of a particular instance (since they determine the target facings of each product), and be more likely to lead to infeasible solutions. By using the adjusted proportions, we can make use of the maximum tightness parameters (τ_u^{max}) to fine-tune the constructive heuristic when necessary. The remainder of the vertical allocation algorithm then tries to define an intuitive assignment, taking into consideration the referred proportions.

The expected number of shelf levels of a given block $u \in \mathcal{V}_p$ can be expressed as $K_u^{EXP} = \alpha_u \cdot K_p$. Let $[x]$ denote the number x rounded to the nearest integer value. We accumulate blocks on a given shelf until the rounded cumulative expected number of shelf levels (represented by $K^{cumulative}$) is greater than 1. Then, we move to the next shelf and repeat a similar process (lines 9–21). The rounding operator is justified by the fact that it prevents smaller blocks from occupying an excessive number of shelf levels. There are three factors that can break this main loop: (1) if all blocks for which $[K_u^{EXP}] \leq 1$ have already been allocated; (2) if there are no more shelves

available; or (3) if there no blocks left to allocate.

```
1  foreach u ∈ V_p do
2  |  α_u ← Calculate adjusted width proportions (α_u^target, α_u^min)
3  end
4  Sort all α_u by ascending order
5  K_p^available ← K_p
6  n ← 1                                  // Index of first element of V_p, sorted by α_u
7  k ← 1                                                  // Index of first element of K_p
8  while [α_n · K_p] ≤ 1 and K_p^available ≠ ∅ and n ≤ |V_p| do
9  |  K^cumulative ← α_n · K_p
10 |  Assign block n to shelf k
11 |  n ← n + 1
12 |  while [K^cumulative] ≤ 1 and n ≤ |V_p| do
13 |  |  K^cumulative ← K^cumulative + α_n · K_p
14 |  |  if [K^cumulative] ≤ 1 then
15 |  |  |  Assign block n to shelf k
16 |  |  |  n ← n + 1
17 |  |  else
18 |  |  |  Remove k from K_p^available
19 |  |  |  k ← k + 1
20 |  |  end
21 |  end
22 end
```

Algorithm 4: First stage of the procedure for assigning horizontal blocks to shelves

Note that this first stage alone may not be sufficient to generate a complete assignment of a set of blocks to a set of shelf levels. In fact, when the procedure described in Algorithm 4 ends, a total of six different possible situations can arise, associated with six different courses of action, which are summarized in Table 5.1. By following the proper course of action, the procedure for horizontally assigning blocks to shelf levels is completed.

Table 5.1: Possible courses of action after Algorithm 4

1	**There are no remaining empty shelves**
	(a) There are no blocks left to allocate – allocation is finished (***Situation #1***)
	(b) There are still blocks left to allocate – assign remaining blocks to the last shelf level (***Situation #2***)
2	**There are still empty shelves**
	(a) There are no blocks left to allocate – assign largest block to the remaining shelf/shelves (***Situation #3***)
	(b) There are still blocks left to allocate
	i. Number of empty shelves = Number of remaining blocks – assign a single block to each of the remaining shelves (***Situation #4***)
	ii. Number of empty shelves > Number of remaining blocks – assign blocks in such a way that the largest blocks get the highest number of shelves (***Situation #5***)
	iii. Number of empty shelves < Number of remaining blocks – assign the smallest among the remaining blocks to the last shelf which is not empty until the number of remaining blocks is equal to the number of empty shelves (***Situation #6***)

If a block u is the only one on a given shelf level, its width will be equal to the total width of the parent block p. On the other hand, if it shares the same shelf level with other blocks, the width is obtained by multiplying the width of the parent block by the ratio of α_u over the sum of all α_m (considering all blocks m that are also assigned to that shelf and belong to the same parent block).

This step concludes the horizontal allocation procedure. To the extent to which this is possible, this method tends to yield a fair arrangement of blocks, which is a complicated task given the constraints of the problem. A practical example of the application of the algorithm that was just described is present in Appendix C.

5.1.3 Product allocation

The allocation of products to shelves bears a close resemblance with the horizontal allocation of blocks, as the arrangement of products themselves cannot be L-shaped. As a result, having a separate block for each product in the last hierarchical level proves itself useful in this stage, as we can extend the outer loop of Algorithm 2 so as to include $c = C - 1$ and apply the same procedure as in 5.1.2.2 to the last level blocks. By doing this, we get to know which products go into each shelf. Note that, when a single product i is assigned more than one shelf level, the minimum (l_i) and maximum (u_i) number of facings are evenly divided between each of the shelf levels on which the product is present, so that the overall lower and upper bound constraints of the problem are not violated.

After assigning the products to the planogram's shelf levels, it is necessary to ensure that the width that was previously attributed to each block is enough to accommodate its corresponding products. If this is not a case, blocks need to be resized, in a process described in 5.1.3.1. Then, the target facings of each product are recalculated so as to account for the blocks' dimensions, and products are allocated on the shelves accordingly. The recalculation of target facings is performed using a dynamic programming (DP) algorithm, which is explained in 5.1.3.2.

5.1.3.1 Block resizing

While it is crucial for vertical blocks to be resized before products are placed on the shelves, that is not the case with horizontal blocks. This becomes easier to grasp with the example provided by Figure 5.2. In each implementation direction (vertical and horizontal), the left planogram depicts the result of the block allocation stage, with three blocks being considered. Let us assume that after products are assigned a particular shelf level, allocating the minimum number of facings for each product of block A on the top shelf leads to the block's original width being exceeded. The referred figure makes it clear that vertical blocks would have to be resized in such a situation, as the allocation of products causes a misalignment of blocks between the bottom and top shelves, thus compromising the required uniform rectangular shape. On the other hand, this situation would not be problematic for horizontal blocks, as their uniform rectangular shape would still be preserved.

As a result of this, it is necessary to verify whether the width that was previously attributed to each of the vertically implemented blocks (considering the dummy block as a vertical block) is sufficient to accommodate its corresponding products and, if not, have a mechanism that is able to adjust the blocks' widths. We start by calculating the minimum necessary width for each vertical block u, L_u^{min} in accordance with Equation (5.13), where PS_{ik} is an auxiliary binary variable which takes the value of 1 if the last level block that corresponds to product i has been allocated to shelf

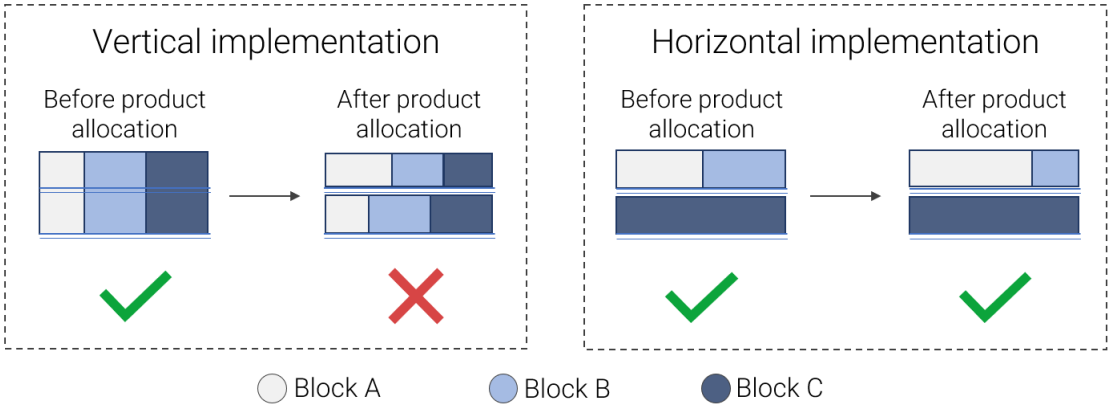


Figure 5.2: Adjusting the width of a block in a particular shelf level: comparison between vertical and horizontal blocks

k (and 0 otherwise). For each of the shelves on which block u is present, we calculate the width occupied by the minimum number of facings of the products of that block that are present on that specific shelf. By taking the maximum of those values, we get the minimum width that is required to ensure a uniform rectangular shape for block u.

$$L_u^{min} = \max_{k \in \mathcal{K}_u} \left\{ \sum_{i \in \mathcal{N}_u} a_i \cdot l_i \cdot PS_{ik} \right\} \tag{5.13}$$

Then, we adjust the blocks' widths by applying the procedure that was described in Algorithm 3, *mutatis mutandis*. In this case, α_u^{target} denotes the current width proportion, whereas α_u^{min} is equal to $L_u^{min}/L_p, \forall u \in \mathcal{V}_p$. In the cases where $\sum_{u \in \mathcal{V}_p} L_u^{min} > L_p$, the adjustment will lead to certain blocks not having enough space for their corresponding products in some shelves and, as a consequence, it is highly likely that the algorithm will yield an infeasible solution. In this case, it is necessary to modify the τ_u^{max} parameters and restart the block allocation process. After a block's width is adjusted, all widths of its corresponding downstream blocks are also corrected proportionally.

5.1.3.2 Product placement

With the blocks' widths adjusted, it is now possible to start allocating products, in a process that is performed one shelf level at a time. As seen in Figure 5.2, the allocation of products must not lead to the vertical blocks' widths being exceeded. For each shelf $k \in \mathcal{K}$, we consider a set of *reference blocks*, $\mathcal{R}_k$, which consists of vertical blocks that are present on shelf k, that together contain all products that are placed on shelf k. When a vertical block is upstream from other vertical blocks that are present on the same shelf level, only the blocks at the lowest hierarchical level are considered as reference blocks, so as to avoid having overlapping blocks in $\mathcal{R}_k$. The width of each of these blocks corresponds to the space that is available, in each of the shelf levels on which the block is present, for placing its corresponding products. This capacity limit guarantees

that the reference blocks are aligned across all shelves, and that therefore their uniform rectangular shapes are maintained.

For each reference block on each of the shelves of the planogram, we will have a separate allocation problem which is equivalent to recalculating the target facings for each product. As seen in Subsection 5.1.1, after ensuring a minimum number of facings to each product, a Bounded Knapsack Problem is obtained. The BKP is then converted into an equivalent Knapsack Problem, as described in the referred subsection. To address it, however, a different approach is needed: given the significantly smaller knapsack capacity, a greedy heuristic is very likely to lead to poor allocation decisions. Considering the relatively small size of each KP, a dynamic programming method was used. With this approach, we get an optimal allocation of products for any given block configuration determined in the previous step. DP is a technique that can be successfully applied to problems whose optimal solution is a combination of optimal solutions to sub-problems (Kellerer et al., 2004) – the so-called *optimal substructure* property. As the authors point out, that is the case with the KP, which can therefore be solved using DP. In order to successfully implement a DP algorithm and solve the corresponding sub-problem for each reference block, it is necessary to multiply the block's width and each item's profit and width by a large enough value (denoted by A), such that all of these values are converted into integer values.

Given that a conventional DP algorithm is implemented and that the mathematical formulation of the problem is, *mutatis mutandis*, the same as in Subsection 5.1.1, we present a broad overview of the method that is described by Kellerer et al. (2004). For any given reference block $p \in \mathcal{R}_k$, let $KP_j^{pk}(d)$ represent the corresponding Knapsack Problem with total capacity d (with $d = 0, 1, \ldots, A \cdot L_p$) and with the set of items $\{1, \ldots, j\}$. In this case, each item that is added to the knapsack represents an additional facing of a particular product on that shelf, with the number of facings of each product being limited by the corresponding upper bound. Let $s_j^{pk}(d)$ denote the optimal solution value of problem $KP_j^{pk}(d)$, with $s_0^{pk}(d) = 0$ for all considered values of d. Given the optimal substructure of the KP, if $s_{j-1}^{pk}(d)$ is known for all values of d, we can consider one additional item (i.e. product facing) and compute the corresponding optimal solutions $s_j^{pk}(d)$ using recursion (5.14).

$$s_j^{pk}(d) = \begin{cases} s_{j-1}^{pk}(d) & \text{if } d < A \cdot a_j \\ \max\{s_{j-1}^{pk}(d),\ s_{j-1}^{pk}(d - A \cdot a_j) + A \cdot p_j\} & \text{if } d \geq A \cdot a_j \end{cases} \tag{5.14}$$

This recursion yields a table s^{pk} (with the values of j and d as rows and columns, respectively), which is why this method is commonly referred to as the tabulation method. Considering J to be the last value of j to be considered (i.e. last row of the table), the optimal solution value is given by $s_J^{pk}(A \cdot L_p)$. However, this recursion does not explicitly provide the items that were selected to obtain that value. To achieve that, we store, for every value of d in iteration j, whether item j was added to the knapsack or not. For that purpose, we define the binary variable $F_j(d)$ as in (5.15).

$$F_j(d) = \begin{cases} 1 & \text{if } s_j^{pk}(d) = s_{j-1}^{pk}(d - A \cdot a_j) + A \cdot p_j \\ 0 & \text{if } s_j^{pk}(d) = s_{j-1}^{pk}(d) \end{cases} \tag{5.15}$$

This variable takes the value of 1 whenever item j is added to the knapsack in iteration j. Hence, if $F_J(A \cdot L_p) = 1$, item J is included in the final solution to the KP and we then proceed by checking the value of $F_{J-1}(A(L_p - a_j))$. On the other hand, if $F_J(A \cdot L_p) = 0$, then item J is excluded from the optimal solution and we proceed with $F_{J-1}(A \cdot L_p)$. This method computes both the optimal solution to the KP and its corresponding value in pseudo-polynomial time and space, which is acceptable given the small size of the sub-problems.

Having determined the number of facings of each product, it is possible to place them on their corresponding shelves. The left and right coordinates of each reference block delimit the interval of horizontal locations where its products can be placed. As referred above, the horizontal sequence of products is arbitrary, since no horizontal elasticity effects are taken into consideration, with the only constraint being that two facings of the same product are always placed on adjacent locations. Finally, for aesthetic purposes, for each reference block, we spread the items over the available shelf space such that the space between two adjacent items is constant. With this procedure repeated for each reference block in each shelf level, the product allocation stage is complete and we obtain a fully defined planogram, as in Figure 5.3. In this example, a total of four knapsack problems are solved during the product allocation stage, given that there are two reference blocks in each of the two shelf levels.

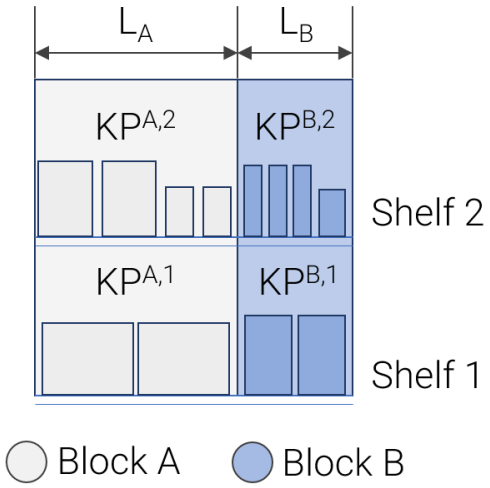


Figure 5.3: Final planogram after product allocation

5.2 Iterated Local Search

Iterated Local Search is a metaheuristic proposed by Lourenço et al. (2010) that applies perturbation mechanisms to locally optimal solutions that were previously obtained by applying a local search mechanism. The procedure is conceptually simple and is depicted in Algorithm 5. Three main stages comprise the ILS metaheuristic: (1) local search, where any search procedure can be used (including other metaheuristics); (2) perturbation, where we modify a specific incumbent solution to a reasonable extent, in an attempt to escape from local optima; and (3) acceptance

criterion, in which a rule, that can be either deterministic or stochastic, is used to determine which solution should be subject to the next perturbation. Subsections 5.2.1, 5.2.2, and 5.2.3 will illustrate each of this steps in the context of the present Shelf Space Allocation Problem and Subsection 5.2.4 presents an overview of the method that was implemented, given the previously described steps.

$s_0 \leftarrow$ Generate Initial Solution
$s^* \leftarrow$ Local search (s_0)
repeat
  $s' \leftarrow$ Perturbation(s^*, history)
  $s^{*\prime} \leftarrow$ Local search (s')
  $s^* \leftarrow$ Acceptance Criterion(s^*, $s^{*\prime}$, history)
until Termination condition is met

Algorithm 5: Iterated Local Search (Lourenço et al., 2010)

5.2.1 Local search

As aforementioned, the problem under analysis is highly constrained, which leads to a very narrow space of feasible solutions. Hence, local search mechanisms should make use of a limited number of moves so as to avoid spending the majority of the time evaluating infeasible solutions. In our local search phase, we do not alter the blocks' widths, which are assumed to be fixed throughout the procedure. Such modifications are performed exclusively on the perturbation phase. Furthermore, as the placement of products on the space allocated to each of the reference blocks in each shelf level is known to be optimal, it is pointless to have mechanisms that modify the number of facings of the products in each shelf, as they cannot, on their own, improve the current solution. As a consequence, the only local search mechanism will consist in modifying the sequence of blocks that belong to the same parent block, as depicted in Figure 5.4. These modifications will only take place in horizontally implemented blocks, since horizontal elasticity effects are disregarded.

A local search move starts by randomly selecting a block $p \in \mathcal{S}^H$ that will have its corresponding sequence of child blocks modified. Blocks with larger sequences will have a correspondingly higher probability of being selected, in order to reduce the number of cases in which the exact same move is applied multiple times. Given that horizontal blocks can only swap places in a given sequence if they have the same width, blocks that share a shelf level with other block(s) will be temporarily treated as a single temporary block. There may be occasions in which an infeasible solution is reached, as certain moves may lead to products being placed on shelves on which they do not fit due to their height. In those cases, a strong negative penalization is applied to the corresponding objective function, to ensure that that solution is discarded.

The total number of iterations in each local search is limited to a maximum value. After each iteration, two criteria for accepting a new solution as the incumbent solution are considered: (1) *better* criterion, in which we only accept a given solution as the incumbent if there is an improvement in the objective function (stronger intensification); and (2) *random walk* criterion,

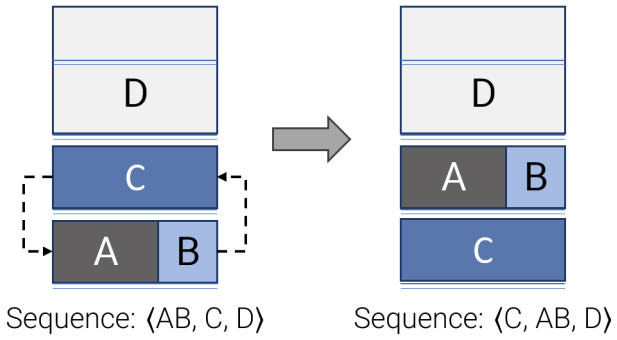


Figure 5.4: Example of a local search move

in which we accept the most recent solution as the incumbent, regardless of the differences in the objective function (stronger diversification).

5.2.2 Perturbation mechanism

The perturbation mechanism is responsible for modifying the current solution to an extent that allows it to escape from local optima. To allow for the necessary diversification of the search space, the blocks' dimensions must be changed. Rather than directly redefining the blocks' widths or placement, a perturbation alters the number of target facings of each product. Then, the different blocks and products are allocated according to the algorithms described in Subsections 5.1.2 and 5.1.3. Note that, as indicated in Figure 5.1, the block allocation methods depend on the output of the target facings calculation stage. Hence, a direct manipulation of this output will lead to a different block arrangement and, consequently, to a different planogram. In this case, however, no constraints on the blocks' maximum tightness are posed, which allows the algorithm to explore a higher range of block configurations. Thus, τ_u^{max} is a parameter that is only considered during the constructive procedure.

Algorithm 6 describes the process for modifying the target number of facings. Since each product must comply with its minimum display requirements, we can remove up to a total of $\sum_{i \in \mathcal{N}}(TF_i - l_i)$ target facings from the incumbent solution. The strength of the perturbation is determined by the parameter β, which represents the proportion of target facings that is removed from the current solution (line 1). Accordingly, a β of 0 implies keeping the current solution unchanged, whereas a β of 1 implies discarding the current solution altogether. With the value of β defined, the corresponding number of target facings is removed (lines 2–5). Finally, facings are added until the shelf capacity is reached (lines 8–22). The main difference between the procedure described in Algorithm 1 (which corresponds to the greedy heuristic used for calculating the initial target facings) and the one under analysis is the fact that while the former is deterministic, the latter is stochastic. The element of randomness is incorporated when selecting products for removal (line 3) and insertion (line 16) of facings. This selection can be made using two different methods: (1) *blind* selection, in which each product has the same probability of being selected for removal/insertion of facings; and (2) *biased* selection, in which, for each product i, the higher

the ratio p_i/a_i, the lower the probability of being selected for removal and, conversely, the more likely it is to be selected for insertion. Whenever it is not possible to select a particular product for insertion or removal, the corresponding selection probability is set to 0.

$facingsToRemove \leftarrow [\beta \times \sum_{i \in \mathcal{N}}(TF_i - l_i)]$
for $i \leftarrow 1$ ***to*** $facingsToRemove$ **do**
  $i^* \leftarrow$ Select product to remove target facing
  $TF_{i^*} \leftarrow TF_{i^*} - 1$
end
$allocationComplete \leftarrow false$
$emptySpace \leftarrow \sum_{k \in \mathcal{K}} w_k - \sum_{i \in \mathcal{N}} a_i \cdot TF_i$
while *allocationComplete* = *false* **do**
  $flag \leftarrow false$
  for $i \leftarrow 1$ ***to*** N **do**
    if $a_i \leq emptySpace$ **and** $TF_i < u_i$ **then**
      $flag \leftarrow true$
    end
  end
  if *flag* = *true* **then**
    $i^* \leftarrow$ Select product to add target facing
    $TF_{i^*} \leftarrow TF_{i^*} + 1$
    $emptySpace \leftarrow emptySpace - a_{i^*}$
  else
    $allocationComplete \leftarrow true$
  end
end

Algorithm 6: Perturbation mechanism: modify target facings

Using the target facings to perturb the incumbent solution brings two main advantages: (1) we are automatically ensuring compliance with the defined set of constraints (except in the cases in which the block allocation stage leads to infeasible arrangements); and (2) we can easily control the strength of the perturbation by changing the parameter β, which can be useful to diversify the search after several number of iterations without improvement.

5.2.3 Acceptance criteria

The acceptance criteria procedure determines when to accept a new solution as the incumbent solution to which the perturbation mechanism will be applied. In the present approach, three different acceptance criteria are considered. The first two are the *better* and the *random walk* that are analogous to the ones described in the local search phase, with the corresponding acceptance functions being as defined in (5.16) and (5.17), where s^* denotes the incumbent solution (i.e. the solution that was subject to a perturbation), $s^{*\prime}$ represents the newly obtained solution, and Z corresponds to the objective function. The third one uses a stochastic acceptance function that is commonly used in Simulated Annealing algorithms, with better solutions being always accepted and worse solutions being accepted with a certain probability, as in Equation (5.18). Following Stützle (2006), we denote this acceptance criterion as $LSMC(s^*, s^{*\prime}, T, X)$, which stands for *large step Markov chains* (this was the designation that was commonly attributed to early ILS algorithms). Here, X represents a random variable that follows a uniform distribution over $[0,1]$ and T corresponds to the temperature parameter of the SA framework. T is updated according to a geometric cooling schedule, as in $T(n) = T_0 \cdot \rho^n$, where T_0 corresponds to the initial temperature,

ρ denotes a constant in the interval $(0,1)$, and n indicates the current iteration.

$$Better(s^*, s^{*\prime}) = \begin{cases} s^{*\prime} & \text{if } Z(s^{*\prime}) > Z(s^*) \\ s^* & \text{otherwise} \end{cases} \tag{5.16}$$

$$RandomWalk(s^*, s^{*\prime}) = s^{*\prime} \tag{5.17}$$

$$LSMC(s^*, s^{*\prime}, T, X) = \begin{cases} s^{*\prime} & \text{if } Z(s^{*\prime}) > Z(s^*) \text{ or } X \leq e^{\frac{s^{*\prime} - s^*}{T}} \\ s^* & \text{otherwise} \end{cases} \tag{5.18}$$

5.2.4 Overall procedure

Having described each of the main stages of the ILS framework, a holistic, high-level perspective of the method that was implemented in the context of the present work is now provided. Algorithm 7 describes the overall procedure when *better* and *random walk* acceptance criteria are being used. After generating the initial solution and performing a search over its neighborhood (lines 1 and 2), the parameter β is set to an initial value, β_0, (line 5) which will determine the strength of the first perturbations. After executing each of the three main stages of ILS that were presented in the previous subsections (lines 7–9), the new incumbent solution, s^*, is compared the best solution found so far, s^{best} (lines 10–13). After a given number of iterations without improvement, the strength of the perturbation is increased by a value of $step$ (lines 16–19). The procedure stops when β can no longer be increased (line 20). Algorithm 8 presents the method when LSMC is used as acceptance criterion, whose main difference is the introduction of the temperature parameter. After a certain number of perturbations (*maxIterPerTemperature*), the temperature decreases (lines 23–27).

```
1  s_0 ← Generate Initial Solution
2  s* ← Local search(s_0)
3  s^best ← s*
4  nIterWithoutImprovement ← 0
5  β ← β_0
6  repeat
7      s' ← Perturbation(s*, β, strategy)
8      s*' ← Local search(s')
9      s* ← Acceptance Criterion(s*, s*')
10     if Z(s*) > Z(s^best) then
11         s^best ← s*
12         nIterWithoutImprovement ← 0
13     else
14         nIterWithoutImprovement ← nIterWithoutImprovement + 1
15     end
16     if nIterWithoutImprovement = maxIterWithoutImprovement then
17         β ← β + step
18         nIterWithoutImprovement ← 0
19     end
20 until β > 1
```

Algorithm 7: Overall procedure (*better* and *random walk* acceptance criteria)

```
1  s_0 ← Generate Initial Solution
2  s* ← Local search(s_0)
3  s^best ← s*
4  β ← β_0
5  n, countIterSA, nIterWithoutImprovement ← 0
6  T ← T_0 · ρ^n
7  repeat
8      s' ← Perturbation(s*, β, strategy)
9      s*' ← Local search(s')
10     X ← Random(0, 1)
11     s* ← Acceptance Criterion(s*, s*', T, X)
12     if Z(s*) > Z(s^best) then
13         s^best ← s*
14         nIterWithoutImprovement ← 0
15     else
16         nIterWithoutImprovement ← nIterWithoutImprovement + 1
17     end
18     if nIterWithoutImprovement = maxIterWithoutImprovement then
19         β ← β + step
20         nIterWithoutImprovement ← 0
21     end
22     countIterSA ← countIterSA + 1
23     if countIterSA = maxIterPerTemperature then
24         n ← n + 1
25         countIterSA ← 0
26         T ← T_0 · ρ^n
27     end
28 until β > 1
```

Algorithm 8: Overall procedure (*LSMC* acceptance criterion)

Chapter 6

Computational results

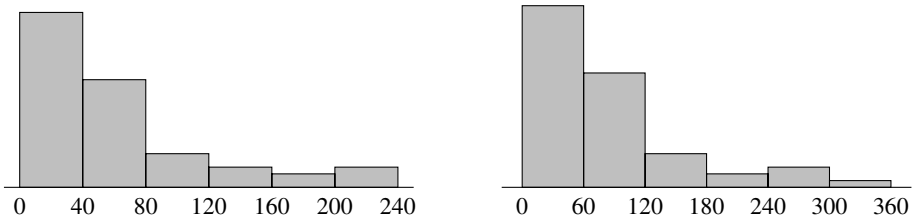
This chapter is dedicated to the set of computational experiments that were performed in order to test the quality of the generated solutions via the heuristic procedure. All algorithms described in the previous chapter were implemented in C++. The tests were conducted on an Intel® Xeon® E5-2640 v3 2.60GHz processor and 16GB of RAM were available.

The structure of the present chapter is as follows. Section 6.1 provides a short description of the instances that were used to test our approach. In Section 6.2, the evaluation function used to assess the quality of the obtained solutions is presented. Sections 6.3 and 6.4 provide additional information regarding the computational tests that were performed using the constructive procedure and the Iterated Local Search metaheuristic and present the corresponding results. Lastly, Section 6.5 explains how the output of the heuristic is converted into a planogram.

6.1 Problem instances

The developed algorithms were tested on a set of benchmark instances provided by Bianchi-Aguiar et al. (2014), for which a brief overview will be given in the present section. For a more detailed description, we refer to Bianchi-Aguiar et al. (2018b). A total of 54 instances were considered, all of which were originally obtained from a European grocery retailer. Each of these instances provides the necessary input information (with respect to the sets of products, shelves, and merchandising rules) to solve the problem under analysis, given the formulation that was introduced in Chapter 4. Key information regarding each of the considered instances is summarized in Appendix D.

The histograms in Figure 6.1 are illustrative of the size of the instances that we are dealing with. While in most cases the number of considered products (6.1(a)) and blocks (6.1(b)) is moderate (with nearly 50% of the instances containing less than 40 products and less than 60 blocks), there are some examples which are substantially larger. Hence, this set of instances will allow for the evaluation of the algorithm's performance on a wide range of scenarios. In Figure 6.1(b), we are considering the last level blocks (which correspond to individual products) in the total number of blocks, as they add to the problem's complexity.



(a) Number of products (b) Total number of blocks

Figure 6.1: Distribution of the instances according to the number of products (a) and blocks (b)

6.2 Evaluation function

In order to allow for a fair comparison, the objective function from Bianchi-Aguiar et al. (2018b) was used, as in Equation (4.1). However, the referred equation alone does not account for the infeasibilities that may arise from the application of the method. For that purpose, it is necessary to introduce a penalty function so that infeasible solutions can be discarded.

The procedure that was described in Section 5.1 can lead to infeasible solutions under three different situations: (1) when a product is placed on a shelf whose height is smaller than the product's height; (2) when the length of all products that are placed on a given shelf exceeds the shelf's length; and (3) when there are reference blocks which are not aligned across all shelf levels on which they are present. Let e_1, e_2, and e_3 denote the excess height (in case 1) or width (in cases 2 and 3) that is associated with each of these situations, which are calculated as shown in Equations (6.1) – (6.3), with $(x)^+$ denoting the positive part of x. The total penalization P is calculated according to (6.4), with ϕ_s corresponding to the weight attributed to each deviation $s \in \{1,2,3\}$. The deviation values are squared so that a greater emphasis is placed on stronger deviations, which will be useful to guide the search towards better solutions during the improvement stage. Finally, the total penalization is deducted from the objective function, with the final objective function of the heuristic procedure (Z^H) being computed as $Z^H = Z - P$.

$$e_1 = \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} W_{ik} \cdot (b_i - h_k)^+ \tag{6.1}$$

$$e_2 = \sum_{k \in \mathcal{K}} \left[\Big(\sum_{i \in \mathcal{N}} a_i \cdot W_{ik} \Big) - w_k \right]^+ \tag{6.2}$$

$$e_3 = \sum_{k \in \mathcal{K}} \left[\Big(\sum_{\substack{u \in \mathcal{R}_k \\ i \in \mathcal{N}_u}} a_i \cdot W_{ik} \Big) - L_u \right]^+ \tag{6.3}$$

$$P = \phi_1 e_1^2 + \phi_2 e_2^2 + \phi_3 e_3^2 \tag{6.4}$$

6.3 Constructive heuristic

One of the main challenges of the present dissertation was to design a constructive heuristic that was able to generate a good initial solution within reasonable running times. Having out-

lined the procedure, it was necessary to test the algorithms in order to find whether the developed approach was viable.

6.3.1 Parameter tuning

The constructive heuristic takes one input parameter for each block u, which corresponds to its maximum tightness, τ_u^{max}. In order to test the performance of the constructive heuristic on its own, it was necessary to find appropriate values for such parameters. Given the extent of existing possibilities, individually setting τ_u^{max} for each block would be impractical. Thus, we divided τ_u^{max} into a sum of three separate components, as in (6.5), and then performed a simple parameter search on these three parameters. The *base* component corresponds to a baseline value. Then, if the block is horizontally implemented, $horizontal_u$ is added to τ_u^{max}. Finally, the last component to be added is proportional to the block's hierarchical level. From this formulation, one can easily conclude that blocks that are on the same hierarchical level and have the same display orientation will have the same values of τ_u^{max}.

$$\tau_u^{max} = base + horizontal_u + (c-1) \cdot level_u, \quad \forall u \in \mathcal{M} \cup \mathcal{N} \tag{6.5}$$

To obtain appropriate values for each of the three parameters, we conducted a random search. This procedure was selected given its simplicity and the fact that the impact of the parameters on the objective function is unclear and therefore hard to predict beforehand. Furthermore, as Bergstra and Bengio (2012) point out, a random search is able to find parameter configurations which are as good or even better than the ones obtained by grid search within substantially smaller computation times. The ranges of the values tested for each component are indicated in Table 6.1.

Table 6.1: Ranges tested for each component of maximum tightness

$base \sim Uniform(0.15, 0.85)$
$horizontal \sim Uniform(-0.05, 0.50)$
$level \sim Uniform(-0.05, 0.15)$

6.3.2 Summary of results

Summary statistics of the results obtained using the constructive procedure alone are listed on the left column of Table 6.2. For comparison purposes, the third scenario of Bianchi-Aguiar et al. (2018b) (which is the only one that considers product families to their full extent) is used as reference. A total of five models are presented in the referred work: the main model for the block allocation problem (BAP); the linear relaxation of the BAP model (BAP-LR), which can be seen as a theoretical upper bound; the matheuristic that solves BAP hierarchically (H-BAP); and the model extensions designed to improve feasibility and efficiency (H-BAP-IF and H-BAP-IE, respectively). For each instance, the relative difference (abbreviated by "Diff (%)") between the results obtained using the heuristic procedure and a particular model is computed. As an example, the relative difference to BAP-LR can be calculated as $\frac{(Z^H - Z^{lr})}{Z^{lr}}$, with Z^{lr} denoting the result obtained via linear relaxation.

Concerning computational efficiency, Figure 6.2 graphically depicts the running time (in seconds) of the constructive heuristic as a function of the total number of blocks (including last level blocks, i.e. $|\mathcal{M} \cup \mathcal{N}|$), which serves as a proxy for the size of an instance. In most cases, the method runs relatively fast, with 80% of the solutions being generated in less than 2 seconds. However, as the instance sizes get larger, computation times start to increase in what appears to be a non-linear fashion. While this can get problematic, the impact will only become clear on sufficiently large instances (with $|\mathcal{M} \cup \mathcal{N}| > 200$), which constitute only 11% of all cases.

A total of 50 (out of 54) feasible solutions were generated, which is already an improvement compared to the results of Bianchi-Aguiar et al. (2018b), which reported 40, 42, 47, and 49 feasible solutions for the BAP, H-BAP, H-BAP-IF, and H-BAP-IE models, respectively (disregarding solutions that lead to negative objective function values). Moreover, the average solution is around 8.0% worse than the ones obtained using the H-BAP-IE (their most efficient model with respect to computation time, thus allowing for fairer comparisons) while requiring, on average, just 4.8% of the computation time, which indicates significant efficiency gains when only the constructive stage is applied. Additionally, the average difference with respect to the upper bound (BAP-LR) is of -19.2%.

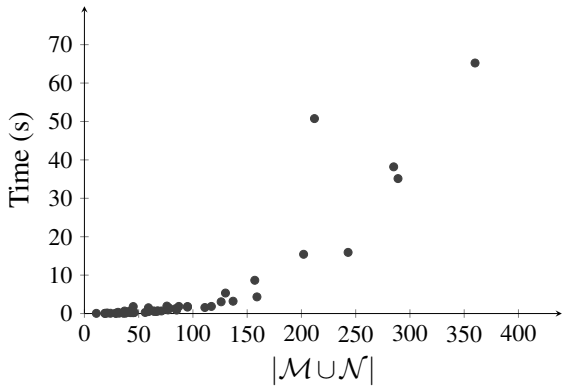


Figure 6.2: Running time as a function of the instance size

The full set of results concerning the constructive phase be found in Table 6.3, in which, for each instance, the running time of a single run of the constructive procedure is shown.

6.4 Iterated Local Search

6.4.1 Parameter tuning

The ILS metaheuristic, as described in Algorithms 7 and 8, requires the definition of a number of input parameters. To begin with, it is important to define the number of consecutive iterations without improvements in the objective function that must be performed before increasing the perturbation strength (*maxIterWithoutImprovement*). Rather than specifying a fixed value for the parameter, we wanted the parameter to be instance-specific, as larger instances will likely need more iterations to allow for a proper search over the solution space. Hence, for each instance,

we wanted the maximum number of iterations without improvement to be a function of the total number of blocks (including last level blocks), $|\mathcal{M} \cup \mathcal{N}|$, as in (6.6).

$$maxIterWithoutImprovement = \left\lceil \lambda \times |\mathcal{M} \cup \mathcal{N}| \right\rceil \tag{6.6}$$

Where λ is a positive constant and $\lceil x \rceil$ corresponds to the ceiling function applied to x. In order to find an appropriate value for λ, we first selected five instances with a good potential for improvement (using the results published in Bianchi-Aguiar et al. (2018b) as a reference point): AB_1, AI_3, AI_4, FL_1, and SM_5. Then, we applied the ILS method considering different values for λ. We tested λ for all values in the set $\{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$. For this specific purpose of fine-tuning the parameter λ, the perturbation strategy at each iteration was chosen randomly between *blind* and *biased*, with the acceptance criterion being also picked at random between *better* and *random walk*. The results are presented in Figure 6.3. While in some cases (namely, AB_1 and AI_4), a substantial improvement is obtained even for small values of λ, some instances (particularly AI_3 and SM_5) require a larger number of iterations in order to reach significant improvements to the initial solution. As a result, the value of $\lambda = 0.2$ was selected for the ILS procedure, given that no substantial improvements take place for larger values of λ in any of the considered situations (there is only a slight increase from $\lambda = 0.20$ to $\lambda = 0.25$ in SM_5).

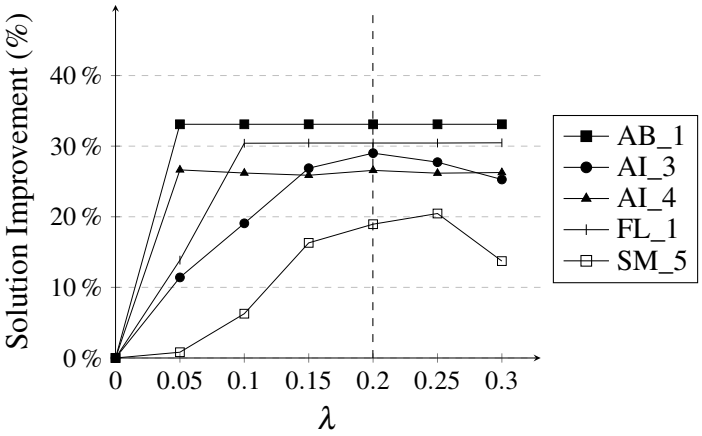


Figure 6.3: Solution improvement as a function of λ

The ILS method considered an initial perturbation strength $\beta_0 = 0.20$ and stepped increases of 0.20. Each of the three acceptance criteria (*better*, *random walk*, and *LSMC*) was tested separately.

Using the LSMC criterion, however, implies establishing a *cooling schedule*, $T(n)$, which is a function that specifies how the temperature parameter T varies with the number of iterations n. As indicated in Algorithm 8, the temperature varies according to a geometric progression $T(n) = T_0 \cdot \alpha^n$. The initial temperature T_0 was selected such that, after the first perturbation (i.e. $n = 0$), the probability of accepting a solution which was 25% worse than the best solution found so far (which would be, in the first iteration, the initial solution) would be 75%. Additionally, α was obtained by defining a final temperature and a number of different temperatures to be tested

(30). The final temperature was computed considering a 0.5% probability of accepting a solution which was 25% worse than the best one found so far. For each temperature T, 5 iterations were performed (i.e. $maxIterPerTemperature = 5$).

Regarding the perturbation strategy, three possibilities were tested. Apart from the aforementioned *blind* and *biased* strategies, a *mixed* strategy that randomly selected one of these two alternatives at each perturbation was also considered. As for the local search acceptance function, we only used the *random walk* criterion. Given that the local search procedure is already very restrictive by itself, using a *better* criterion would not be a very effective strategy.

Finally, to avoid excessively large computation times, two measures were undertaken: (1) the maximum number of iterations without improvement was restricted to a maximum value of 15 (as in Equation (6.7)); and (2) a time limit of 10 minutes was imposed.

$$maxIterWithoutImprovement = \begin{cases} \lceil \lambda \times |\mathcal{M} \cup \mathcal{N}| \rceil & \text{if } \lceil \lambda \times |\mathcal{M} \cup \mathcal{N}| \rceil \leq 15 \\ 15 & \text{otherwise} \end{cases} \tag{6.7}$$

6.4.2 Results

Table 6.2 summarizes the results that were obtained by applying the Iterated Local Search algorithm, where ILS-B, ILS-RW, and ILS-LSMC indicate the ILS method that uses the *better*, *random walk*, and *LSMC* acceptance criterion, respectively. The *random walk* criterion is the one that most often leads to the best solution (for 14 instances) and yields the highest average improvement to the initial solution, of 4.8%, which appears to be an indication that strongly diversifying the search can be an effective strategy for solving our problem. Nevertheless, not only are the results very similar to the ones obtained using other acceptance criteria, but also *Random walk* has the longest average running time. When compared with the constructive procedure alone, an additional feasible solution was found for all considered criteria, thus leading to a new total of 51 feasible solutions.

As expected, the average relative differences to the models of Bianchi-Aguiar et al. (2018b) become smaller after applying Iterated Local Search to the initial solution. Compared with H-BAP, the average difference (in absolute terms) is slightly above 6%, which reduces to less than 4% for the H-BAP-IE model. It is therefore possible to conclude that our approach, when compared with state-of-the-art methods, leads to solutions of similar quality.

With the configuration of parameters that was used for the ILS procedure, the average running times are still substantially smaller than the average running time of the BAP (2249.9 seconds), H-BAP (559.6 seconds), and H-BAP-IF (517.6 seconds) models, but are higher than the average running time reported for the H-BAP-IE model (which was of 105.5 seconds). However, this computation time increase from H-BAP-IE to ILS may not be very meaningful on its own, as it may be the case that a good solution is found rather quickly when using the ILS procedure, with the remainder of the time being spent without actually improving the solution. In this case, reducing the maximum number of iterations without improvement would reduce computation time and would probably have little or no impact on the quality of the obtained solution.

Table 6.2: Summary of the results obtained for the constructive procedure and the ILS

	Constructive	**ILS-B**	**ILS-RW**	**ILS-LSMC**
Average running time (s)	5.1	205.5	232.3	202.8
# Best result	–	9	14	12
# Feasible solutions	50	51	51	51
Average improvement	–	4.5%	4.8%	4.7%
Average diff (%) to BAP-LR	-19.2%	-15.6%	-15.4%	-15.5%
Average diff (%) to BAP	-9.2%	-4.2%	-3.9%	-3.9%
Average diff (%) to H-BAP	-10.9%	-6.5%	-6.2%	-6.4%
Average diff (%) to H-BAP-IF	-10.4%	-6.1%	-5.8%	-5.9%
Average diff (%) to H-BAP-IE	-8.0%	-3.9%	-3.6%	-3.7%

Figure 6.4 indicates, for each instance, the relative difference (in percentage terms) between the results obtained using Iterated Local Search (considering the best solution out of the three acceptance criteria), H-BAP-IF, and H-BAP-IE and the upper bound Z^{lr}, which is obtained via linear relaxation of the BAP model (BAP-LR). The points of H-BAP-IF and H-BAP-IE situated above the ILS line correspond to situations in which these models yield better solutions (i.e. closer to the upper bound) than the heuristic approach and, conversely, points below the referred line correspond to worse solutions.

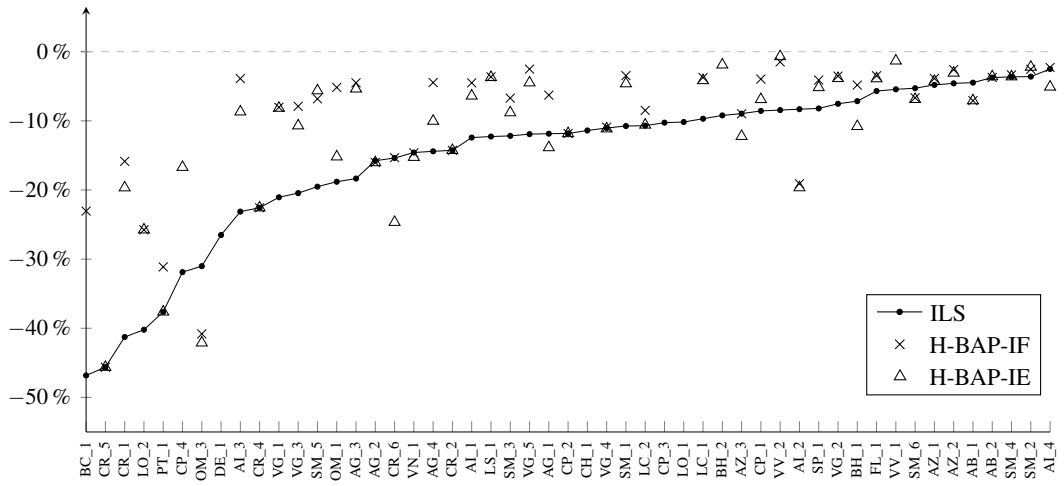


Figure 6.4: Relative difference between the results yielded by the ILS, H-BAP-IF, and H-BAP-IE methods and the upper bound Z^{lr} obtained via linear relaxation of the BAP model

The detailed resulted of the Iterated Local Search procedure are displayed in Table 6.3. For each acceptance criterion, we display the best result out of the three that were obtained (one for each perturbation strategy). As for the penalty function, we considered the coefficients ϕ_1, ϕ_2, and ϕ_3 to be equal to 100,000. Hence, the cases in which the objective function is negative correspond to infasible solutions. In the referred table, the last column indicates the best improvement (in percentage terms) with respect to the initial solution. The average best improvement was 5.0%, with a markedly high dispersion (CV = 1.67) in the considered values.

Finally, in Table 6.4, the best result obtained after applying the ILS metaheuristic is compared with the results published by Bianchi-Aguiar et al. (2018b) (Scenario 3). Only feasible solutions were compared.

Table 6.3: Results and performance of the heuristic method

Instance	**Constructive**		**ILS**						
			ILS-B		**ILS-RW**		**ILS-LSMC**		**Best**
	Z^H	Time (s)	Z^H	Time (s)	Z^H	Time (s)	Z^H	Time (s)	**improvement (%)**
AG_1	2349.7	1.8	2349.7	102.2	2406.6	250.3	2349.7	180.1	2.4%
AG_2	2011.2	0.2	2139.5	25.1	2139.5	47.3	2139.5	38.7	6.4%
AG_3	27267.5	3.2	27371.9	390.0	27433.9	TLR	27416.8	507.4	0.6%
AG_4	4760.7	1.9	4932.0	230.6	4914.8	472.8	4925.2	317.7	3.6%
AI_1	3443.1	0.8	3445.6	76.7	3449.0	123.0	3444.6	90.0	0.2%
AI_2	6150.0	0.7	6317.1	138.0	6492.2	230.0	6391.5	160.6	5.6%
AI_3	1930.0	0.8	2425.6	131.3	2478.7	173.5	2486.4	165.2	28.8%
AI_4	15619.9	0.6	19644.2	92.4	19761.6	260.0	19664.3	155.5	26.5%
AB_1	386.5	0.2	514.4	32.3	514.4	51.3	514.4	50.8	33.1%
AB_2	-393367.0	15.4	6940.8	TLR	6890.6	TLR	6943.5	TLR	
AZ_1	4375.4	0.5	4697.3	89.3	4697.3	127.1	4697.3	150.2	7.4%
AZ_2	48839.7	0.7	49609.3	84.3	49710.1	174.2	49722.6	120.0	1.8%
AZ_3	61923.4	0.1	61923.4	10.2	64219.4	17.6	61923.4	14.6	3.7%
BH_1	39791.9	4.3	39809.3	TLR	39791.9	TLR	39791.9	TLR	0.0%
BH_2	66641.2	8.7	67717.1	TLR	67700.3	TLR	67709.8	TLR	1.6%
BC_1	34587.6	65.2	34840.0	TLR	34587.6	TLR	34683.8	TLR	0.7%
CA_2	-781582000.0	1.6	-638609000.0	234.6	-611308000.0	224.4	-618112000.0	224.9	
CR_1	2169.6	3.1	2194.6	502.4	2209.8	252.6	2202.4	271.2	1.9%
CR_2	872.2	0.3	872.2	39.8	872.2	38.1	872.2	37.5	0.0%
CR_3	-224640.0	0.1	-224635.0	14.2	-224635.0	13.3	-224635.0	13.2	
CR_4	442.6	0.1	442.6	13.4	442.6	13.1	442.6	13.2	0.0%
CR_5	616.7	0.1	632.1	7.4	632.1	7.1	632.1	8.6	2.5%
CR_6	581.9	0.1	581.9	5.0	581.9	4.8	607.9	5.7	4.5%
CH_1	1183.4	35.1	1206.1	TLR	1201.9	TLR	1202.2	TLR	1.9%
CP_1	429.9	0.7	429.9	234.3	429.9	139.4	431.8	150.0	0.4%
CP_2	690.2	0.0	690.2	4.3	690.2	2.6	690.2	2.7	0.0%
CP_3	8864.9	1.4	9040.0	207.8	9048.3	181.2	8946.1	198.6	2.1%
CP_4	2129.4	1.9	2151.3	541.9	2157.7	264.7	2155.4	262.9	1.3%
DE_1	4937.6	38.2	4937.6	TLR	4937.6	TLR	4938.3	TLR	0.0%
FL_1	7309.2	0.1	9536.6	17.0	9533.2	19.0	9534.3	18.2	30.5%
LS_1	6370.1	0.3	6486.0	77.0	6484.1	72.6	6483.5	92.9	1.8%
LC_1	18895.0	0.7	18943.9	122.2	19019.4	122.5	18944.0	121.1	0.7%
LC_2	7095.4	1.3	7510.9	174.7	7380.2	164.6	7479.7	172.8	5.9%
LO_1	3901.6	50.8	3901.6	TLR	3901.6	TLR	3901.6	TLR	0.0%
LO_2	41.3	0.1	41.3	4.4	41.3	5.0	41.3	4.4	0.0%
OM_1	681.9	1.8	758.6	247.8	756.6	218.5	761.8	169.1	11.7%
OM_2	-9649740.0	1.7	-9024740.0	263.7	-9024740.0	262.2	-9024740.0	269.3	

OM_3	300.3	0.1	358.3	11.5	358.3	15.7	358.3	13.2	19.3%
PT_1	56112.2	0.9	56182.9	190.3	56182.9	186.7	56182.9	191.3	0.1%
SP_1	8610.5	0.6	8895.1	109.6	8921.9	130.5	8901.3	116.5	3.6%
SM_1	116251.0	15.9	118826.0	TLR	121297.0	TLR	118470.0	TLR	4.3%
SM_2	27086.1	0.3	27370.3	58.8	27313.0	71.6	27222.8	44.0	1.0%
SM_3	50851.1	0.5	50851.1	65.5	50851.1	111.6	50852.3	65.8	0.0%
SM_4	21600.8	0.2	22087.6	33.7	22093.0	53.9	22091.9	29.1	2.3%
SM_5	7237.8	0.3	8608.8	40.3	8482.2	78.2	8639.2	47.1	19.4%
SM_6	15263.9	0.1	16441.1	10.7	16683.8	28.4	16714.8	15.6	9.5%
VG_1	8927.1	0.9	8934.9	245.2	8937.2	409.8	8939.1	241.3	0.1%
VG_2	129369.0	0.3	130105.0	41.1	130105.0	57.7	129593.0	53.8	0.6%
VG_3	14900.3	0.1	15035.7	3.4	15035.7	6.2	15035.7	3.3	0.9%
VG_4	16012.7	0.2	16012.7	30.1	16018.6	47.1	16019.1	30.2	0.0%
VG_5	59496.3	1.5	60766.3	203.2	61434.6	339.5	60568.2	224.9	3.3%
VN_1	15595.0	0.9	15605.5	155.9	15605.5	237.8	15605.5	155.3	0.1%
VV_1	57695.9	5.3	57695.9	514.5	57695.9	TLR	57695.9	488.4	0.0%
VV_2	2978.2	1.9	2978.2	363.0	2978.2	513.9	2978.2	341.2	0.0%
TLR = Time limit (600 seconds) reached								**Average**	**5.0%**

Table 6.4: Comparison of the best obtained result with the results reported by Bianchi-Aguiar et al. (2018b)

Instance	**ILS (Best)**	**BAP-LR**		**BAP**		**H-BAP**		**H-BAP-IF**		**H-BAP-IE**	
		Z^{lr}	Diff (%)	Z	Diff (%)	Z	Diff (%)	Z	Diff (%)	Z	Diff (%)
AG_1	2406.6	2730.3	-11.9%	-142247.9		2558.7	-5.9%	2558.8	-5.9%	2351.8	2.3%
AG_2	2139.5	2539.9	-15.8%	2154.0	-0.7%	2132.8	0.3%	2132.8	0.3%	2132.8	0.3%
AG_3	27433.9	33598.4	-18.3%	*		32094.6	-14.5%	32087.0	-14.5%	31793.4	-13.7%
AG_4	4932.0	5762.3	-14.4%	-148264.8		5505.3	-10.4%	5505.3	-10.4%	5184.0	-4.9%
AI_1	3449.0	3938.1	-12.4%	3748.6	-8.0%	3760.7	-8.3%	3760.7	-8.3%	3686.5	-6.4%
AI_2	6492.2	7080.8	-8.3%	6474.0	0.3%	6805.2	-4.6%	6805.2	-4.6%	6465.4	0.4%
AI_3	2486.4	3234.4	-23.1%	2120.6	17.2%	2616.8	-5.0%	2616.8	-5.0%	2598.6	-4.3%
AI_4	19761.6	20267.6	-2.5%	19831.4	-0.4%	19805.5	-0.2%	19805.5	-0.2%	19235.3	2.7%
AB_1	514.4	538.5	-4.5%	500.7	2.7%	500.4	2.8%	500.4	2.8%	500.4	2.8%
AB_2	6943.5	7213.3	-3.7%	6222.8	11.6%	*		6944.2	0.0%	6953.4	-0.1%
AZ_1	4697.3	4934.3	-4.8%	4746.4	-1.0%	*		4743.1	-1.0%	4723.9	-0.6%
AZ_2	49722.6	52107.9	-4.6%	51107.5	-2.7%	50758.2	-2.0%	50758.2	-2.0%	50504.1	-1.5%
AZ_3	64219.4	70535.7	-9.0%	64203.2	0.0%	64203.2	0.0%	64203.2	0.0%	61907.8	3.7%
BH_1	39809.3	42876.6	-7.2%	40820.1	-2.5%	40807.2	-2.4%	40807.2	-2.4%	38246.0	4.1%
BH_2	67717.1	74591.3	-9.2%	*		-218294000.0		-218294000.0		73174.9	-7.5%
BC_1	34840.0	65517.9	-46.8%	*		51320.0	-32.1%	50416.3	-30.9%	*	
CA_2	-611308000.0	290.4		243.2		243.5		243.5		239.2	
CR_1	2209.8	3762.7	-41.3%	3124.6	-29.3%	3165.9	-30.2%	3165.9	-30.2%	3022.6	-26.9%

CR_2	872.2	1017.3	-14.3%	872.2	0.0%	872.2	0.0%	872.2	0.0%	872.2	0.0%
CR_3	-224635.0	641.0		460.6		460.6		460.6		460.6	
CR_4	442.6	571.6	-22.6%	442.6	0.0%	442.6	0.0%	442.6	0.0%	442.6	0.0%
CR_5	632.1	1162.6	-45.6%	632.1	0.0%	632.1	0.0%	632.1	0.0%	632.1	0.0%
CR_6	607.9	718.4	-15.4%	608.3	-0.1%	*		608.3	-0.1%	541.2	12.3%
CH_1	1206.1	1361.2	-11.4%	*		-393847.0		-393847.0		-53232.5	
CP_1	431.8	472.3	-8.6%	453.6	-4.8%	453.6	-4.8%	453.6	-4.8%	439.7	-1.8%
CP_2	690.2	782.8	-11.8%	690.2	0.0%	690.2	0.0%	690.2	0.0%	690.2	0.0%
CP_3	9048.3	10082.7	-10.3%	-355477.4		-18659500.0		-18659500.0		-705417.0	
CP_4	2157.7	3167.3	-31.9%	*		-47366100.0		-47366100.0		2638.0	-18.2%
DE_1	4938.3	6719.7	-26.5%	*		*		-27167400.0		-2249420.0	
FL_1	9536.6	10112.4	-5.7%	9761.0	-2.3%	9761.0	-2.3%	9761.0	-2.3%	9718.7	-1.9%
LS_1	6486.0	7393.2	-12.3%	7124.2	-9.0%	7124.2	-9.0%	7124.2	-9.0%	7118.7	-8.9%
LC_1	19019.4	21061.5	-9.7%	20281.9	-6.2%	20276.0	-6.2%	20276.0	-6.2%	20184.8	-5.8%
LC_2	7510.9	8410.7	-10.7%	7626.1	-1.5%	7696.7	-2.4%	7696.7	-2.4%	7519.8	-0.1%
LO_1	3901.6	4343.3	-10.2%	*		-11239700.0		-11239700.0		-481437.0	
LO_2	41.3	69.1	-40.2%	51.3	-19.5%	51.3	-19.5%	51.3	-19.5%	51.3	-19.5%
OM_1	761.8	938.3	-18.8%	-24368.6		889.8	-14.4%	889.8	-14.4%	795.7	-4.3%
OM_2	-9024740.0	299.5		259.1		259.1		259.1		257.8	
OM_3	358.3	519.4	-31.0%	358.3	0.0%	307.4	16.5%	307.4	16.5%	300.7	19.1%
PT_1	56182.9	90074.2	-37.6%	62034.9	-9.4%	*		62035.0	-9.4%	56182.9	0.0%
SP_1	8921.9	9719.7	-8.2%	9332.0	-4.4%	9321.3	-4.3%	9321.3	-4.3%	9216.7	-3.2%
SM_1	121297.0	135908.5	-10.8%	*		131061.0	-7.4%	131207.0	-7.6%	129632.0	-6.4%
SM_2	27370.3	28394.2	-3.6%	27804.1	-1.6%	27672.1	-1.1%	27672.1	-1.1%	27763.7	-1.4%
SM_3	50852.3	57901.9	-12.2%	53382.4	-4.7%	54013.8	-5.9%	54013.7	-5.9%	52799.9	-3.7%
SM_4	22093.0	22926.9	-3.6%	22303.1	-0.9%	22156.8	-0.3%	22156.8	-0.3%	22098.1	0.0%
SM_5	8639.2	10734.4	-19.5%	10246.5	-15.7%	10002.6	-13.6%	10002.6	-13.6%	10130.1	-14.7%
SM_6	16714.8	17646.6	-5.3%	16784.5	-0.4%	16469.9	1.5%	16469.9	1.5%	16431.2	1.7%
VG_1	8939.1	11323.1	-21.1%	10421.8	-14.2%	10411.8	-14.1%	10411.8	-14.1%	10399.4	-14.0%
VG_2	130105.0	140699.9	-7.5%	135736.8	-4.1%	135737.0	-4.1%	135737.0	-4.1%	135265.0	-3.8%
VG_3	15035.7	18901.9	-20.5%	17431.5	-13.7%	17409.1	-13.6%	17409.1	-13.6%	16878.1	-10.9%
VG_4	16019.1	18006.0	-11.0%	16040.5	-0.1%	16040.6	-0.1%	16040.6	-0.1%	16000.1	0.1%
VG_5	61434.6	69752.5	-11.9%	68151.0	-9.9%	67996.4	-9.7%	67996.4	-9.7%	66622.1	-7.8%
VN_1	15605.5	18269.9	-14.6%	15605.5	0.0%	*		15596.8	0.1%	15481.3	0.8%
VV_1	57695.9	61016.7	-5.4%	*		-69175900.0		-69175900.0		60209.9	-4.2%
VV_2	2978.2	3252.9	-8.4%	*		3204.8	-7.1%	3204.9	-7.1%	3230.3	-7.8%
Average Diff (%)		**-15.2%**		**-3.7%**		**-6.0%**		**-5.5%**		**-3.3%**	

* No feasible solution was found
We assumed the results of AI_2 and AI_3 to be switched in Bianchi-Aguiar et al. (2018b)

6.5 Visualization of planograms

The present work would have no practical significance if we could not translate the final output of the heuristic into an actual planogram, which is the main tool that supports retailers in their in-store replenishment processes. To facilitate the task of converting the heuristic's outcome into a planogram, an intuitive solution representation was used. For each shelf level, we considered a vector in which each element is a pair (i.e. 2-tuple) that contains a product and its continuous horizontal (left) location on that shelf level. Such a representation can easily be converted into a planogram, as depicted in Figure 6.5.

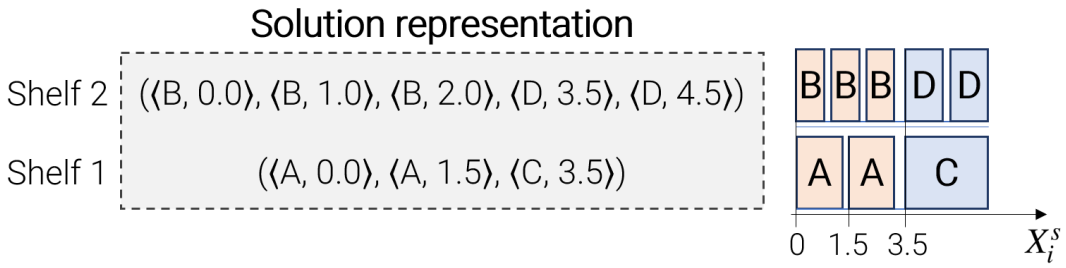
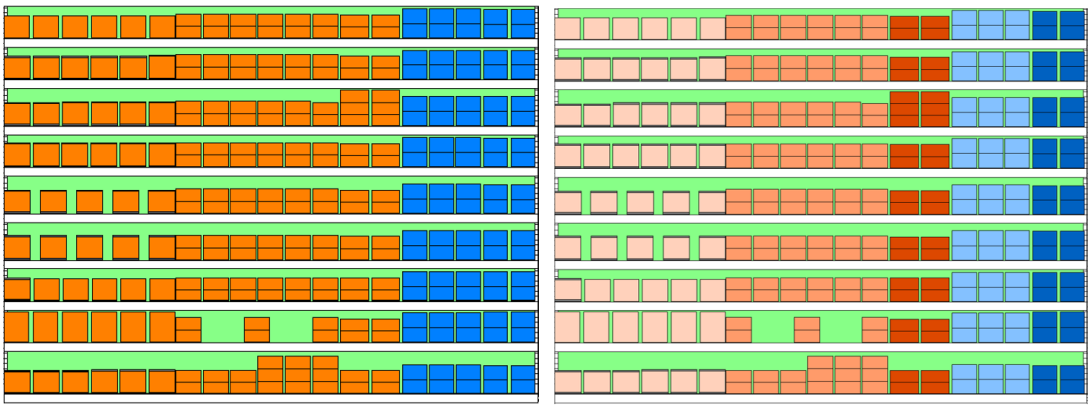


Figure 6.5: Solution representation and corresponding planogram

A script converts the obtained solution into a format that can be read by a planogram visualization tool and outputs the corresponding file. In our case, we output a .psa file, which can be opened using most of the space planning tools that are commonly used in retail. Figure 6.6 provides an example of a planogram that was obtained using the heuristic procedure for instance AI_1. In this instance, both first and second level blocks are vertically oriented. The planogram is also useful in the sense that it allows us to validate the procedure and the generated solutions.



(a) Highlight: first level blocks (b) Highlight: second level blocks

Figure 6.6: Planogram corresponding to instance AI_1

Chapter 7

Conclusion

This chapter presents the concluding remarks on the work that was developed and presented in the preceding chapters. In Section 7.1, the obtained results are discussed and the corresponding findings are summarized. Then, Section 7.2 clarifies the relevance of this dissertation with respect to real-world retail environments. Finally, in Section 7.3 some limitations of the present approach are indicated and suggestions for future work are made accordingly.

7.1 Discussion and main findings

The present dissertation proposed a heuristic approach for solving the Shelf Space Allocation Problem under merchandising rules, which can be divided into a constructive and an improvement stage. The constructive procedure starts by exploring the problem's inherent hierarchical nature, which is a consequence of the established set of merchandising rules, and assigns a particular region of the planogram to each product family. With product families allocated, the final placement of products on the shelves is determined by a dynamic programming method that solves the Knapsack Problems that arise in the context of the considered formulation of the SSAP. In the improvement stage, the Iterated Local Search metaheuristic is applied: local search techniques are combined with perturbation mechanisms in an iterative process, with the goal of improving the initially obtained solution.

In order to assess the effectiveness of the proposed approach, a series of computational tests were performed. For that purpose, a set of benchmark instances (Bianchi-Aguiar et al., 2014) was used. The constructive heuristic yielded feasible solutions within very short computation times in the vast majority of the cases. However, as the instance sizes increase (especially with regard to the complexity of the structure of blocks and the number of products to be allocated), the computational burden starts to become more noticeable. Given that the ILS makes use of the same algorithms that were designed for the constructive phase, these conclusions also hold for the improvement stage.

Feasible solutions were obtained for a total of 50 out of 54 instances using the constructive heuristic alone, with an additional feasible solution being achieved after applying the ILS method,

which represents an improvement when compared with Bianchi-Aguiar et al. (2018b). This attests the versatility of our approach, since the tests were conducted on a wide range of instances with very different characteristics (to put it in perspective, the total number of products N ranged from 7 in VG_3 to 240 in DE_1). Furthermore, the heuristic procedure also compares well with state-of-the-art approaches (Bianchi-Aguiar et al., 2018b) in terms of solution quality. An average gap of 15.2% with respect to the linear relaxation model was obtained. Additionally, when compared with the H-BAP-IE model (which is the one that leads to shorter computation times, thus allowing for a fair comparison), we get an average difference of just 3.3%.

7.2 Implications for practice

The outcome of this thesis is clearly of practical significance. To begin with, it is important to emphasize that the presented algorithms were tested on real instances from a European grocery retailer. Moreover, our approach is realistic, as it incorporates constraints regarding the retailer's merchandising preferences, which substantially increases the difficulty of the problem. These constraints are not included in the vast majority of the optimization approaches to shelf space allocation that can be found in the literature.

The developed approach could be interesting for retailers for a variety of reasons, the first of which being computational efficiency, particularly for smaller instances. Additionally, and as mentioned before, the constructive procedure is able to output solutions of acceptable quality on its own. This could be useful in contexts in which retailers are skeptical of fully automating the planogram generation procedure. In those situations, the constructive procedure could expeditiously provide a base planogram (that followed all of the established merchandising rules) on top of which small manual adjustments would be performed. Bianchi-Aguiar et al. (2016) described the case of a retail company in which planograms were manually designed in a time-consuming process (an average of three hours were spent in a single planogram) due to the necessity of complying with a complex set of merchandising rules. The method proposed in this dissertation would substantially increase the efficiency of the processes associated with the creation of planograms, thus allowing managers to dedicate more of their time to other important activities within space management.

An additional advantage of the presented method is the fact that it makes it possible to interrupt the procedure at any point in time and obtain a feasible solution (provided that an initial feasible solution is available), which can be convenient in some situations. This may not be possible with a mathematical programming-based approach.

At this point, one might argue that the reported computational efficiency is attained at the expense of the quality of the generated solutions. It is possible, however, to counter such a perspective. First, in the majority of the cases, the loss of quality is negligible or even non-existent. In addition, it is worth noting that a solution with a higher objective function value does not necessarily translate into a more effective planogram (i.e. a planogram which leads to an increase in consumer demand), given the subjectiveness which is inherent to the chosen objective function.

Hence, the most crucial attribute of an automated planogram generation tool would be its ability to efficiently generate realistic planograms. Besides, when the retailer's merchandising rules are complex and therefore lead to a high number of constraints, there is not much room left for optimization anyway.

Finally, a significant advantage of an heuristic approach is the fact that it does not require the use of optimization software, which would constitute an additional cost to the retailer.

7.3 Future work

The main limitation of the procedure that was described in this dissertation is the fact that it tends to scale poorly as the instance sizes increase. This limitation could be tackled by dividing those instances into smaller ones and solving each of these smaller instances independently. This is exactly the case with the SM instances, where instances SM_2 to SM_6 correspond to sub-instances of the SM_1 instance. While generating an initial solution for SM_1 takes around 16 seconds, instances SM_2 to SM_6 are solved in less than 2 seconds (all computation times combined). An heuristic could be developed to divide the instances according to a given criteria.

Future works could also introduce an alternative objective function for the SSAP, less prone to subjective considerations. This could be achieved by adopting a cost minimization perspective, considering inventory holding and in-store replenishment costs. Moreover, shelf space management could also be enhanced by integrating other retail decisions in the SSAP, such as assortment selection, inventory policies, and macro space planning.

Regarding the merchandising rules, a natural extension of this work would be to consider the possibility of merging smaller blocks. Since these rules specify that blocks must form rectangular shapes on the shelves, aggregating blocks would reduce the number of situations in which an excessive number of facings is attributed to a particular product so that the corresponding block conforms to a rectangular shape.

Lastly, our work was restricted to the allocation of products to shelves. However, other types of equipment for product display (e.g. pegboards, tables, and bins) are very common in practice. Accordingly, the present work could be further extended in that direction.

Bibliography

Bai, R. (2005). *An investigation of novel approaches for optimising retail shelf space allocation*. PhD thesis, University of Nottingham.

Bergstra, J. and Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(1):281–305.

Bianchi-Aguiar, T. (2015). *The Retail Shelf Space Allocation Problem: New Optimization Methods Applied to a Supermarket Chain*. PhD thesis, Faculdade de Engenharia da Universidade do Porto.

Bianchi-Aguiar, T., Carravilla, M. A., and Oliveira, J. F. (2015). Replicating shelf space allocation solutions across retail stores. Working paper.

Bianchi-Aguiar, T., Hübner, A. H., Carravilla, M. A., and Oliveira, J. F. (2018a). Retail Shelf Space Planning Problems: A Comprehensive Review and Classification Framework. Working paper.

Bianchi-Aguiar, T., Silva, E., Guimarães, L., Carravilla, M. A., and Oliveira, J. F. (2014). Problem instances for the shelf space allocation problem with family grouping. `https://fe.up.pt/~mtbaguiar/SSAP/`.

Bianchi-Aguiar, T., Silva, E., Guimarães, L., Carravilla, M. A., and Oliveira, J. F. (2018b). Allocating products on shelves under merchandising rules: Multi-level product families with display directions. *Omega*, 76:47–62.

Bianchi-Aguiar, T., Silva, E., Guimarães, L., Carravilla, M. A., Oliveira, J. F., Amaral, J. G., Liz, J., and Lapela, S. (2016). Using Analytics to Enhance a Food Retailer's Shelf-Space Management. *Interfaces*, 46(5):424–444.

Borin, N., Farris, P. W., and Freeland, J. R. (1994). A Model for Determining Retail Product Category Assortment and Shelf Space Allocation. *Decision Sciences*, 25(3):359–384.

Buttle, F. (1984). Retail Space Allocation. *International Journal of Physical Distribution & Materials Management*, 14(4):3–23.

Corstjens, M. and Doyle, P. (1981). A Model for Optimizing Retail Space Allocations. *Management Science*, 27(7):822–833.

Curhan, R. C. (1972). The Relationship between Shelf Space and Unit Sales in Supermarkets. *Journal of Marketing Research*, 9(4):406–412.

Desmetab, P. and Renaudinb, V. (1998). Estimation of product category sales responsiveness to allocated shelf space. *International Journal of Research in Marketing*, 15(5):443–457.

Drèze, X., Hoch, S. J., and Purk, M. E. (1994). Shelf management and space elasticity. *Journal of Retailing*, 70(4):301–326.

Ebster, C. and Garaus, M. (2011). *Store Design and Visual Merchandising: Creating Store Space That Encourages Buying*. Business Expert Press, 2nd edition.

Eisend, M. (2014). Shelf space elasticity: A meta-analysis. *Journal of Retailing*, 90(2):168–181.

eMarketer (2018). Total retail sales worldwide from 2015 to 2020 (in trillion U.S. dollars). In *Statista - The Statistics Portal*. Retrieved May 13, 2018, from `https://www.statista.com/statistics/443522/global-retail-sales/`.

Ensor, P. S. (1988). The Functional Silo Syndrome. AME Target (p. 16). Rolling Meadows, IL: Association for Manufacturing Excellence.

Fisher, M. (2009). OR FORUM-Rocket Science Retailing: The 2006 Philip McCord Morse Lecture. *Operations Research*, 57(3):527–540.

Geismar, H. N., Dawande, M., Murthi, B. P. S., and Sriskandarajah, C. (2015). Maximizing Revenue Through Two-Dimensional Shelf-Space Allocation. *Production and Operations Management*, 24(7):1148–1163.

Hansen, J. M., Raut, S., and Swami, S. (2010). Retail Shelf Allocation: A Comparative Analysis of Heuristic and Meta-Heuristic Approaches. *Journal of Retailing*, 86(1):94–105.

Hwang, M., Choi, B., and Lee, G. (2009). A genetic algorithm approach to an integrated problem of shelf space design and item allocation. *Computers & Industrial Engineering*, 56(3):809–820.

Hübner, A. H. and Kuhn, H. (2012). 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.

Hübner, A. H., Kuhn, H., and Sternbeck, M. G. (2013). Demand and supply chain planning in grocery retail: an operations planning framework. *International Journal of Retail & Distribution Management*, 41(7):512–530.

Hübner, A. H. and Schaal, K. (2017a). A shelf-space optimization model when demand is stochastic and space-elastic. *Omega*, 68:139–154.

Hübner, A. H. and Schaal, K. (2017b). An integrated assortment and shelf-space optimization model with demand substitution and space-elasticity effects. *European Journal of Operational Research*, 261(1):302–316.

Hübner, A. H. and Schaal, K. (2017c). Effect of replenishment and backroom on retail shelf-space planning. *Business Research*, 10(1):123–156.

Irion, J., Lu, J.-C., Al-Khayyal, F., and Tsao, Y.-C. (2012). A piecewise linearization framework for retail shelf space management models. *European Journal of Operational Research*, 222(1):122–136.

Janeiro, A. C. (2014). Optimization algorithms for the shelf space allocation problem. Master's thesis, Faculdade de Engenharia da Universidade do Porto.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer-Verlag Berlin Heidelberg, 1st edition.

Koschat, M. A. (2008). Store inventory *can* affect demand: Empirical evidence from magazine retailing. *Journal of Retailing*, 84(2):165–179.

Kurtuluş, M. and Nakkas, A. (2011). Retail Assortment Planning Under Category Captainship. *Manufacturing & Service Operations Management*, 13(1):124–142.

Larson, P. D. and DeMarais, R. A. (1990). Psychic Stock: An Independent Variable Category of Inventory. *International Journal of Physical Distribution & Logistics Management*, 20(7):28–34.

Lim, A., Rodrigues, B., and Zhang, X. (2004). Metaheuristics with Local Search Techniques for Retail Shelf-Space Optimization. *Management Science*, 50(1):117–131.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2010). Iterated local search: Framework and applications. In Potvin, J.-Y. and Gendreau, M., editors, *Handbook of Metaheuristics*, chapter 11. Springer, Boston, MA.

Murray, C. C., Talukdar, D., and Gosavi, A. (2010). Joint Optimization of Product Price, Display Orientation and Shelf-Space Allocation in Retail Category Management. *Journal of Retailing*, 86(2):125–136.

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

Schaal, K. and Hübner, A. (2017). When does cross-space elasticity matter in shelf-space planning? A decision analytics approach. *Omega*.

Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European Journal of Operational Research*, 174(3):1519–1539.

Tongsari, K. (1995). Optimization of shelf space allocation in three dimensions. Master's thesis, The Faculty of the Russ College of Engineering and Technology, Ohio University.

van Nierop, E., Fok, D., and Franses, P. H. (2008). Interaction Between Shelf Layout and Marketing Effectiveness and Its Impact on Optimizing Shelf Arrangements. *Marketing Science*, 27(6):1065–1082.

Varley, R. (2006). *Retail Product Management: Buying and Merchandising*. Routledge, 2nd edition.

Within, T. M. (1957). *Theory of Inventory Management*. Princeton University Press. Princeton, NJ.

Wolfe, H. B. (1968). A Model for Control of Style Merchandise. *Industrial Management Review (pre-1986)*, 9(2):69–82.

Yang, M.-H. (2001). An efficient algorithm to allocate shelf space. *European Journal of Operational Research*, 131(1):107–118.

Yang, M.-H. and Chen, W.-C. (1999). A study on shelf space allocation and management. *International Journal of Production Economics*, 60–61:309–317.

Zufryden, F. S. (1986). A Dynamic Programming Approach for Product Selection and Supermarket Shelf-Space Allocation. *Journal of the Operational Research Society*, 37(4):413–422.

Appendix A

Notation of the SSAP formulation of Bianchi-Aguiar et al. (2018b)

Sets

$\mathcal{K}$	Shelves
$\mathcal{N}$	Products
$\mathcal{M}$	Blocks (product families)
$\mathcal{N}_u$	Products belonging to block u
$\mathcal{M}_u$	Child blocks of block u
$\mathcal{S}^H$	Blocks that should have their child blocks aligned horizontally
$\mathcal{S}^V$	Blocks that should have their child blocks aligned vertically
$\mathcal{V}_H$	Downstream blocks belonging to u

Indices

k	Shelf index
i, j	Product index
u, m	Block (product family) index

Decision Variables

W_{ik}	The integer number of facings of product $i \in \mathcal{N}$ on shelf $k \in \mathcal{K}$
X_i^s	The continuous horizontal location of product $i \in \mathcal{N}$, measured from the lower-left corner of the planogram to the lower-left corner of the first facing of the product
Y_{mk}	= 1 if block $m \in \mathcal{M} \cup \mathcal{N}$ is located on shelf $k \in \mathcal{K}$
T_{mnk}	= 1 if block m is displayed immediately after block n on shelf $k \in \mathcal{K}$, $u \in \mathcal{M}, m, n \in \mathcal{V}_u \cup \{0\}$
F_{mnk}	the continuous flow from block m to block n on shelf $k \in \mathcal{K}$, $u \in \mathcal{M}$, $m, n \in \mathcal{V}_u \cup \{0\}$,
L_{ik}	Shelf length assigned to product $i \in \mathcal{N}$ on shelf $k \in \mathcal{K}$
X_m^s	The horizontal location of block $m \in \mathcal{M} \cup \mathcal{N}$ (left coordinate)
X_m^e	The horizontal location of block $m \in \mathcal{M} \cup \mathcal{N}$ (right coordinate)
FL_{mk}	= 1 if $k \in \mathcal{K}$ is the first shelf of block $m \in \mathcal{M} \cup \mathcal{N}$
LL_{mk}	= 1 if $k \in \mathcal{K}$ is the last shelf of block $m \in \mathcal{M} \cup \mathcal{N}$

Parameters

K	Total number of shelves
M	Total number of blocks (product families)
N	Total number of products
w_k	Width of shelf k
h_k	Height of shelf k
a_i	Width of product i
b_i	Height of product i
p_i	Profit per facing of product i
l_i	Minimum number of facings of product i
u_i	Maximum number of facings of product i
γ_k	Effectiveness of shelf k to generate revenue (shelf attractiveness)
v	Maximum deviation of blocks between shelves
w_m^{max}	Width of the largest product from each block m

Appendix B

SSAP Heuristic: additional notation

Constructive procedure

Variables

TF_i	Target facings of product i
TF'_i	Difference between the target facings of product i (TF_i) and the minimum number of facings of product i (l_i)
L_u	Width of block u
K_u	Number of shelves on which block u is present
τ_u	Tightness of block u
α_u	Proportion of the width of the parent block of block u that is attributed to u
α_u^{min}	Minimum proportion of the width of the parent block of block u that should be attributed to u
α_u^{target}	Target proportion of the width of the parent block of block u that should be attributed to u
K_u^{EXP}	Expected number of shelf levels to attribute to block u
L_u^{min}	Minimum necessary width to assign to vertical block u
PS_{ik}	= 1 if product i is on shelf k
$s_j^{pk}(d)$	Optimal solution value of problem $KP_j^{pk}(d)$
$F_j(d)$	= 1 if product j is added to the knapsack in iteration j of the corresponding Knapsack Problem

Sets

$\mathcal{K}_u$	Shelves on which block u is present
$\mathcal{G}_c$	Blocks that belong to hierarchical level c
$\mathcal{R}_k$	Reference blocks on shelf k

Parameters and indices

c	Index of a block level
C	Number of hierarchical levels of blocks
τ_u^{max}	Maximum tightness of block u
$KP_j^{pk}(d)$	Knapsack Problem associated with reference block p in shelf level k with total capacity d
A	Integer value which is sufficiently large so as to yield an integer result when multiplied by either a product's profit or width

Iterated Local Search

Variables

$s^{*\prime}$	Newly obtained solution
s^{*}	Incumbent solution
s^{best}	Best solution found in the ILS procedure
β	Perturbation strength

Parameters and indices

T	Temperature (LSMC acceptance criterion)
T_0	Initial temperature (LSMC acceptance criterion)
ρ	Common ratio of the geometric cooling schedule (LSMC acceptance criterion)
n	Index of an iteration of the ILS procedure
X	Random variable following a uniform distribution over [0, 1]

Appendix C

Horizontal allocation algorithm: practical example

In order to further clarify the algorithm for allocating blocks horizontally, which was presented in 5.1.2.2, a practical example will now be presented. For this purpose, we consider a situation in which four blocks have to be allocated onto four shelf levels, as depicted in Figure C.1. The blocks' adjusted width proportions (α_u) are presented in Table C.1. Note that the blocks are already sorted according to their widths proportions (by ascending order).

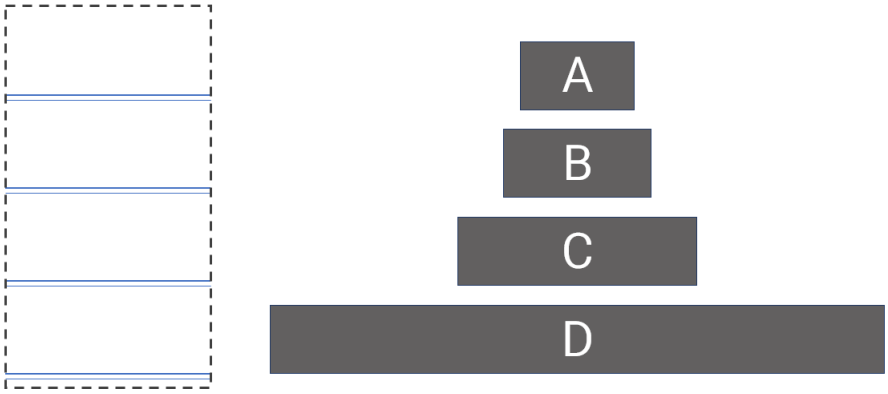


Figure C.1: Shelves and blocks considered for the horizontal allocation example

Table C.1: Width proportions of the blocks considered for the horizontal allocation example

Block	Width proportion
A	0.10
B	0.13
C	0.21
D	0.56

We can now begin the *while* loops of Algorithm 4. By allocating block A to the first shelf level, we get $K^{cumulative} = 0.10 \times 4 = 0.40$. Given that $[K^{cumulative}] \leq 1$, we can try to see if it is possible to add an additional block to that shelf level. Thus, we enter the inner *while* loop (lines 12–21). After updating $K^{cumulative}$, which is now equal to $0.40 + 0.13 \times 4 = 0.92$, we conclude that the condition $[K^{cumulative}] \leq 1$ still holds: hence, block B is added to the first shelf level (lines

14–16). Updating $K^{cumulative}$ again, we get $K^{cumulative} = 0.92 + 0.21 \times 4 = 1.76$ and therefore $[K^{cumulative}] > 1$. No more blocks will be added to the first shelf level (lines 17–19).

Then, we reinitialize $K^{cumulative}$ as being equal to $0.21 \times 4 = 0.84$ (block C). By adding block D, $K^{cumulative} = 0.84 + 0.56 \times 4 = 3.08$ and thus $[K^{cumulative}] > 1$. As a result, only block C will be present on the second shelf level.

Given that, for block D, $\alpha_D \times 4 = 2.24 (> 1)$, we exit the outer *while* loop of Algorithm 4 and complete the first stage of the procedure for assigning horizontal blocks to shelves. Figure C.2 summarily depicts the procedure that was just described.

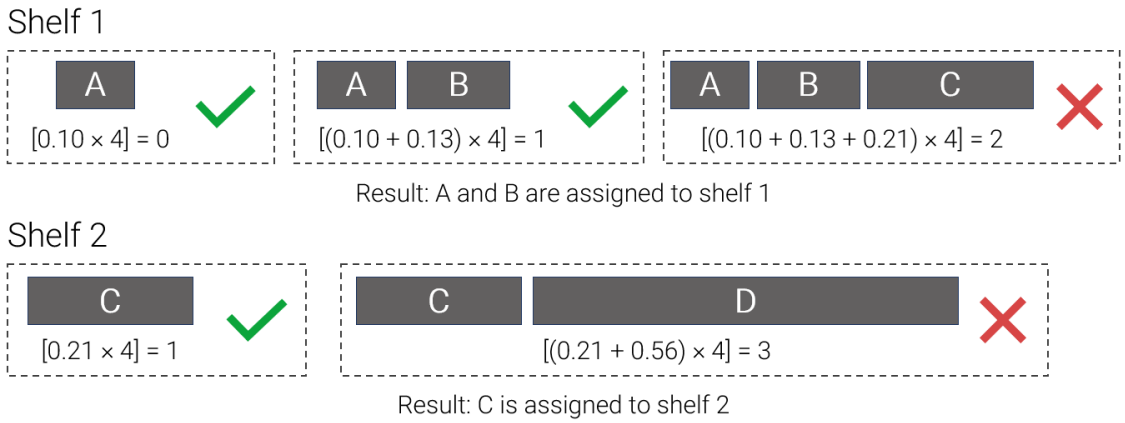


Figure C.2: Example of the first stage of the procedure for assigning horizontal blocks to shelves

Given that there are still two shelf levels available (1 and 2) and that one block has not yet been allocated (D), this situation falls into *Situation #5* of Table 5.1. In this case, the assignment is trivial: block D will be present on the shelves 3 and 4. The final result of the algorithm is as summarized in Table C.2.

Table C.2: Final assignment of blocks to shelves

Shelf	Block(s)
1	A, B
2	C
3	D
4	D

Appendix D

Problem instances

Table D.1, which was retrieved from Bianchi-Aguiar et al. (2018b), summarizes important information regarding each of the considered instances, namely: number of products (N), number of blocks (M), number of shelves (K), number of hierarchical levels (L), and number of vertical (MV) and horizontal blocks (MH).

Table D.1: Problem Instances (Bianchi-Aguiar et al., 2018b)

Name	N	M	K	L	MV	MH	Name	N	M	K	L	MV	MH	Name	N	M	K	L	MV	MH
FL_1	16	16	6	4	5	10	**CR_2**	32	13	7	3	0	12	**SM_3**	49	10	6	3	7	2
AZ_3	10	11	5	3	8	2	**CR_1**	82	44	5	4	4	39	**SM_1**	171	47	6	4	36	10
AZ_2	25	12	5	3	9	2	**PT_1**	38	22	6	5	0	21	**CA_2**	77	34	6	5	2	31
AZ_1	32	27	5	3	0	26	**AI_1**	37	22	7	4	5	16	**AG_2**	19	19	7	5	0	18
LS_1	26	5	8	2	0	4	**AI_2**	41	27	9	5	4	22	**AG_1**	39	6	7	2	5	0
VG_3	7	4	7	2	0	3	**AI_4**	47	24	9	4	7	16	**AG_4**	85	32	5	4	10	21
VG_2	19	18	6	4	2	15	**AI_2**	47	17	7	3	3	13	**AG_3**	113	24	8	4	17	6
VG_4	28	15	6	3	2	12	**VN_1**	45	32	6	3	0	31	**AB_1**	28	14	8	3	0	13
VG_5	42	17	6	4	10	6	**SP_1**	49	15	7	4	6	8	**AB_2**	160	42	8	4	0	41
VG_1	60	29	8	4	0	28	**OM_3**	22	16	5	4	2	13	**VV_2**	84	3	5	2	2	0
CP_2	8	11	6	3	0	10	**OM_1**	54	41	5	4	6	34	**VV_1**	121	9	5	2	8	0
CP_1	24	18	5	3	5	12	**OM_2**	78	17	6	3	0	16	**LO_2**	15	4	5	2	0	3
CP_4	47	29	8	3	28	0	**LC_1**	46	21	6	4	4	16	**LO_1**	206	128	7	3	122	5
CP_3	51	27	6	5	21	5	**LC_2**	59	25	6	3	4	20	**BH_1**	108	25	7	3	0	24
CR_6	16	8	5	2	0	7	**SM_6**	19	7	6	3	4	2	**BH_2**	131	26	5	4	17	8
CR_5	19	10	5	3	0	9	**SM_2**	31	8	6	3	5	2	**CH_1**	190	99	6	4	15	83
CR_4	22	14	5	3	0	13	**SM_4**	34	11	6	3	8	2	**BC_1**	239	121	6	5	10	110
CR_3	25	11	5	3	0	10	**SM_5**	38	10	6	3	7	2	**DE_1**	240	45	8	4	40	4

N – Number of Products, M – Number of Blocks (excluding last level blocks), K – Number of Shelves,
L – Number of Hierarchical Levels, MV – Number of Vertical Blocks, MH – Number of Horizontal Blocks