Vol. 36, No. 3 (2026) | DOI: 10.37190/ord/218947

OPEN ACCESS

Operations Research and Decisions

www.ord.pwr.edu.pl

OPERATIONS RESEARCH AND DECISIONS QUARTERLY

Flower-cutting heuristics for shelf space allocation problem with simultaneous vertical and horizontal product categorization on the retail store shelves

Kateryna Czerniachowska[1]

[1]*Department of Process Management, Wroclaw University of Economics and Business, Wroclaw, Poland*

[*]*Corresponding author, email address: Kateryna.Czerniachowska@ue.wroc.pl*

Abstract

We address the retail shelf space allocation problem, which incorporates product simultaneous categorization across horizontal and vertical shelves. The problem also includes products with varying storage conditions and limited compatibility to be sold nearby. Our mathematical model focuses on optimizing the visibility of the products on the shelves in the retail store. We introduce a novel flower-cutting heuristic, which employs two variations of internal sorting rules. Seventeen parameters were used to narrow the solution space and generate efficient solutions without relying on randomization or exhaustive searches. When compared to optimal solutions obtained from a commercial solver, our approach yields solutions with 98.58%-100% accuracy. Optimal solutions were achieved for 17 out of 25 instances using both heuristics. The developed heuristics offer an approach that focuses on optimizing resource allocation problems.

Keywords: *heuristics, shelf space allocation, retail store, knapsack problem, decision making/process*

1. Introduction

In retailing, resources are essential elements that support the smooth operation and success of a retail business. Key resources include physical space, such as store layouts and shelf space, which directly influence product visibility and customer experience. Human resources, such as trained sales staff, managers, and customer service teams, are crucial for delivering quality service and maintaining customer satisfaction. Technology resources, like point-of-sale systems, inventory management tools, and e-commerce platforms, enable retailers to streamline operations, track sales, and engage with customers both in-store and online. Finally, financial resources, such as capital for purchasing inventory and marketing campaigns, help drive growth and competitiveness in the retail market.

In retailing, several optimization problems can be tackled using available resources to enhance efficiency, profitability, and customer satisfaction. Some of these problems include:

Received 27 July 2025, accepted 6 March 2026, published online 7 March 2026
ISSN 2391-6060 (Online)/

- Shelf space allocation: Optimizing the use of limited shelf space to maximize sales and improve the customer shopping experience. Retailers can use data analytics to determine the best arrangement of products based on demand, sales velocity, and customer preferences.
- Inventory management: Ensuring the right products are stocked at the right time and in the right quantities to meet demand without overstocking or understocking. Techniques like demand forecasting, reorder point optimization, and safety stock calculations can be used to improve inventory turnover and reduce holding costs.
- Product assortment optimization: Selecting the optimal mix of products to offer based on customer demand, seasonal trends, and local preferences. This involves analyzing customer purchasing patterns, market trends, and competitor offerings to adjust the product assortment and maximize profitability.
- Staff scheduling: Optimizing labour resources by scheduling employees based on expected foot traffic, sales volume, and peak hours. This helps retailers ensure they have enough staff available without overstaffing, which can reduce labour costs while maintaining excellent customer service.
- Pricing strategy: Setting dynamic pricing strategies that maximize revenue based on factors like demand, competition, and inventory levels. Optimization techniques such as price elasticity modelling and testing can help retailers determine the best price points for different products.
- Supply chain optimization: Improving the efficiency of the supply chain by determining the best distribution strategies, reducing lead times, and optimizing order quantities. Retailers can use resource allocation models and transportation optimization algorithms to lower costs and improve delivery timelines.
- Promotional campaigns: Optimizing marketing and promotional resources to maximize their impact. Retailers can analyze past campaign performance, customer response, and seasonal trends to design targeted promotions and discounts that drive sales and increase customer engagement.

By applying optimization techniques, retailers can make better use of their resources and improve profitability, efficiency, and customer satisfaction. Optimization techniques have proven to be highly effective in addressing these types of problems. By employing mathematical models and algorithms, optimization techniques enable decision-makers to find the most efficient solutions to complex problems. These techniques have been instrumental in a wide range of applications, from resource allocation and scheduling to logistics and supply chain management.

This study explores the challenge of shelf space allocation in retail stores, aiming to ensure that the correct quantity of the right product is in the right place at the right time. The goal is to maximize profit or product movement by allocating products to shelves. Tailored retail solutions for retail chain supermarkets can greatly influence the ease of product visibility and improve the overall efficiency of retail operations.

In this study, we introduce and apply two variations of the flower-cutting heuristic method to tackle the shelf space allocation problem on retail shelves with both vertical and horizontal product categories. These heuristics utilize algorithmic reasoning to methodically identify the necessary steps for achieving

effective solutions. Steering parameters are also implemented to ensure the quality of the obtained solution. While the approaches may seem intricate in their execution, they are conceptually simple at their core.

In the context of the problem under investigation, the parameters related to flower height can be analogously interpreted as parameters governing profitability within the problem domain. Similarly, the gardener's length of movement corresponds to the widths of product categories to which individual products are assigned. The interval between cut flowers allows for cutting not all flowers in the selected area as it would take too much time, but it helps to decrease the time spent investigating the selected area and helps to select the flower among similar ones. This corresponds to the method of decreasing the time needed to generate and check similar solutions. Meanwhile, the size of the basket represents the number of solutions that will be generated and compared based on the specific characteristics of the computer system under consideration.

It's important to note that there may be multiple areas within the garden where flowers can be cut, each with varying lengths. Moreover, the desired flower height, starting from the fact that the flowers are supposed to be of enough quality to be cut, may also vary in different areas. Our objective is to identify the biggest flower without necessitating cutting every flower in the garden. Therefore, we propose methodologies for systematically investigating the garden to selectively target areas with the biggest flowers.

Instead of indiscriminately cutting all areas of the garden, our approach focuses on strategic investigation methods to pinpoint areas where the flowers are the biggest. By employing techniques such as selective sampling or prioritized scanning based on predetermined criteria, we aim to efficiently identify and target those areas that are most likely to contain the biggest flower in the garden.

Furthermore, we recognize that the distribution of flower heights across the garden is not uniform. Certain regions may exhibit higher flower growth due to factors such as sunlight exposure or soil conditions. Therefore, our investigation methods incorporate considerations of spatial variability to ensure thorough coverage of the garden while prioritizing areas with the greatest potential for high flowers.

Through the implementation of these selective investigation strategies, we aim to streamline the process of identifying the biggest flower while minimizing unnecessary flower cutting and maximizing the gardener's efficiency. This targeted approach not only saves time and resources but also enhances the likelihood of successfully achieving the overarching goal of identifying the biggest flower in the garden.

The paper is organized as follows. Section 2 provides the theoretical foundation for the conceptual model. In Section 3, we provide the problem definition and formulation. In Section 4, we describe flower-cutting heuristics with two variations tailored to address the specific problem at hand. These algorithms are designed to navigate the complexities of shelf space distribution, ensuring optimal product arrangement across predefined categories. After outlining the algorithms, Section 5 presents a computational comparison of the two methods, assessing their performance in terms of efficiency and effectiveness against a commercial CPLEX solver. This comparison aims to offer insights into their relative strengths and practical use in solving the shelf space allocation problem. Through this analysis, we seek to identify the advantages and limitations of each heuristic, offering guidance for future improvements and applications in similar optimization scenarios. The paper concludes in Section 6, which also suggests directions for future research.

2. Related literature

Retail shelf space allocation has been extensively studied over the last few decades, focusing on optimizing product placement to increase profit, enhance visibility, and improve operational efficiency. Prior research demonstrates that solutions range from exact mathematical programming approaches to heuristic and metaheuristic methods, depending on the complexity of the problem and the size of the retail environment.

The earliest approaches to the Shelf Space Allocation Problem (SSAP) employed mathematical programming techniques, particularly linear and mixed-integer programming. These studies often assume simplified scenarios, such as a single vertical categorization or unconstrained shelf adjacency, enabling the derivation of optimal solutions using solvers like CPLEX. While these methods provide benchmarks for optimality, their applicability is limited in real-world settings due to the combinatorial explosion of the solution space as the number of products and constraints increases. Consequently, exact approaches are mainly suitable for small- to medium-sized problem instances.

A more recent trend focuses on parameter-driven heuristics that exploit problem structure, particularly product categorization and shelf-level constraints. These studies show that by using pre-defined rules, prioritization schemes, or reduced solution spaces, it is possible to maintain high solution quality while significantly lowering computational requirements. Category-based approaches, where products are grouped by profit, size, or vertical categories, allow more strategic allocation, especially in multi-level shelves or complex retail layouts. These methods demonstrate that integrating domain knowledge into heuristics can produce efficient, near-optimal solutions suitable for large-scale SSAP instances.

Few integrated models simultaneously examined product selection, shelf space allocation and inventory replenishment decisions to optimize retailers' profit under the given operating limitations [12, 13]. These models often try to find a balance between consumer demand patterns and inventory holding costs, striving to maximize profitability through efficient resource utilization. Additionally, advanced algorithms are frequently employed to find optimal product mix on-shelf arrangements, ensuring that high-demand items are readily available while minimizing overstock and stockouts.

A lot of research [9–11, 15, 16] examined a retail shelf-space allocation problem. These investigations typically focus on optimizing the use of limited shelf space to enhance product visibility and sales, incorporating factors such as consumer purchasing behaviour, product profitability, and cross-merchandising effects. Advanced methodologies such as linear and non-linear programming, as well as simulation models, have been employed to develop more sophisticated and effective shelf-space allocation strategies.

To address scalability issues, researchers have increasingly turned to heuristic and metaheuristic methods. Techniques such as simulated annealing, genetic algorithms, and tabu search have been applied to SSAP to generate near-optimal solutions within practical computational times. These methods can handle more complex constraints, including product adjacency restrictions, category-specific shelf allocations, and multi-level racks. However, many of these heuristics rely heavily on stochastic processes, which can result in inconsistent performance across different problem instances. Furthermore, they often lack structured mechanisms to reduce the solution space efficiently, leading to longer computation times for large-scale instances.

Gajjar and Adil [10] suggested three techniques to solve Yang and Chen's [19]'s linear shelf space

allocation model. These heuristics included building an initial solution and then utilizing neighbourhood searches to refine it. Gajjar and Adil [10] created a novel technique for creating the initial solution using the fractional solution of linear programming relaxation of the linear shelf space allocation model [10]. This method outperformed the approach taken by Yang [18]. A suggested neighbourhood relocation combined with the new neighbourhood search yielded a better improvement over the original solution. Furthermore, three new heuristics were determined to perform better than the current heuristic created by Lim et al. [15] based on empirical results. Through empirical research, the authors contrasted three developed heuristics with the current heuristics used in stores [15]. An example of an existing retail store that made more money than the present allocation scheme is used to test the suggested heuristics [10].

Lim et al. [16] created an extension of the shelf space allocation problem to accommodate a realistic range of requirements and restrictions encountered in the industry. Lim et al. [16] employed a combination of heuristics in a 5-phase "squeaky wheel" optimization with a local search method to achieve near-optimal solutions to this problem [16].

Metaheuristics are mostly employed to prevent local optimum traps in order to address the shelf space allocation problem. These advanced methods are often inspired by the mechanisms of the natural world, including the processes of evolution (genetic and differential evolution), the behaviour of animals during foraging (particle swarm, bacterial, ant, and artificial bee colonies), as well as physical and other phenomena (simulated annealing and harmony search) [20].

There is a lot of interest in the retail industry in improving product allocation despite the fact that this topic has not received much research. In order to optimize shelf-space allocation, a workable linear allocation model is examined in the research by Lim et al. [15]. In order to handle additional needs like product groups and non-linear profit functions, the authors expanded the model. In addition to offering a network flow solution, Lim et al. [15] proposed a method that blended a metaheuristic approach to space allocation with a robust local search. This technique is adaptable and efficient, as it can solve both linear and non-linear problems of realistic size while delivering near-optimal solutions with easily implemented algorithms in reasonable timescales. It offers retailers alternatives for more efficient and profitable shelf management, as well as higher-quality planograms [15].

The cross-elasticity impact on each period can likewise be captured by the suggestion of the linear integer programming approach. Adding real-life elements, however, exponentially increases the problem's complexity. Consequently, Raut et al. [17] also suggested using evolutionary algorithms and greedy heuristics to overcome the issue. In addition, the authors suggested a novel kind of chromosomal organization in the genetic algorithm to address the multi-period shelf-space problem [17].

The retail shelf-space decision model presented in the study by Hansen et al. [11] took into account product cross-elasticity, vertical and horizontal position impacts, and a non-linear profit function. Hansen et al. [11] suggested formulating the non-linear profit function using linear programming, which can provide an optimal solution to the shelf space issue. Hansen et al. [11] outlined prospective developments in heuristic and meta-heuristic algorithms and conducted a field experiment and simulation to compare the strategies [11].

Some of the metaheuristics are categorized as socio-inspired metaheuristics [14], while others are referred to as nature-inspired metaheuristics in literature surveys [7]. Socio-inspired metaheuristics, such as Particle Swarm Optimization and Ant Colony Optimization, draw inspiration from social behaviours

and interactions observed in nature. On the other hand, nature-inspired metaheuristics, including Genetic Algorithms and Simulated Annealing, are based on natural processes like evolution and thermodynamics. Both categories aim to efficiently solve complex optimization problems by mimicking the adaptive and intelligent behaviours found in their respective inspirations.

Chakhlevitch and Cowling [6] were the first to coin the term hyperheuristics. The concept is that utilizing a series of heuristics rather than a single heuristic improves efficiency since the algorithm can handle a group of issue cases. As a result, the algorithm solved the issue while simultaneously identifying an efficient solution [6].

The growing interest in hyperheuristics is partially due to challenges in utilizing (meta)heuristics, but it is mostly due to a desire to automate the selection of appropriate heuristics for a given situation. Although heuristic strategies have been proven to be effective in solving real-world computational search issues, they are not without flaws. One disadvantage is that it is difficult to apply to new situations or different instances of identical problems [20]. Finally, custom-made procedures found in the literature are costly to develop and maintain [5].

Bai et al. [3] made a substantial contribution to the developing of planograming instruments that are sufficiently effective. In their study, the authors explored the impacts of shelf space on demand while modelling shelves as two-dimensional. Furthermore, Bai et al. [3] suggested a hyper-heuristic approach that is effective in solving this two-dimensional planogram issue [3].

A hyper-heuristic approach based on simulated annealing was presented by Bai and Kendall [2] to handle many issue instances with varying space ratios and problem sizes. The outcomes of the research by Bai and Kendall [2] demonstrated that, for every issue instance, the developed simulated annealing hyper-heuristic performed noticeably better than two traditional simulated annealing methods and other hyper-heuristics [2].

An aggregated optimization model for shelf space allocation is presented by Binguleret al. [4]. To make it more practical, a modified version of the integer programming paradigm was created. The goal was to find the optimal position for each product item on the shelf to optimize the objective function while accounting for the space elasticity factor. The authors suggested the use of a simulated annealing (SA) method to assign products to shelf space while keeping in mind certain retailer's restrictions [4].

Using a reduced variable neighbourhood search-based (RVNS) hyperheuristic for the shelf space allocation problem, Yu et al. [20] developed a high-level heuristic called HyRVNS, a dual-purpose framework that finds the optimal combination of heuristics to apply in order to get a better solution while also solving the problem itself [20]. Burke et al.'s [5] survey suggested that HyRVNS belongs to the perturbation hyperheuristic category and uses a randomized selection of non-learning low-level heuristics.

Using piecewise linearization, Gajjar and Adil [8] developed an alternative linear model to the non-linear shelf space allocation model [8]. With their linear model, they used linear programming relaxation to attain tight upper bounds. In comparison to Bai's [1] meta/hyper-heuristics, simulated annealing heuristics, and greedy heuristics [1]. Gajjar and Adil [8] created a local search heuristic that produced superior results. However, when the issue size (items, shelves) increased, the CPU time needed to solve the non-linear shelf space allocation model via local search heuristic increased significantly. Therefore, to solve the non-linear shelf space allocation model of realistic size, Gajjar and Adil [9] presented in their study a dynamic programming heuristic, which is significantly quicker than the local search heuristic [9].

The increased focus on hyperheuristics is mostly due to an attempt to automate the process of selecting the best heuristics for a given issue. However, some of the challenges associated with employing (meta)heuristics have also played a role [20]. These challenges include the need for significant expertise to fine-tune metaheuristic parameters and the difficulty of generalizing solutions across different problem domains.

Despite these drawbacks, heuristics remain widely used due to their practicality in providing quick and reasonably good solutions in many real-world applications. Their ability to deliver timely solutions makes them valuable in fields where decision-making speed is crucial, even if it means sacrificing the guarantee of absolute optimality.

Hyperheuristics address these issues by providing a higher level of abstraction, capable of dynamically choosing or generating heuristics based on problem-specific characteristics and performance feedback. Consequently, hyperheuristics not only enhance the robustness and adaptability of solution approaches but also reduce the dependency on expert intervention for heuristic optimization.

Although the literature provides numerous methodologies for SSAP, a clear gap exists in applying deterministic, parameter-driven heuristics that simultaneously handle vertical and horizontal product categorization, incorporate multiple real-world constraints (e.g., product adjacency, allergen separation, profit prioritization), and systematically reduce the solution space for large-scale instances. Most existing heuristics either focus on single-dimension optimization, lack rigorous control over the solution space, or employ stochastic processes with inconsistent outcomes.

The methodology proposed in this research addresses this gap by introducing flower-cutting heuristics that:

- Exploit both vertical and horizontal product categorization.
- Apply a deterministic, parameter-driven approach for consistent solution quality.
- Use 17 carefully tuned parameters to reduce the solution space while preserving near-optimal profits.

Ensure scalability and adaptability to large and complex SSAP instances, which are infeasible for exact solvers or standard metaheuristics.

Resulting in combining these elements, the proposed method represents a novel contribution to SSAP research, offering a practical, efficient, and robust solution methodology suitable for real-world retail store optimization.

3. Problem definition and formulation

The shelf space allocation problem related to retail stores is examined in the current research. Products are sequentially categorized vertically and horizontally. Both vertical and horizontal categorization techniques can be helpful at the same time, depending on the retail store's features, the products it holds, and how it is operated. This kind of categorization has various advantages. The employer needs to be aware of the risks associated with product storage. It's critical to remember that proper categorization and labelling is a part of accident prevention.

Many retailers use a combination of vertical and horizontal categorization to leverage the benefits of both strategies. For example, vertical categorization might be used to display a brand's entire product line, while horizontal categorization can highlight featured products across different brands. This hybrid approach can enhance customer experience, improve navigation, and optimize product visibility. In this research, we provide the rules according to which it is possible to simultaneously assign category tags to the products and shelves in the SSAP under investigation.

H – the horizontal shelf level for brand assortment. Anyone cannot place other products here. It is not possible to classify these products in another horizontal category.

H^+ – the horizontal shelf level for general assortment. Products are able to fill multiple shelves. This tag category allows for the specification of shelf levels (up, middle, and bottom); it is used for a general store assortment and encompasses all shelf levels.

V^+ – the vertical category is used for a specific product category and encompasses all shelf levels. This tag is the vertical category. Every product has a single vertical category allocated to it. Every product, though, can be placed under a few different horizontal categories.

Typically, shelves with tags H^+ contain a large number of products without any further information on the product category or level. As a result, they fit on any shelf in a category. Compared to products with tags H, those with tags H^+ are harder to place.

It is assumed that the shelves on the racks can be adjusted in height and depth such that various shelves have varying dimensions. The most well-liked and often requested products should subsequently be positioned at the customer's eye level. A rule like this expedites the search procedure and cuts down on order servicing time. A single rack's shelves are all the same length, ensuring uniformity in rack structure and simplifying the process of adjusting shelf heights.

The products that are stored on the shelf may have a wide range of characteristics. These attributes can be very diverse and include attributes like weight, size, shape, fragility, and perishability, to name a few. Thus, we deal with incompatible products in this research. Because of the risks involved, such products should not be distributed in close proximity to one another. Additionally, the products may differ from one another in terms of visual attractiveness, branding components, and package materials, all of which have an impact on the nearby product. As a result, similar products cannot be kept next to one another if they are arranged on the same shelf, as their proximity might cause confusion or diminish their individual branding impact.

Products that need separate storage are another category of unique products that are part of the research. These are goods that, for example, need to be kept on different shelves due to various storage conditions in order to avoid cross-contamination or mixing with another product that is identically placed on the shelf. For instance, allergenic foods must be stored separately from non-allergenic ones to prevent cross-contact, which could pose serious health risks to consumers. Similarly, products with strong odours should be kept away from those that can easily absorb smells. By adhering to these guidelines, the retailer can ensure that the storage system is both efficient and safe, accommodating the diverse needs of different products while minimizing the risks of contamination, damage, or misplacement.

When putting products on the shelf, retail managers and employees must be sure to consider these unique characteristics. By understanding and accommodating the unique attributes of each product, they can ensure optimal storage conditions, minimize the risk of damage or spoilage, and facilitate efficient re-

trieval and inventory management processes. Recognizing the varied characteristics, such as weight, size, fragility, and perishability, allows for a more tailored and effective approach to retail store organization.

This proactive strategy helps in maintaining the integrity and quality of the products, ultimately leading to better customer satisfaction. For instance, placing heavier items on lower shelves reduces the risk of accidents, while keeping perishable goods in temperature-controlled sections prevents spoilage. Additionally, organizing products based on their visual appeal and branding ensures that similar-looking items are not placed next to each other, thus avoiding confusion and maintaining a clear brand identity.

It's critical to arrange products evenly on store shelves to promote effective inventory control, provide easy access to products, and reduce the possibility of product damage. As a result, the products in the vertical category should be distributed evenly along the shelves. For this reason, the category size tolerance between shelves is specified. This tolerance ensures that no single shelf is disproportionately loaded with a specific type of product, which can lead to uneven weight distribution and potential safety hazards. By adhering to a defined category size tolerance, store staff can create a more organized and stable storage environment, preventing overloading and facilitating smoother inventory management.

It is possible to distinguish clearly between various products and categories by using rack dividers. This makes it simpler for pickers and customers to locate the products and helps avoid product mix-ups. The vertical category can be shown on the rack by using the minimum category size setting. Additionally, the use of rack dividers plays a pivotal role in distinguishing between various products and categories. These dividers provide clear separation and organization, making it simpler for merchandisers and customers to locate products quickly and accurately. By preventing product mix-ups, rack dividers contribute to reducing errors in order fulfilment and enhancing overall efficiency. They also aid in maintaining the integrity of different product categories, ensuring that each item is stored in its designated space without interference from neighbouring products.

Furthermore, the vertical category can be effectively displayed on the rack by utilizing the minimum category size setting. This setting helps maintain a consistent and recognizable arrangement of products, enabling staff to quickly identify and access the items they need. By implementing these strategies, retail managers can optimize the storage layout, streamline the cutting process, and minimize the risk of damage or misplacement.

The sort of product, its packaging, the presence of labels, and the package's look all affect how a product is oriented on the shelf toward the customer. However, the fundamental idea is to make sure the product has optimal visibility and that the customer can see it well enough. Therefore, depending on the type of packaging and label visibility, products on shelves can be stacked in three orientations: front, side, or top, to make storage easier while using less storage space. Making thoughtful decisions on product orientation and the number of its SKUs can improve retail throughput and decrease out-of-stocks.

Insightful forecasts for future demand and sales trends can be obtained from historical product movement data. By looking at past sales volumes, seasonality, and shifts in customer behaviour, businesses can increase stock levels, lower the risk of overstocking or stockouts, and more correctly anticipate their inventory requirements. It follows that the profit from product sales or product movement is assumed to be known.

The specifics of the examined issue are displayed in Figure 1. The colours grey, yellow, and black indicate vertical category assignment (tag V^+). Brand goods are arranged into a horizontal category (tag

H) at eye level. These things obviously cannot be kept on other shelves. Brand products cannot be used to store other products from the general assortment that are kept on the shelves. Product packaging from a general assortment, such as bottles, tubes, jars, and glass, allows for the specification of multiple shelf heights. The horizontal levels are allocated according to product packages (such as bottles, tubes, jars, and glass) and the picker's and customer's convenience (eye level, arm level, knee level, etc.) - tag H^+. Standard packaging and containers facilitate stacking and contribute to a unified look on the racks.

In the first grey category, plastic bottles can be stored on more than one shelf. In the first grey category, purple products are required to be stored on different shelves. In the third black category, blue products are products, i.e., products that must be stored on the same shelf in the same category but not near because of their similarity and the possibility of confusion. The highest eye-level shelf of all categories is for placing the brand assortment. Other products from the general assortment could be placed on any other shelves.

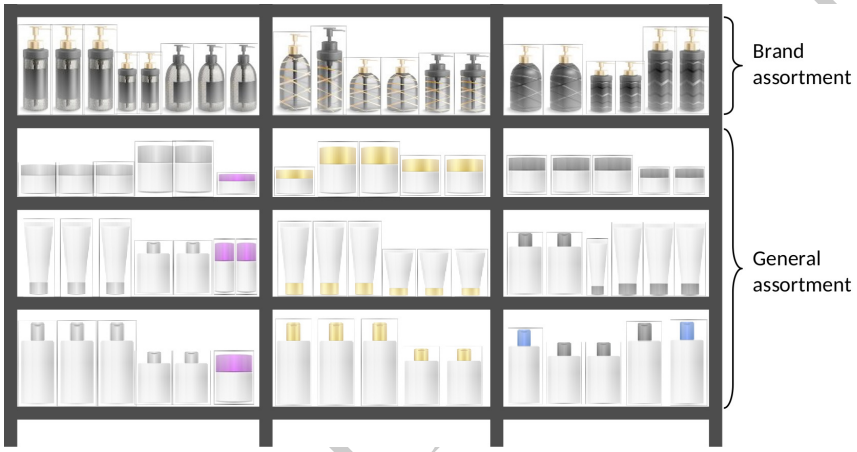

Figure 1. Planogram with vertical and horizontal category bands: vertical – grey, yellow, black; horizontal – brand, general products

The objective of the SSAP within the retail store is to maximize profit and the efficiency of product movement (profit). To achieve this, the retailer must determine the shelf placement for each product, taking into account its vertical and horizontal categorizations. Additionally, the retailer needs to decide on the number of Stock Keeping Units (SKUs) to be allocated per shelf, the orientation of the product for the customer (whether front, side, or top), and consider several types of constraints: shelf, product, multi-shelf, and category-specific constraints.

Moreover, the vertical and horizontal categories help determine the most effective placement strategy to minimize retrieval times. Careful consideration of these factors ensures that each product is positioned in a manner that maximizes accessibility and efficiency, thereby enhancing overall operational performance.

Parameters and indices used in a model: S - total number of shelves; P - total number of products; K - total number of categories; T - total number of tags; i, a, b - shelf index, i, a, b = 1,...,S; j, c, d - product index, j, c, d = 1,...,P; k - category index, k = 1,...,K; t - tag index, t = 1,...,T; r - orientation

index, $r \in \{0, 1, 2\}$;

$$r = \begin{cases} 0, & \text{for front orientation,} \\ 1, & \text{for side orientation,} \\ 2, & \text{for top orientation.} \end{cases}$$

Parameters of the shelf i: s_i^l - length; s_i^h - height; s_i^d - shelf depth; s_{ti}^g - shelf binary tag t;

$$s_{ti}^g = \begin{cases} 1, & \text{if shelf } i \text{ is tagged,} \\ 0, & \text{otherwise.} \end{cases}$$

Parameters of the product j: p_j^w - width; p_j^h - height; p_j^d - depth; p_j^u - unit movement/profit; p_{tj}^t - tag t; p_j^k - category; p_j^s - group of products for separate storage (not on the same shelf); p_j^n - group of incompatible products (must be allocated on the same shelf but not side by side); p_{jr}^o - orientation binary parameter;

$$p_{jr}^o = \begin{cases} 1, & \text{if specific orientation is available,} \\ 0, & \text{otherwise;} \end{cases}$$

f_j^{min},f_j^{max} - minimum, maximum number of SKUs; s_j^{min},s_j^{max} - minimum, maximum number of shelves for allocation of the product; p_j^l - limitation of the product in the store;

Additional product parameter expressions:

p_{jr}^w - width considering orientation;

$$p_{jr}^w = \begin{cases} p_j^w, & \text{if } r = 0, \text{ width for front orientation,} \\ p_j^d, & \text{if } r = 1, \text{ depth for side orientation,} \\ p_j^h, & \text{if } r = 2, \text{ height for top orientation;} \end{cases}$$

p_{jr}^d - depth considering orientation;

$$p_{jr}^d = \begin{cases} p_j^d, & \text{if } r = 0, \text{ depth for front orientation,} \\ p_j^w, & \text{if } r = 1, \text{ width for side orientation,} \\ p_j^w, & \text{if } r = 2, \text{ width for top orientation;} \end{cases}$$

p_{jr}^h - height considering orientation;

$$p_{jr}^h = \begin{cases} p_j^h, & \text{if } r = 0, \text{ height for front orientation,} \\ p_j^h, & \text{if } r = 1, \text{ height for side orientation,} \\ p_j^d, & \text{if } r = 2, \text{ depth for top orientation;} \end{cases}$$

Parameters of the category k: c_k^m - minimum category size as a percentage of the shelf length; c_k^t - category size tolerance between shelves in the category as a percentage of the shelf length.

Parameters of the tag t: b_t^n - tag type, $b_t^n = \{H; H^+; V^+\}$; b_{tij}^t - product to shelf compatibility tag;

$b_{tij}^t =$ - for the horizontal, e.g. brand products level shelves;

$$b_{tij}^t = \begin{cases} 1, & \text{if } s_{ti}^t = p_{tj}^t \wedge b_t^n = \{H\}\ , \\ 0, & \text{otherwise,} \end{cases}$$

t = 1,...,T - for the horizontal, e.g. brand products level shelves;

$$b_{tij}^t = \begin{cases} \min(p_{tj}^t, 1), & \text{if } b_t^n = \{V^+\}, \\ 1, & \text{if } p_{tj}^t = 1 \wedge s_{tj}^t = p_{tj}^t \wedge b_t^n = \{H^+\}\ , \\ 0, & \text{if } p_{tj}^t = 1 \wedge s_{tj}^t \neq p_{tj}^t \wedge b_t^n = \{H^+\}\ , \\ 1, & \text{if } p_{tj}^t = 0 \wedge b_t^n = \{H^+\}\ , \end{cases}$$

t = 1,...,T - for the horizontal and vertical, e.g. general assortment shelves.

Decision variables:

$$x_{ijr} = \begin{cases} 1, & \text{if product } j \text{ is placed on shelf } s \text{ at orientation } r, \\ 0, & \text{otherwise.} \end{cases}$$

Here, x_{ijr} is a product placement binary variable for all $i = 1, \ldots, S$, $j = 1, \ldots, P$, $r \in \{0, 1, 2\}$.

f_{ijr} - the number of SKUs of product j on shelf s at orientation r.

The criteria function of the SSAP can be formulated as follows:

$$\max \sum_{i=1}^{S} \sum_{j=1}^{P} \sum_{r=0}^{2} p_j^u f_{ijr} \tag{1}$$

subject to the following constraints:

1. Shelf constraints:

   The total product width is within the shelf length.

$$\forall(i)[\sum_{j=1}^{P} \sum_{r=0}^{2} p_{jr}^w f_{ijr} \leq s_i^w] \tag{2}$$

   The product height must fit the shelf height.

$$\forall(i, r, j : p_{jr}^h > s_i^h)[f_{ijr} = 0] \tag{3}$$

   The product depth must fit the shelf depth.

$$\forall(i, r, j : p_{jr}^d > s_i^d)[f_{ijr} = 0] \tag{4}$$

2. Product constraints:

   The product is placed on the shelf.

$$\forall(i, j, r)[x_{ijr} \leq f_{ijr} \leq f_j^{max}] \tag{5}$$

Minimum and maximum number of product SKUs.

$$\forall(i,j)[f_j^{min}x_{ijr} \leq \sum_{r=0}^{2} f_{ijr} \leq f_j^{max}x_{ijr}] \tag{6}$$

Specific orientation (front, side, top) is possible for the product.

$$\forall(i,j,r)[x_{ijr} \leq p_j^o] \tag{7}$$

Only one specific orientation is possible is possible for the product.

$$\forall(i,j)[\sum_{r=0}^{2} x_{ijr} \leq 1] \tag{8}$$

If products are required to be stored separately, they must be placed on different shelves.

$$\forall(i)\forall(c,d:p_c^s = p_d^s, c \neq d, c,d = 1,\ldots,P)[\sum_{r=0}^{2} x_{icr} + \sum_{r=0}^{2} x_{idr} \leq 1] \tag{9}$$

If products are marked as incompatible, they must be placed on the same shelf.

$$\forall(i)\forall(a,b:p_a^n = p_b^n, a,b = 1,\ldots,P)[\sum_{r=0}^{2} x_{iar} = \sum_{r=0}^{2} x_{ibr}] \tag{10}$$

If products are marked as incompatible products, they must not be placed nearby.

$$\forall(k,i,c)[\sum_{j=1,p_j^k=k}^{P}\sum_{r=0}^{2} x_{ijr} - \sum_{j=1,p_j^n=c,p_j^k=k}^{P}\sum_{r=0}^{2} x_{ijr} \geq \sum_{j=1,p_j^n=c,p_j^k=k}^{P}\sum_{r=0}^{2} x_{ijr} - 1] \tag{11}$$

Product on-the-shelf placement and SKU relationships.

$$\forall(i,j,r)[\frac{s_i^w x_{ijr}}{p_{jr}^w} \geq f_{ijr}] \tag{12}$$

3. Multi-shelves constraints:

Minimum and maximum number of shelves on which the product may be placed.

$$\forall(j)[s_j^{min} \leq \sum_{i=1}^{S}\sum_{r=0}^{2} x_{ijr} \leq s_j^{max}] \tag{13}$$

Product storage limit if the product is placed on multiple shelves.

$$\forall(j)[\sum_{i=1}^{S}\sum_{r=0}^{2} f_{ijr} \leq p_j^l] \tag{14}$$

If the product is placed on multiple shelves, the shelves must be allocated nearby.

$$\forall(j)\forall(a,b:|a-b|\neq 1\wedge a<b,a,b=1,\ldots,S)\forall(r)[x_{ajr}+x_{bjr}\leq 1] \tag{15}$$

4. Category constraints:

   Tags compatibility for the shelves and products.

$$\forall(i,j)[\prod_{t=1}^{T} b_{tij}^{t}\geq\sum_{r=0}^{2}x_{ijr}] \tag{16}$$

   Minimum category size, if the products from the category are placed on the shelf.

$$\forall(i,k)[(\sum_{j=1,p_j^k=k}^{P}\sum_{r=0}^{2}p_{jr}^{w}f_{ijr}\geq[s_i^l c_k^m])\vee(\sum_{j=1,p_j^k=k}^{P}\sum_{r=0}^{2}f_{ijr}=0)] \tag{17}$$

   Category size tolerance, i.e., products may differ in width, the resulting category block remains balanced, stable, and visually attractive, consistent with standard retail layout practices.

$$\forall(k)[max_{i=1,\ldots,S}(\sum_{j=1,p_j^k=k}^{P}\sum_{r=0}^{2}p_{jr}^{w}f_{ijr})-min_{i=1,\ldots,S}(\sum_{j=1,p_j^k=k}^{P}\sum_{r=0}^{2}p_{jr}^{w}f_{ijr})\leq[max_{i=1,\ldots,S}(s_i^l)c_k^t]] \tag{18}$$

5. Decision variables:

   The product is placed on the shelf.

$$\forall(i,j,r)[x_{ijr}\in\{0,1\}] \tag{19}$$

   The number of product SKUs.

$$\forall(i,j,r)[f_{ijr}=\{f_j^{min}\ldots f_j^{max}\}] \tag{20}$$

4. Flower-cutting heuristics for SSAP

We present innovative flower-cutting heuristics designed to tackle the challenges observed in the investigated SSAP studies. Our methodology features two unique heuristic variants, each defined by a specific sequence for allocation sorting. To explain how the proposed flower-cutting heuristics can be used to solve the investigated SSAP, let's consider the following terms:

The *shelf allocation* is a sequence of numbers, each indicating whether a product is placed on the shelf. Products on the shelf can be positioned in one of three ways: front-facing (0/1), side-facing (0/2), or top-facing (0/3). The coding system specifies the orientation of products on the shelf. A value of 0 indicates that the product is not placed on the shelf.

The *product allocation* is a sequence of numbers that refers to the number of SKUs positioned on the shelf.

The *total profit of the category* is defined as the sum of profits of products allocated on all shelves of the category.

The *total width of the category* is defined as the maximum sum of the widths of products among all shelves in which products are allocated in the category.

The *total profit* is defined as the sum of the profits of products allocated on all shelves of all categories.

The *profit ratio* is calculated by dividing the total profit made by the products on all of the shelves in the planogram by the total amount of space that these products take up. This computation does not take free space into account.

The principal heuristic is segmented into multiple stages, which are detailed below.

Stage 1: Problem definition and understanding.

- Identify the problem. The SSAP with categorization aimed to be solved is defined. Next, the objective (profit maximization) and constraints (product, shelf, multi-shelf, and category) are defined.
- Define success metrics. The criteria for evaluating the algorithm's performance are the number of solutions to be generated, the heuristic's accuracy (profit ratio compared to the CPLEX solver), and solution time.

Stage 2: Initial design.

- Simplify the problem. We break down the problem into smaller, manageable components, which in our case are vertical product categories, and try to allocate products separately in each category. Next, we combine product allocation for all categories.
- Define principles. We develop a list of rules based on the shelf space allocation knowledge domain, according to which we will generate only profitable solutions without checking the whole solution space. Special attention is given to the fact that such a rule is simple, intuitive, and could generate product allocation fast.

Stage 3: Prototype development.

- Implement basic heuristics. Here, we write the initial code to implement the defined heuristics.
- Create test cases. We develop a diverse set of test cases to evaluate the heuristic algorithm. We test the heuristics on different problem sizes to see how many solutions can be generated and how much we should reduce the solution space.

Stage 4: Evaluation and refinement.

- Initial testing. We run the heuristic algorithm on the test cases. Next, we collect performance data and identify any issues or inefficiencies.
- Analyze results. We compare the algorithm's performance against the success metrics (overall profit) established by the CPLEX solver. Next, we identify patterns or areas where the heuristic performs poorly, i.e., too many solutions are generated, or the solution quality is not sufficient compared to the solver. This is the field of heuristic parameter development.

Stage 5: Optimization.

- Performance tuning. We optimize the algorithm's code for speed and efficiency by introducing a set of parameters that are needed to decrease the solution space and generate more profitable solutions. We utilize advanced data analysis for parameter development if necessary.
- Scalability testing. We test the heuristic algorithm on larger, more complex problem instances.
- Iterative improvement. We continuously refine and optimize the heuristics based on testing results. We try to set different input values for the developed parameters. We conduct multiple iterations to achieve a balance between accuracy and efficiency.

Stage 6: Validation and deployment.

- Final testing. We validate the algorithm against a comprehensive set of test cases. We ensure robustness and reliability under various conditions.
- Documentation. We document the heuristics, implementation details, and usage guidelines. We provide clear instructions for parameter setting and solution quality improvement.
- Deployment. At this step, heuristics could be integrated into the shelf space management software of the retail store. Performance should be monitored, and retailer feedback should be gathered for further improvements.

Creating a heuristic algorithm involves defining a set of rules or guidelines that can efficiently solve a problem. The stages explained above are meticulously designed to enhance the efficiency and accuracy of solution development.

Let's explain the main steps of the flower-cutting heuristics implementation.

- Define principles.

Preparing the garden. Set up the input parameters necessary for creating the garden. This stage lays the foundation for the following steps. Imagine a garden where flowers of different heights and different sizes of flower buds are growing. Moreover, in various garden parts, various densities of flowers exist. This represents a complex solution space for exploration and optimization. Each flower represents a single solution. The gardener must prepare to navigate and analyze this intricate environment, setting the grounds for effective problem-solving strategies.

Identifying flower clearings. The heuristic constructs potential solutions by systematically determining which areas of the garden to focus on during the flower-cutting process. This involves selecting clearings of the garden likely to yield the best results based on predefined criteria. By narrowing down the search area, the heuristic ensures a more efficient and targeted problem-solving approach. Moreover, inside the selected clearing, a lot of flourishing flowers may grow. Therefore, the gardener's task is to define the rule according to which he will cut not all flowers in the selected clearing, i.e., the gardener narrows down the solution space area inside the profitable region.

- Implement basic heuristics.

  Cutting the flowers: Solutions are created based on specific criteria, a predetermined sorting order, and interval parameters, ensuring a systematic approach to the problem. The algorithm, acting as

the gardener, cuts selected flowers on the chosen clearings, leaving lots of flowers growing on the clearings, following precisely defined parameters to enhance both accuracy and efficiency. These parameters include:

- Length of gardener's movement path: This parameter controls the distance travelled by the gardener while picking flowers, optimizing the route to maximize the number of high-quality flowers gathered while minimizing unnecessary movement.
- Target flower height: This parameter specifies the minimum height requirements for flowers to be cut, ensuring that only those meeting certain height thresholds are picked. By focusing on higher flowers, the gardener targets the most profitable ones, enhancing the overall value of the cutting process.
- Target flower neighbouring interval: This parameter controls the interval between the flower that is cut and other flowers on the selected clearing. As in the real garden, if the flowers seem to be of the same size and can be cut, the gardener does not cut all flowers on the clearing but cuts flowers with some intervals between other flowers. In the reduced by the previous parameters solution space when the profitable clearing where higher flowers are growing is found, it could be a lot of the solutions with almost the same profit characteristic, so the goal is to check not all of them but with some interval.
- Gardener's basket size: The basket's capacity determines the total number of flowers that can be cut in a single outing. This parameter is crucial for balancing the quantity and quality of flowers cut. By limiting the basket size, the gardener prioritizes quality over quantity, aiming to collect the biggest in the sense of height and bun size flower available.

- Performance tuning.

  Every generated solution is carefully designed to meet all pre-established requirements, guaranteeing that it is feasible to implement in the specified problem area. These ideas are then subjected to a thorough assessment using predetermined measures, such as resource efficiency or profitability. The tuning parameters help to find the profitable flowers in order to preserve the best possible selection.

  - Parameters of flower clearing creation:

    Parameter 1 – the maximum category width while forming product allocations.

    Parameter 2 – the minimum number of products that can be placed on the shelf while forming product allocations.

    Parameter 3 – the maximum number of products that can be placed on the shelf while forming product allocations.

    Parameter 4 – the set of profitable groups of products to be placed on the shelf.
  - Parameters of moving along the selected flower clearings:

    Parameter 5 – the minimum category width after forming product allocations.

    Parameter 6 – the maximum category width after forming product allocations.

    Parameter 7 – if the grouping option (for each total width, only 1 product allocation with the maximum total profit) is used.

Parameter 8 – the maximum number of product allocations on the shelf.

  * Sorting rule 8.1: category width ↑, category profit ↓.
  * Sorting rule 8.2: category profit ↓, category width ↑.

Parameter 9 – if the grouping option (for each total profit and profit ratio, take only 1 product allocation with the minimum total width) is used.

Parameter 10 – the maximum number of product allocations of the category.

  * Sorting rule 10.1: profit ↓, profit ratio ↓.
  * Sorting rule 10.2: profit ratio ↓, profit ↓.

– Parameters of the interval between cut flowers on the selected clearings:

Parameter 11 – the interval of taking the product allocations on the shelf after taking all product allocations according to parameter 8.

Parameter 12 – the maximum number of product allocations on the shelf created with the interval parameter 11, the sorting rule is the same as in parameter 8.

Parameter 13 – the interval of taking the product allocations of the category after taking all product allocations according to parameter 10.

Parameter 14 – the maximum number of product allocations of the category created with the interval parameter 13, the sorting rule is the same as in parameter 10.

– Parameters of the flowers to be cut:

Parameter 15 – the minimum profit for each category.

Parameter 16 – the maximum profit for each category.

Parameter 17 – the minimum total profit

- Iterative improvement.

The solutions are continuously improved and refined thanks to this selection approach. The method systematically improves the overall quality of the chosen solutions by concentrating on some of the similar solutions and cutting several of the flowers that satisfy the value criteria more. This strategy maximizes the profitability of the final outcome while simultaneously optimizing resource use. Cutting only several flowers among the group with the same characteristics significantly reduces the gardener's time to find the flowers and does not overfill the basket with cut flowers.

The gardener adjusts the parameters and repeats the process of finding and cutting the flowers. Due to the deterministic nature of parameter selection and solution generation, the number of iterations is very low. Tuning parameters are continuously improved upon by taking into account prior attempts to discover successful solutions.

Sorting rules serve as two-tiered prioritization filters. They regulate the sorting and selection mechanisms used to prioritize candidate product allocation during the solution space reduction phase. The sorting rules ensure that the heuristic evaluates allocations in an order aligned with the strategic focus of the chosen variant, whether prioritizing absolute profit or profit efficiency. They also contribute to computational efficiency by reducing the number of candidate allocations considered in subsequent steps.

Through a series of controlled solution space reductions, the number of product allocations evaluated on each shelf and within each category was systematically decreased. Here, the gardener — a personification of the algorithm — methodically modifies different settings and goes through the flower-cutting procedure again. The goal of each iteration is to improve the overall quality of the solution by optimizing these parameters in light of the knowledge acquired from earlier attempts.

The procedure ends when it meets a predetermined threshold for stopping. These standards are set in order to guarantee the effectiveness and efficiency of the algorithm's functioning. The size of the gardener's basket or the quantity of solutions the gardener could gather in the basket is represented by the stopping criterion.

Figure 2 illustrates the flower garden scenario, with a focus on the specific flower clearing where flowers, representing potential solutions, are growing. The number of selected parts of the garden may be more than one in the real solution space. The flower clearing is defined by a starting height threshold, above which flowers are cut and placed in the basket. In this selected clearing, as in the real world, not all flowers are cut, but flowers with some interval between them. The height thresholds, the widths, and the intervals between flowers in the clearings can vary depending on the clearing. Flowers in other clearings are ignored, even if their heights exceed the thresholds of the selected clearing. The objective is to identify and select clearings with the biggest flowers, ensuring no profitable clearing is overlooked. The distances between selected flower clearings are irrelevant to the gardener's task. The algorithm's execution time for generating and selecting solutions to be checked is calculated based solely on the selected clearings where the gardener cuts flowers.

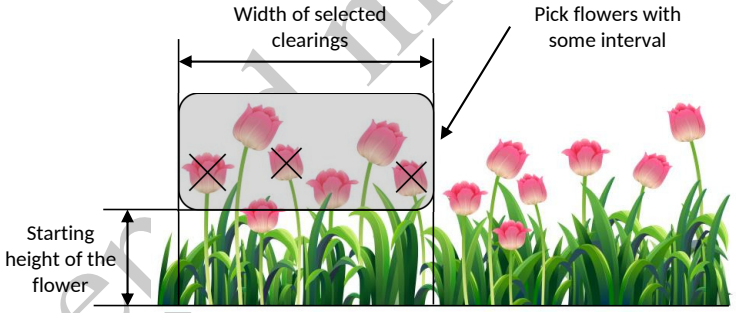

Figure 2. Looking for clearings to pick flowers and cutting flowers with intervals on the clearing

Due to the problem being simplified, the algorithm tries to generate profitable flower clearings for each category. After that, the solution combining all product categories is created. By breaking down the problem into manageable segments, the algorithm can concentrate on optimizing solutions within each specific category, ensuring a thorough exploration of potential high-yielding areas in the garden. After optimizing the solutions for each category, these individual solutions are then combined to create a comprehensive solution encompassing all product categories. This step-by-step approach not only simplifies the complexity of the overall problem but also enhances the likelihood of identifying the most effective and profitable product allocations across the entire solution space.

Because of the flower height and width parameters, profitable clearings from the gardener's point of view are created. By setting specific height and width criteria, the algorithm ensures that only the most promising flowers, representing potential solutions, are selected. This targeted selection process focuses on areas where flowers meet or exceed the defined parameters, optimizing the yield in terms

of both quality and quantity. This reduces the computational time. The resulting clearings are thus tailored to maximize profitability, aligning with the gardener's objective of achieving the best possible outcomes from their efforts in cutting the biggest flower. This strategic approach enables the gardener to efficiently allocate resources and attention to the most lucrative parts of the garden, ensuring a high return on invested physical and time resources.

Due to the implementation of neighbouring cutting intervals within the flower clearings, the time required for the gardener to move along each clearing is significantly reduced. By setting the distance between successive cutting points, the gardener can quickly traverse the area, focusing more on evaluation and selection rather than traversing the clearing he was given. This efficiency gain allows the gardener to investigate a greater number of diverse clearings within the same time frame. Furthermore, if the selected clearing has a lot of flowers of the same size, this interval helps the gardener not to get hung up on cutting all the flowers. Consequently, the gardener can explore a wider variety of potential solutions, increasing the chances of discovering larger and more profitable flowers. This method enhances the overall effectiveness of the search process, enabling a thorough examination of multiple areas and optimizing the likelihood of achieving superior outcomes in terms of flower size and quality.

The size of the gardener's basket imposes a strategic constraint, ensuring that the gardener does not indiscriminately pick all or random flowers. Instead, this limitation forces the gardener to selectively cut only those flowers that are definitely profitable. This selectivity enhances the overall quality of the cut flowers, as the gardener is compelled to prioritize flowers that meet specific profitability criteria. After filling the basket with these carefully chosen flowers, the next objective is to identify the biggest and most valuable flower within this curated collection. This process focuses on profitability or flower size maximization and ensures that the gardener's efforts yield the most advantageous results in terms of quality. By adhering to basket size limitations, the gardener optimizes resource utilization, concentrating efforts on extracting the highest possible benefit from the available floral resources.

This targeted approach not only improves efficiency but also increases the likelihood of uncovering high-quality solutions. Consequently, the methodology supports more precise and effective decision-making in SSAP scenarios.

The representation of individual decision units as flowers in a garden which is the core principle of the heuristic, offers a flexible abstraction that can be adapted whenever solutions are composed by selecting and combining feasible components under constraints. In this context, a flower corresponds to a feasible atomic decision (e.g., a job assignment, routing segment, production batch, or scheduling slot), while a basket corresponds to a complete feasible solution constructed by iteratively selecting the most promising components.

5. Experiment with the flower-cutting heuristics

To evaluate the performance of the developed heuristics in solving the modeled SSAP, an extensive series of experiments was conducted. The primary objective of these experiments was to compare the quality of the solutions produced by the developed heuristics with the optimal solutions generated by the CPLEX solver.

The experimental setup involved running both the heuristics and the CPLEX solver on a diverse set of

problem instances, representing various scenarios and complexities inherent in the SSAP. Each heuristic was methodically tested to assess its ability to deliver high-quality solutions within a reasonable computational timeframe.

Only the best-performing parameter configuration is reported, rather than all the parameter tuning steps. This focused approach ensures that the results presented reflect the most effective and optimized parameter set identified during the experimental process. Throughout the parameter tuning phase, numerous combinations of parameters were systematically evaluated to determine their influence on the performance of the heuristics. By fine-tuning these parameters, the configuration that consistently produced the highest-quality solutions was identified.

For every experiment, key performance metrics, including solution quality, computational time, and proximity to optimal solutions, were meticulously recorded and analyzed. This rigorous evaluation process enabled a comprehensive comparison between the heuristic-generated solutions and those produced by CPLEX, providing valuable insights into the relative effectiveness and efficiency of the proposed heuristics.

The overarching goal of this experiment was not only to evaluate the efficiency of the proposed heuristics but also to explore the potential of expanding the boundaries of problem-solving within the SSAP. By examining how two heuristics performed across varying scenarios, we aimed to uncover patterns and strategies that could lead to more effective shelf space management in real-world applications. This allows to push the limits of traditional optimization techniques and embracing innovative approaches that balance solution quality with computational efficiency. Ultimately, the experiment sought to provide insights into how heuristics can be refined and adapted to tackle increasingly complex and dynamic allocation problems, offering valuable guidance for future advancements in optimization methods.

The computer parameters were:

- Processor: AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz
- System type: 64-bit Operation System, x64-based processor
- RAM: 16 GB
- Operation system: Windows 10

The study included sets of 10, 15, 20, 25, and 30 products that needed to be arranged on four shelf racks with different lengths of shelf: 250, 375, 500, 625, and 750 cm. A variety of product sets were produced with different dimensions, including height, depth, width, and movement/profit, among others.

Two vertical categories were created for product sets of 10, 15, and 20 in the rack layout, and three vertical categories were created for product sets of 25 and 30, defining separate regions for product distribution. The objective of this categorization was to effectively arrange the merchandise, enabling the most efficient use of shelf space.

The experimental design facilitated an extensive evaluation of the developed heuristics' effectiveness across various problem sizes and configurations. By integrating a variety of product parameters and rack specifications, the experiments were structured to offer valuable insights into how well the heuristics manage different real-world scenarios. This approach ensured a thorough understanding of the heuristics' capabilities and limitations in diverse flower-cutting applications.

Different parameters were applied in the test examples experiment. Only one parameter 15 was utilized for 10 product small instances. To shrink the solution space, all 17 parameters were used for large instances (25 and 30 products). To medium instances (15 products), 10 parameters (parameters 5-15) were applied. To the remaining medium instances (20 products), 13 parameters (parameters 2-15) were applied. For both heuristic strategies, the same input parameters were applied in order to reduce the solution space.

In this study, two variants of the proposed flower-cutting heuristic were implemented, denoted as H1 and H2. Both heuristics employ the same core principles of the flower-cutting approach, where the solution space is represented as a garden, each product allocation is considered a flower, and high-profit allocations are "cut" and placed in the gardener's basket. The difference between H1 and H2 lies in the specific rules for selecting and prioritizing product allocations:

- Heuristic H1 prioritizes allocations based on minimum category width first, followed by maximum category profit, using parameters to limit the number of allocations evaluated per shelf (rule 8.1).
- Heuristic H2 emphasizes maximum category profit first, followed by minimum category width, and uses slightly different sorting and selection rules to explore alternative high-quality solutions, using parameters to limit the number of allocations evaluated per shelf (rule 8.2).
- Heuristic H1 prioritizes allocations based on maximum profit first, followed by profit-to-width ratio, using parameters to limit the number of allocations evaluated per category (rule 10.1).
- Heuristic H2 emphasizes profit ratio first, followed by absolute profit, and uses slightly different sorting and selection rules to explore alternative high-quality solutions, using parameters to limit the number of allocations evaluated per category (rule 10.2).

Both heuristics use the same 17 tuning parameters but differ in the sequence and combination of sorting rules and selection intervals applied during the search. By comparing H1 and H2, we can evaluate how variations in the allocation prioritization and selection strategy affect solution quality and computational efficiency.

The performance two just introduced heuristics, H1 and H2, is compared to the best results produced by the commercial CPLEX solver in Table 1. This analysis, which covers several test cases, emphasizes the profit ratio, which shows the ratio of the profits made using the heuristics and the optimal profit of ascertained by the CPLEX solver. For every heuristic and test case, the profit ratio was computed to evaluate how well the heuristics approximated optimal solutions.

According to the analysis, both heuristics produced the optimal results in 17 of the 25 test cases, but not for the same ones. It could be observed that for 20 products set on 250 cm shelf length, the heuristics H2 found an optimal solution, but the heuristics H1 did not. In other cases heuristics H1 was better at finding the optimal solution for 25 products set on 375 cm, while heuristic H2 was not. The effectiveness of the suggested heuristics and the appropriateness of the control settings, which considerably reduced the number of possible solutions while still obtaining the intended results, were highlighted by the fact that both heuristics consistently generated solutions with a quality exceeding roughly 99.9% (99.85% for heuristics H1 and 99.86% for heuristics H2). Moreover, both heuristics produced minimum profit ratio

values of 98.58%. These heuristics concentrate on finding and assessing profitable solutions rather than every potential option.

The solution time of the CPLEX solver varied between 0.36 and 6.48 seconds. The fastest solutions for heuristics H1 and H2 were discovered in 0.05 minutes (2.8 seconds). With an average solution time of 1.95 minutes as opposed to 2.58 minutes for heuristic H2, heuristic H1 performed marginally faster overall. Heuristic H1's slowest solution time was 5.60 minutes. The slowest solution time for heuristic H2 was 7.01 minutes. It was shown that both heuristics took longer to solve medium and large test instances. Interestingly, solving large 3-category instances (25 and 30-product sets) roughly required less time than solving medium instances. This is because more space parameters were set for large instances than for medium ones.

Table 1. Performance of the developed heuristics

Products	Shelf width	Profit H1	Profit H2	Time H1 [min]	Time H2 [min]	CPLEX [s]
10	250	100.00%	100.00%	0.05	0.05	0.44
	375	100.00%	100.00%	0.05	0.05	0.59
	500	100.00%	100.00%	1.19	1.17	0.45
	625	100.00%	100.00%	0.16	0.16	0.78
	750	100.00%	100.00%	0.19	0.20	0.36
15	250	100.00%	100.00%	0.10	0.73	0.56
	375	99.80%	99.80%	1.16	4.23	0.75
	500	100.00%	100.00%	0.21	0.16	0.86
	625	100.00%	100.00%	1.04	1.08	0.78
	750	100.00%	100.00%	1.37	1.40	0.81
20	250	99.94%	100.00%	0.88	0.85	1.08
	375	100.00%	100.00%	2.96	2.97	1.38
	500	100.00%	100.00%	4.02	4.54	0.84
	625	100.00%	100.00%	3.08	6.57	1.20
	750	100.00%	100.00%	5.12	7.01	0.86
25	250	99.42%	99.69%	1.30	1.34	2.08
	375	100.00%	99.81%	3.28	4.88	1.91
	500	100.00%	100.00%	2.55	3.42	1.44
	625	100.00%	100.00%	5.60	5.57	1.89
	750	99.84%	99.84%	1.87	4.17	1.63
30	250	98.58%	98.58%	1.61	1.57	2.52
	375	99.35%	99.35%	2.54	3.44	6.48
	500	99.73%	99.84%	4.61	5.08	2.30
	625	99.59%	99.59%	2.07	2.12	1.89
	750	99.88%	99.88%	1.76	1.84	5.22
Minimum		98.58%	98.58%	0.05	0.05	0.36
Average		99.85%	99.86%	1.95	2.58	1.56
Maximum		100.00%	100.00%	5.60	7.01	6.48

The effect of reducing the solution space by applying parameters 7, 8, 11, 12 is shown in Table 2. Parameter 7 (the first grouping option) was used for all instances except the smallest one (for a set of 10 products), and then parameter 8 was used. The contrast between the evaluated solutions and those that were obtained after applying the previous reduction parameters is shown by the percentages in Table 2. This suggests that even after applying the solution space reduction parameters, not every solution was assessed, but a part of them. An evaluation of product allocations with other parameters but without parameters 7, 8, 11, 12 was given for the smallest example (for a set of 10 products) later, as there is no information on them shown in Table 2.

The reduced solution set was first produced using predetermined ranges for category widths (managed by parameters 5 and 6) that exceed the profit threshold for each category (managed by parameters 15 and 16). Using the minimum and maximum category widths (parameters 5 and 6) as well as the minimum and maximum profit for each category (parameters 15 and 16), the solution space was reduced. After that, the number of solutions on each shelf — which was regulated by parameters 7 and 8 and eventually parameters 11 and 12 for product allocations generated with some interval — was compared with the reduced set.

Mostly, all product allocations were evaluated on the 1st eye-level shelf because it was used only for the specific products, and other shelves were used for the most numerous general assortment. Simply put, there were comparably low product allocations on the 1st shelf; therefore, most of them were evaluated.

For the shelves with a general assortment, applying parameters 7 and 8 notably decreases the solution space. As an example, on the 2nd, 3rd and 4th shelves, sometimes only 0.02%, 0.27%, 0.11%, respectively, were evaluated within an acceptable time frame. But even this number was enough to achieve a solution with a profit ratio 99.84% for 25 product sets on 750 cm and even an optimal solution for 20 product sets on 750 cm (Table 1).

From Table 2 it could be observed that the average evaluated product allocations specified by parameter 8 for the 1st, 2nd, 3rd, 4th shelves were 90.99%, 7.23%, 16.21%, 11.21%, respectively. These values varied from 30.03% to 100.00 for the 1st shelf, from 0.02% to 57.37% for the 2nd shelf, from 0.27% to 100.00% for the 3rd shelf, and from 0.11% to 100.00% for the 4th shelf.

The interval of taking the product allocations on the shelf (parameter 11) was not used for the 1st shelf. For the other shelves, it was set to the value of 2, i.e., after taking product allocations on the shelf specified by parameter 8, the number of product allocations to be taken as specified by parameter 12, and only even product allocations according to the sorting rules 8.1, 8.2 were taken. The number of these product allocations was not high, as the quality of them is significantly worse than the quality of the product allocations specified by parameter 8.

From Table 2, it could be noted that the average number of evaluated product allocations generated with the interval specified by parameter 12 for the 2nd, 3rd, 4th shelves was 9.40%, 19.53%, 11.25%, respectively. These values varied from 0.03% to 63.11% for the 2nd shelf, from 0.31% to 77.99% for the 3rd shelf, and from 0.14% to 59.45% for the 4th shelf.

The number of product allocations generated with intervals on the shelves specified by parameter 12 was significantly lower than the number of product allocations specified by parameter 8. This shows that not all generated product allocations from the previous steps were taken for further evaluation.

The effect of using parameters 9, 10, 13, 14 to shrink the solution space is shown in Table 3. Following the initial product allocations for each category, parameter 10 was applied, which limits the maximum number of product allocations per category across all shelves. The grouping parameter 9 was used only for 3-category instances (25 and 30 product sets). For the remaining 2-category instances grouping parameter 9 was not used.

From Table 3, it could be observed that the average evaluated product allocations generated without interval parameters for the 1st, 2nd, 3rd categories were 52%, 50%, 59%, respectively. These values varied from 3% to 100.00% for the 1st category, from 12% to 100% for the 2nd category, from 26% to 100.00% for the 3rd category.

Table 2. The usage of the maximum number of product allocations on the shelf (Parameters 7, 8, 11, 12) for heuristics H1 and H2

Products	Shelf width	Applying parameters 7, 8				Applying parameters 7, 8, 11, 12		
		Shelf 1	Shelf 2	Shelf 3	Shelf 4	Shelf 2	Shelf 3	Shelf 4
15	250	100.00%	57.37%	100.00%	100.00%	63.11%		
	375	100.00%	10.00%	50.15%	12.92%	11.25%	60.18%	15.50%
	500	100.00%	1.67%	18.32%	4.03%	2.09%	21.98%	4.83%
	625	100.00%	2.75%	24.82%	5.10%	2.93%	26.48%	5.44%
	750	100.00%	1.41%	12.69%	2.52%	1.51%	13.53%	2.69%
20	250	100.00%	8.20%	11.68%	15.40%	9.22%	13.35%	18.48%
	375	100.00%	1.57%	2.11%	2.91%	1.72%	2.37%	3.27%
	500	100.00%	3.98%	5.50%	9.30%	4.38%	6.19%	10.47%
	625	100.00%	0.72%	1.37%	0.72%	0.79%	1.54%	0.81%
	750	100.00%	0.10%	0.27%	0.13%	0.11%	0.31%	0.14%
25	250	100.00%	6.79%	22.28%	15.80%	14.33%	77.99%	55.31%
	375	100.00%	0.97%	7.15%	5.57%	1.93%	16.08%	12.54%
	500	100.00%	0.41%	3.26%	2.13%	0.82%	7.33%	4.78%
	625	100.00%	0.16%	1.64%	0.99%	1.06%	9.84%	6.51%
	750	100.00%	0.02%	0.37%	0.11%	0.03%	0.58%	0.17%
30	250	100.00%	34.29%	33.77%	38.22%	51.44%	50.66%	59.45%
	375	100.00%	10.82%	19.27%	6.19%	16.24%	43.35%	9.63%
	500	51.28%	2.26%	4.47%	1.20%	3.52%	8.94%	2.06%
	625	38.46%	0.76%	1.57%	0.35%	1.19%	3.13%	0.59%
	750	30.03%	0.30%	3.59%	0.68%	0.37%	7.17%	1.02%
Minimum		30.03%	0.02%	0.27%	0.11%	0.03%	0.31%	0.14%
Average		90.99%	7.23%	16.21%	11.21%	9.40%	19.53%	11.25%
Maximum		100.00%	57.37%	100.00%	100.00%	63.11%	77.99%	59.45%

The interval of taking the product allocations of the category (parameter 13) was set to the value of 2 and was used for all categories, i.e. after taking product allocations on the category specified by parameter 10, the number of product allocations to be taken as specified by the parameter 14 and only even product allocations according to the sorting rules 10.1, 10.2 were taken. The number of these product allocations was not high as the quality of them is significantly worse than the quality of the product allocations specified by parameter 10.

From Table 3 it could be noted that the average number of evaluated product allocations generated with the interval for the 2nd, 3rd categories was 1.64%, 3.08%, 8.66%, respectively. These values varied from 0.05% to 8.30% for the 1st category, from 0.27% to 10.94% for the 2nd category and from 0.49% to 18.34% for the 3rd category.

The number of product allocations generated with intervals on the shelves specified by parameter 14 was significantly lower than the number of product allocations specified by parameter 12. This means that not all generated product allocations on the previous steps were taken for further evaluation.

The values set for reduction parameters 5 and 6, which stand for the minimum and maximum category widths following the formation of product allocations, are described in Table 4. The average category width can be computed once possible product allocations on each shelf have been determined. It allows to estimate the category width and profit, even if it is unknown which specific product allocations will be chosen for the final solution.

One value is set for parameter 5, which establishes the minimum category width following the formation of product allocations. To guarantee accuracy, this number is contrasted with the average category

Table 3. The usage of the maximum number of product allocations of the category (Parameters 9, 10, 13, 14) for heuristics H1 and H2

Products	Shelf width	Applying parameters 9, 10			Applying parameters 9, 10, 13, 14		
		Category 1	Category 2	Category 3	Category 1	Category 2	Category 3
15	250	9%	12%	–	2.93%	4.06%	–
	375	3%	12%	–	0.30%	1.22%	–
	500	45%	100%	–	4.51%	–	–
	625	9%	67%	–	0.94%	6.68%	–
	750	5%	26%	–	0.51%	2.64%	–
20	250	100%	43%	–	–	1.94%	–
	375	100%	100%	–	–	–	–
	500	14%	25%	–	1.43%	1.13%	–
	625	41%	19%	–	4.12%	0.84%	–
	750	83%	28%	–	8.30%	2.78%	–
25	250	94%	46%	41%	0.05%	10.94%	11.20%
	375	53%	59%	58%	0.11%	0.27%	16.03%
	500	63%	68%	30%	0.39%	4.97%	1.10%
	625	71%	66%	49%	0.34%	2.22%	0.49%
	750	29%	32%	26%	0.38%	4.23%	11.39%
30	250	92%	100%	100%	0.13%	–	18.34%
	375	40%	25%	68%	0.12%	1.19%	4.40%
	500	37%	35%	91%	0.20%	2.34%	10.46%
	625	76%	57%	73%	2.18%	1.87%	9.96%
	750	78%	75%	52%	2.50%	–	3.19%
Minimum		3%	12%	26%	0.05%	0.27%	0.49%
Average		52%	50%	59%	1.64%	3.08%	8.66%
Maximum		100%	100%	100%	8.30%	10.94%	18.34%

width of the product allocations and the shelf width. The maximum category width, or parameter 6, is expressed as a percentage of the shelf width. It also contrasted with the average category width of the product allocations and the shelf width.

Since settings for the generation of product allocation for small instances took up the complete shelf width, Table 4 demonstrates that parameters 5 and 6 were not applied to the set of 10 products. Values above 100% for parameter 5 for the remaining product sets show that the minimum category width was set higher than the average category widths calculated from possible product allocations.

Category 1 was given more space since it was substantially more profitable than Category 2 and Category 3. On the one hand, the minimum category width for Category 1 compared to the average category width of product allocations was, on average, 90%, varying from 59% to 116%. The minimum category width for Category 2 compared to the average category width of product allocations was, on average, 63%, varying from 43% to 89%. The minimum category width for Category 3, compared to the average category width of product allocations, was, on average, 66%, varying from 50% to 90%. On the other hand, the minimum category width for Category 1 compared to the shelf width was, on average, 43%, varying from 24% to 61%. The minimum category width for Category 2 compared to the shelf width was, on average, 29%, varying from 19% to 43%. The minimum category width for Category 3 compared to shelf width was, on average, 25%, varying from 20% to 36%.

On the one hand, the maximum category width for Category 1 compared to the average category width of product allocations was, on average, 117%, varying from 92% to 138%. The maximum category width for Category 2, compared to the average category width of product allocations, was, on average, 91%,

varying from 66% to 130%. The maximum category width for Category 3, compared to the average category width of product allocations, was, on average, 94%, varying from 80% to 110%. On the other hand, the maximum category width for Category 1 compared to the shelf width was, on average, 55%, varying from 36% to 75%. The maximum m category width for Category 2 compared to the shelf width was, on average, 41%, varying from 28% to 56%. The maximum category width for Category 3 compared to shelf width was, on average, 36%, varying from 29% to 44%.

Table 4. The usage of the minimum and maximum width after forming product allocations (parameters 5 and 6)

Prod.	Sh.width	Min.vs avg.cat.width(P.5)			Min.vs sh.width(P.5)			Max.vs avg.cat. width(P.6)			Max.vs sh.width(P.6)		
		C1	C2	C3	C1	C2	C3	C1	C2	C3	C1	C2	C3
15	250	80%	47%	–	44%	26%	–	120%	91%	–	66%	50%	–
	375	83%	63%	–	45%	35%	–	112%	102%	–	61%	56%	–
	500	78%	76%	–	43%	42%	–	93%	102%	–	51%	56%	–
	625	111%	47%	–	61%	26%	–	138%	71%	–	75%	38%	–
	750	110%	63%	–	60%	33%	–	125%	80%	–	68%	42%	–
20	250	93%	43%	–	52%	24%	–	129%	80%	–	72%	44%	–
	375	101%	52%	–	56%	29%	–	135%	66%	–	75%	37%	–
	500	108%	50%	–	58%	25%	–	134%	81%	–	72%	40%	–
	625	116%	55%	–	61%	30%	–	135%	69%	–	70%	38%	–
	750	99%	75%	–	51%	43%	–	117%	84%	–	60%	48%	–
25	250	71%	63%	60%	28%	24%	24%	111%	105%	110%	44%	40%	44%
	375	68%	89%	63%	27%	36%	24%	92%	118%	92%	36%	48%	35%
	500	82%	79%	63%	32%	30%	24%	108%	101%	89%	42%	38%	34%
	625	78%	68%	90%	30%	26%	35%	99%	84%	106%	38%	32%	42%
	750	79%	61%	80%	32%	23%	36%	92%	79%	94%	37%	29%	43%
30	250	59%	65%	50%	24%	26%	20%	103%	130%	105%	42%	52%	42%
	375	73%	78%	56%	28%	31%	21%	108%	116%	80%	41%	45%	31%
	500	95%	53%	64%	37%	19%	22%	128%	97%	84%	50%	35%	29%
	625	110%	63%	66%	43%	22%	22%	132%	88%	88%	52%	31%	30%
	750	105%	68%	67%	41%	23%	23%	126%	84%	91%	49%	28%	31%
Minimum		59%	43%	50%	24%	19%	20%	92%	66%	80%	36%	28%	29%
Average		90%	63%	66%	43%	29%	25%	117%	91%	94%	55%	41%	36%
Maximum		116%	89%	90%	61%	43%	36%	138%	130%	110%	75%	56%	44%

The process for setting values for the total profit parameter 17 and the category profit parameters 15 and 16 is presented in Table 5. The minimum category profit (parameter 15) was used for all instances, but the maximum category profit (parameter 16) and the minimum total profit (parameter 17) were used for 25 and 30 product sets only because these are large instances. In every test case, Parameter 15 was used. Thus, in relation to the average category profit of product allocations, the minimum profit for each category is set at 79% to 124% for category 1, 47% to 152% for category 2, and 76% to 164% for category 3, with the average values of 127%, 96%, 111% respectively, for all categories.

The category's maximum profit, as defined by parameter 16, which was only applied to 25 and 30 product set instances, in relation to the average category profit of product allocations, ranges from 112% to 163% for category 1, 94% to 161% for category 2, and 90% to 204% for category 3 with the average values of 132%, 128%, 131% respectively to all categories. The total profit below which the solutions are not taken into consideration was set between 108% and 140% with the average value of 121% for all cases where parameter 17 was applied. The set values were compared to the sum of the average profits of product allocations of all categories.

The quantity of product allocations and solutions produced by heuristics H1 and H2 is shown in

Table 5. The usage of the minimum and maximum category profit (parameters 15 and 16) and the minimum total profit (parameter 17)

Products	Shelf width	Min. vs Avg. Cat. Profit (Par. 15)			Max. vs Avg. Cat. Profit (Par. 16)			Total Profit (Par. 17)
		C1	C2	C3	C1	C2	C3	
10	250	115%	47%	–	–	–	–	–
	375	90%	120%	–	–	–	–	–
	500	81%	86%	–	–	–	–	–
	625	79%	115%	–	–	–	–	–
	750	103%	96%	–	–	–	–	–
15	250	147%	71%	–	–	–	–	–
	375	131%	88%	–	–	–	–	–
	500	116%	110%	–	–	–	–	–
	625	149%	71%	–	–	–	–	–
	750	143%	75%	–	–	–	–	–
20	250	162%	62%	–	–	–	–	–
	375	158%	74%	–	–	–	–	–
	500	164%	77%	–	–	–	–	–
	625	162%	75%	–	–	–	–	–
	750	147%	105%	–	–	–	–	–
25	250	111%	76%	164%	129%	108%	204%	127%
	375	104%	110%	131%	123%	130%	171%	120%
	500	122%	103%	119%	134%	122%	140%	120%
	625	111%	90%	131%	124%	116%	145%	115%
	750	104%	84%	122%	112%	94%	133%	108%
30	250	110%	152%	78%	117%	161%	91%	108%
	375	108%	138%	76%	114%	151%	90%	111%
	500	141%	122%	89%	151%	136%	107%	129%
	625	157%	117%	100%	163%	123%	108%	134%
	750	153%	132%	104%	159%	143%	119%	140%
Minimum		79%	47%	76%	112%	94%	90%	108%
Average		127%	96%	111%	132%	128%	131%	121%
Maximum		164%	152%	164%	163%	161%	204%	140%

Table 6. While the number of product allocations on each shelf is displayed in Table 2, the number of product allocations in each category is displayed in Tables 3 and 4, which also shows the number of product allocations that result from combining allocations from all shelves and categories to create the final solution. Solutions, which are product allocations that meet all constraints, were found among these allocations. However, if too few product allocations are examined, constraint breaches may prohibit a solution from being formed. The primary objective was resolved when the solution with the highest total profit was chosen from the group of options.

Despite employing identical steering parameters, heuristics H1 and H2 provide different numbers of solutions because of the particular rules that are applied within each heuristic. Both strategies used the same number of product allocations.

The average number of product allocations to be examined across all test instances was $7.79 \cdot 10^8$, with a range of $1.73 \cdot 10^5$ to $6.93 \cdot 10^9$. Heuristic H1 produced an average of $2.05 \cdot 10^5$ solutions, while heuristic H2 produced an average of $4.62 \cdot 10^5$ solutions. The solution values generated by both heuristics ranged from $1.00 \cdot 10^0$ to $1.20 \cdot 10^6$ for heuristic H1 and $2.02 \cdot 10^6$ for heuristic H2. Although heuristic H1 produced somewhat fewer solutions than heuristic H1, it also found better-quality solutions (Table 1).

The total number of shelf allocations in a general case is $(r+1)^{PS} = 4^{PS}$.

Table 6. Number of generated product allocations and solutions in heuristics H1, H2

Products	Shelf width	Generated allocations	Solutions H1	Solutions H2
10	250	$4.78 \cdot 10^5$	$3.00 \cdot 10^4$	$3.00 \cdot 10^4$
	375	$2.78 \cdot 10^6$	$2.82 \cdot 10^4$	$2.82 \cdot 10^4$
	500	$8.97 \cdot 10^6$	$1.00 \cdot 10^6$	$1.00 \cdot 10^6$
	625	$1.14 \cdot 10^6$	$1.17 \cdot 10^5$	$1.17 \cdot 10^5$
	750	$1.73 \cdot 10^5$	$1.68 \cdot 10^5$	$1.68 \cdot 10^5$
15	250	$2.10 \cdot 10^8$	$2.41 \cdot 10^2$	$2.93 \cdot 10^5$
	375	$6.93 \cdot 10^9$	$1.97 \cdot 10^4$	$5.00 \cdot 10^0$
	500	$6.46 \cdot 10^6$	$4.82 \cdot 10^4$	$9.36 \cdot 10^4$
	625	$3.98 \cdot 10^8$	$3.49 \cdot 10^4$	$7.44 \cdot 10^4$
	750	$1.86 \cdot 10^9$	$6.89 \cdot 10^2$	$2.24 \cdot 10^4$
20	250	$6.17 \cdot 10^7$	$5.31 \cdot 10^2$	$3.65 \cdot 10^3$
	375	$3.89 \cdot 10^6$	$9.10 \cdot 10^1$	$9.10 \cdot 10^1$
	500	$1.55 \cdot 10^9$	$1.56 \cdot 10^3$	$3.12 \cdot 10^5$
	625	$7.22 \cdot 10^8$	$1.02 \cdot 10^5$	$2.02 \cdot 10^6$
	750	$1.08 \cdot 10^8$	$4.32 \cdot 10^5$	$1.27 \cdot 10^6$
25	250	$2.39 \cdot 10^8$	$3.37 \cdot 10^4$	$5.39 \cdot 10^4$
	375	$3.52 \cdot 10^8$	$1.02 \cdot 10^6$	$1.99 \cdot 10^6$
	500	$2.10 \cdot 10^8$	$3.77 \cdot 10^5$	$8.83 \cdot 10^5$
	625	$1.18 \cdot 10^8$	$1.20 \cdot 10^6$	$1.84 \cdot 10^6$
	750	$4.17 \cdot 10^7$	$4.65 \cdot 10^5$	$8.20 \cdot 10^5$
30	250	$1.20 \cdot 10^8$	$1.00 \cdot 10^0$	$1.00 \cdot 10^0$
	375	$2.13 \cdot 10^9$	$2.49 \cdot 10^3$	$4.42 \cdot 10^5$
	500	$3.49 \cdot 10^9$	$1.08 \cdot 10^3$	$1.30 \cdot 10^4$
	625	$5.70 \cdot 10^8$	$2.83 \cdot 10^2$	$1.36 \cdot 10^3$
	750	$3.49 \cdot 10^8$	$3.38 \cdot 10^4$	$8.08 \cdot 10^4$
Minimum		$1.73 \cdot 10^5$	$1.00 \cdot 10^0$	$1.00 \cdot 10^0$
Average		$7.79 \cdot 10^8$	$2.05 \cdot 10^5$	$4.62 \cdot 10^5$
Maximum		$6.93 \cdot 10^9$	$1.20 \cdot 10^6$	$2.02 \cdot 10^6$

The number 4 represents the possible allocations of a product: (1) not placed on the shelf, (2) placed on the shelf in a front orientation, (3) placed on the shelf in a side orientation, and (4) placed on the shelf in a top orientation. All products can be simultaneously placed on all shelves. The total number of product allocations for each set of products in a general case is represented by this calculation $\prod_{j=1}^{P}(f_j^{max} - f_j^{min} + 1)^S$.

The number of potential shelf and product allocations in the general scenario, which represents the full solution space determined using the aforementioned formulae, is shown in Table 7. However, there is a notable disparity in the number of shelf and product allocations created by heuristics H1 and H2, as indicated in Table 6.

For instance, in the largest case, the total number of possible product allocations across all shelf widths is $1.33 \cdot 10^{156}$. However, heuristics H1 generated only 91 solutions, and heuristics H2 required only 100 solutions to sometimes achieve optimal solutions. Both heuristics could achieve near-optimal solutions, obtaining only 1 solution for 30 products set on 250 cm shelves.

This demonstrates the significant validity and practicality of using heuristic rules with steering parameters. These intuitive guidelines are highly rational and effective for guiding decision-making processes. They serve as robust tools for problem-solving, leveraging practical insights and logical reasoning to navigate complex scenarios effectively. They are strong problem-solving instruments that use logical thinking and real-world insights to successfully negotiate challenging situations.

Table 7. Number of all possible shelf and product allocations in the general case

Products	Number of shelf allocations	Number of product allocations
10	$1.21 \cdot 10^{24}$	$1.10 \cdot 10^{52}$
15	$1.33 \cdot 10^{36}$	$1.15 \cdot 10^{78}$
20	$1.46 \cdot 10^{48}$	$1.21 \cdot 10^{104}$
25	$1.61 \cdot 10^{60}$	$1.27 \cdot 10^{130}$
30	$1.77 \cdot 10^{72}$	$1.33 \cdot 10^{156}$

The proposed flower-cutting heuristics offer several significant advantages:

- Efficiency: The solution space is not exhaustively explored, reducing the need for extensive computational power and time. By focusing on more promising areas of the search space, the approach optimizes the use of resources.
- Parameter-driven reduction: A carefully selected set of parameters narrows the solution space, ensuring that attention is given only to potentially profitable shelf and product allocations. This targeted approach improves the efficiency of solution formation by eliminating irrelevant possibilities.
- Scalability: As problem instances grow larger, the heuristic adapts by introducing additional parameters that effectively manage the increased complexity. This ensures the method remains applicable across a wide range of problem sizes without compromising performance.
- Deterministic approach: Unlike stochastic methods such as genetic algorithms or simulated annealing, which depend on randomness, this heuristic employs a deterministic strategy. It ensures consistency in its solutions, offering predictable and repeatable results.
- Termination condition: In contrast to many algorithms that iterate until a predetermined number of iterations is reached, the flower-cutting heuristic terminates once a sufficient number of high-quality solutions are found without relying on a fixed iteration count. This flexible termination mechanism is based on the complex integration of various tuning parameters, making the approach more adaptive and efficient in finding optimal solutions.
- Improved decision-making: The heuristic allows for a more nuanced exploration of the solution space by considering only high-quality solutions at each step. This reduces unnecessary calculations and focuses efforts on refining the best possibilities, leading to faster convergence on optimal outcomes.
- Adaptability to dynamic environments: The heuristic can be adjusted in real time, allowing it to react to changes in problem constraints or new data. This makes it well-suited for environments where conditions evolve, such as in retail management or retail information systems.
- Enhanced robustness: The algorithm's reliance on carefully tuned parameters rather than randomness makes it more robust across a variety of problem settings. It is less likely to be affected by outliers or noise in the data, resulting in more stable and reliable performance.

Key features of the proposed flower-cutting heuristics include:

- Category-based application: These heuristics operate more efficiently when applied to groups of items organized into categories rather than handling each item individually. By considering the broader context of related items, it optimizes the allocation process, leading to more effective and coherent solutions.
- Pre-solution investigation: Before selecting the best parameters, the heuristic conducts a comprehensive exploration of the solution space. This careful examination ensures that the chosen parameters are well-suited to the problem at hand, increasing the likelihood of achieving optimal results.
- Iterative parameter tuning: The process of finding the parameters that are as best as possible often involves multiple iterations. This iterative approach enables the heuristic to refine and adjust its parameters over time, leading to more precise and accurate outcomes as it hones in on the best configuration.
- Flexible termination condition: Unlike traditional heuristics that rely on a fixed termination criterion, the flower-cutting heuristic determines when to stop based on a meaningful combination of all tuning parameters. This approach ensures that the solution process concludes when a satisfactory set of solutions is found, allowing for more dynamic and context-sensitive termination.

These experiments yielded essential empirical data that helped identify both the advantages and drawbacks of the proposed heuristics in addressing the SSAP. By comparing different approaches, the analysis offered significant perspectives for both researchers and professionals involved in retail optimization. It further contributed to refining the process of selecting the most appropriate solution strategies, ensuring alignment with the unique challenges, constraints, and objectives associated with specific problems in the field. This broader understanding equips decision-makers with a more informed basis for method selection, fostering enhanced efficiency in practical applications.

6. Conclusions

Proper distribution of products not only enhances operational efficiency but also ensures that the storage system can accommodate a wide variety of items without overburdening specific shelves or sections. Consequently, products in the vertical category should be distributed evenly along the shelves to maintain balance and accessibility.

A well-organized retail store with evenly distributed products, clear categorization, and strategic use of rack dividers not only enhances operational efficiency but also improves the accuracy and speed of order fulfillment. This comprehensive approach to retail store organization ensures that products are stored safely, accessed easily, and managed effectively, ultimately leading to better service and satisfaction for customers.

In this research, we model the retail store SSAP with simultaneous vertical and horizontal product categorization on the racks in which the following main features exist: the products required to be stored on different shelves, the products required to be stored on the same shelf in the same category but not side by side because of odour, chemical reactions or the possibility of confusion by the picker or customer, the flexible or strict border between vertical categories, specific shelf levels for brand and general assortment products.

Retail store staff must also consider the necessity of separate storage for certain products to prevent cross-contamination or adverse interactions. This includes storing allergenic items away from non-allergenic ones and ensuring that products with strong odours do not compromise the quality of odour-sensitive items. By implementing these meticulous organizational practices, retail store managers can enhance overall operational efficiency, streamline inventory control, and uphold the safety and quality of the stored goods.

In this research, we proposed flower-cutting heuristics for the retail shelf space allocation problem on store equipment with simultaneous vertical and horizontal product categorization on the racks applicable to the retail store. The gardener, as the personification of the heuristics, tries to cut the biggest flower in the garden without cutting the flowers in the whole garden and without making the selected clearings bald in the garden. The flower represents the solution; the garden is the solution space, and the selection by the gardener is the reduced solution space. The gardener does not cut growing nearby flowers as in reality. The criterion function is the profit maximization of the solution. The gardener's basket, to which the flowers could be put, restricts the number of solutions that could be generated and compared based on the available time and hardware resources. Therefore, the four groups of parameters that reduce the solution space representing the areas with the biggest flowers are developed: parameters related to clearings in the garden, the length of the gardener's movement along the clearings, the interval between cut flowers, and the flower size (profitability). There were 17 tuning parameters implemented. They are:

- Parameters of flower clearing creation: the maximum category width while forming product allocations (parameter 1), the minimum number of products that can be placed on the shelf while forming product allocations (parameter 2), the maximum number of products that can be placed on the shelf while forming product allocations (parameter 3), the set of profitable groups of products to be placed on the shelf (parameter 4).
- Parameters of moving along the selected flower clearings: the minimum and maximum category width after forming product allocations (parameters 5 and 6), if the grouping option (for each total width, only 1 product allocation with the maximum total profit) is used (parameter 7), the maximum number of product allocations on the shelf (parameter 8) with two sorting rules inside ((1) category width $\uparrow$, category profit $\downarrow$, (2) category profit $\downarrow$, category width $\uparrow$), if the grouping option (for each total profit and profit ratio, take only 1 product allocation with the minimum total width) is used (parameter 9), the maximum number of product allocations of the category (parameter 10) with two sorting rules inside ((1) profit $\downarrow$, profit ratio $\downarrow$, (2) profit ratio $\downarrow$, profit $\downarrow$).
- Parameters of the interval between cut flowers on the selected clearings: the interval of taking the product allocations on the shelf after taking all product allocations according to parameter 8 (parameter 11), the maximum number of product allocations on the shelf created with the interval parameter 11, the sorting rule is the same as in parameter 8 (parameter 12), the interval of taking the product allocations of the category after taking all product allocations according to the parameter 10 (parameter 13), the maximum number of product allocations of the category created with the interval parameter 13, the sorting rule is the same as in parameter 10 (parameter 14).
- Parameters of the flowers to be cut: the minimum and maximum profit for each category (parameters 15 and 16), and the minimum total profit (parameter 17).

The goal of continuously improving tuning parameters is very clear: to increase the solutions' profitability and efficacy. The method converges to optimal solutions with fewer iterations by continuously optimizing the input parameters. This guarantees that the solutions are solid and appropriate for the given situation while also quickening the optimization process.

In the current research, we report only the best-selected parameter set after the tuning process. By presenting only the best-selected parameter set, we provide a clear and concise overview of the most successful configuration. This highlights the optimal conditions under which the heuristics perform most effectively, offering valuable insights for future applications and research in shelf space allocation problems.

Our focus on the best tuning parameters allows for a streamlined and impactful presentation of results, demonstrating the practical viability and robustness of the proposed heuristics under almost optimal conditions. This ensures that the conclusions drawn from the research are based on the most reliable and high-performing configurations available.

The following are the stopping criteria. The gardener's basket size restricts the gardener from picking all or random flowers, but he cuts the flowers that are definitely profitable. After that, the goal is to find the biggest flower that the gardener cut. The number of solutions put into the basket is the specified number of iterations. It is possible to configure the algorithm to stop after a predetermined number of iterations.

This guarantees that the computational effort stays within realistic bounds, avoiding wasteful resource usage while permitting in-depth solution space exploration and optimization.

In this research, 25 testing data sets were used to evaluate the performance of each heuristic, and the results were compared to the optimal solutions found by the CPLEX solver. Heuristic H1 and heuristic H2 found approximately the same solutions, achieving an average profit ratio of 99.85% (Heuristic H1) and 99.86% (Heuristic H2), both varying from 98.58% to 100.00%. Both heuristics were able to find optimal solutions for 17 of the 25 test cases, but not for the same ones.

The maximum number of product allocations checked by both heuristics was $6.93 \cdot 10^9$, with the total number of solutions generated by heuristics H1 $1.20 \cdot 10^6$ and $2.02 \cdot 10^6$ generated by heuristics H2. In comparison, the total number of shelf and product allocations in the entire solution space was $1.77 \cdot 10^{72}$ and $1.33 \cdot 10^{156}$, respectively. This significant reduction in the solution space demonstrates the power of heuristics to limit the number of possible solutions that need to be explored. The adaptability of the proposed methodology further supports its efficiency, as it allows for a thorough and systematic examination of viable solutions. This reduces computational complexity while maintaining a high level of solution quality, highlighting the practicality and value of using such heuristics for retail store optimization problems. The results provide useful insights into the strengths of these heuristics, suggesting their potential applicability for solving large-scale, complex optimization problems effectively.

Additionally, this demonstrates how the principles underlying the proposed heuristics enable a substantial reduction in the solution space without compromising the calibre of the output. For most cases, the heuristics' solution times (which ranged from 0.05 to 5.60 minutes for heuristics H1 and from 0.05 to 7.01 minutes for heuristics H2) were sufficient.

The flower-cutting heuristic becomes broadly applicable when:

- Solutions are composed of individual feasible elements (allocations, assignments, segments, tasks).

- The solution space is prohibitively large and requires parameter-driven reduction before evaluation.
- Feasibility depends on multiple structural constraints, making direct enumeration inefficient.
- Decision components can be ranked using problem-specific performance metrics (profit, efficiency, waste, cost, time).
- Iterative constructive assembly of solutions is feasible, similar to constructive heuristics in scheduling, routing, and layout design.

Due to these properties, the heuristic provides an adaptable framework that can be extended beyond retail shelf optimization. By redefining flowers, feasibility rules, reduction parameters, and construction criteria, the algorithm can be systematically tailored to other large-scale combinatorial problems in Industrial Engineering.

Future research should focus on improving the developed heuristics. Continuous monitoring of the algorithm's performance in real-world scenarios is suggested to identify any degradation or changes in SSAP characteristics. Regularly updating and refining the heuristics based on new insights or retail data is advised. Special attention should be given to collecting and incorporating feedback from retailers to address practical issues and improve usability, as well as adjusting the developed parameters of solution space reduction or developing new parameters.

In this research, the basket size representing the number of solutions is a single termination criterion. Achieving satisfactory solution quality should be implemented as other termination criteria. This kind of stopping criterion is the attainment of a solution quality that meets or exceeds a predefined threshold. This ensures that the solutions generated are not only feasible but also of high quality, fulfilling the desired objectives of the problem-solving process. This criterion helps maintain a balance between computational efficiency and the need for high-quality outcomes. The results of the study have implications for retailers who sell goods through retail chains.

Acknowledgement

The authors are grateful to two anonymous reviewers for their valuable comments and suggestions made on the previous draft of this manuscript.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors

References

[1] Bai, R. An investigation of novel approaches for optimising retail shelf space allocation. *Control* (September 2010), 217.

[2] Bai, R., and Kendall, G. An investigation of automated planograms using a simulated annealing based hyper-heuristic. In *Operations Research/Computer Science Interfaces Series*, vol. 32. 2005, pp. 87–108.

[3] Bai, R., van Woensel, T., Kendall, G., and Burke, E. A new model and a hyper-heuristic approach for two-dimensional shelf space allocation. *4OR 11*, 1 (2013), 31–55.

[4] Binguler, A., Bulkan, S., and Agaoğlu, M. A heuristic approach for shelf space allocation problem. *Journal of Military and Information Science 4*, 1 (2015), 38.

[5] Burke, E., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society* (2013).

[6] Chakhlevitch, K., and Cowling, P. Hyperheuristics: Recent developments. In *Studies in Computational Intelligence*. 2008.

[7] Du, K., and Swamy, M. *Search and Optimization by Metaheuristics: Techniques and Algorithms Inspired by Nature*. 2016.

[8] Gajjar, H., and Adil, G. A piecewise linearization for retail shelf space allocation problem and a local search heuristic. *Annals of Operations Research 179*, 1 (2010), 149–167.

[9] Gajjar, H., and Adil, G. A dynamic programming heuristic for retail shelf space allocation problem. *Asia-Pacific Journal of Operational Research 28*, 2 (2011), 183–199.

[10] Gajjar, H., and Adil, G. Heuristics for retail shelf space allocation problem with linear profit function. *International Journal of Retail and Distribution Management 39*, 2 (2011), 144–155.

[11] Hansen, J., Raut, S., and Swami, S. Retail shelf allocation: A comparative analysis of heuristic and meta-heuristic approaches. *Journal of Retailing 86*, 1 (2010), 94–105.

[12] Hariga, M., Al-Ahmari, A., and Mohamed, A. A joint optimisation model for inventory replenishment, product assortment, shelf space and display area allocation decisions. *European Journal of Operational Research 181*, 1 (2007), 239–251.

[13] Hwang, H., Choi, B., and Lee, M. A model for shelf space allocation and inventory control considering location and inventory level effects on demand. *International Journal of Production Economics 97*, 2 (2005), 185–195.

[14] Kumar, M., and Kulkarni, A. Socio-inspired optimization metaheuristics: A review. In *Studies in Computational Intelligence*, vol. 828. 2019, pp. 241–265.

[15] Lim, A., Rodrigues, B., and Zhang, X. Metaheuristics with local search techniques for retail shelf-space optimization. *Management Science 50*, 1 (2004), 117–131.

[16] Lim, A., Zhang, Q., and Rodrigues, B. A heuristic for shelf space decision support in the retail industry. In *AMCIS 2002 Proceedings* (2002), p. 30.

[17] Raut, S., Swami, S., and Moholkar, M. Heuristic and meta-heuristic approaches for multi-period shelf-space optimization: The case of motion picture retailing. *Journal of the Operational Research Society 60*, 10 (2009), 1335–1348.

[18] Yang, M. Efficient algorithm to allocate shelf space. *European Journal of Operational Research 131*, 1 (2001), 107–118.

[19] Yang, M., and Chen, W. Study on shelf space allocation and management. *International Journal of Production Economics 60* (1999), 309–317.

[20] Yu, V., Maglasang, R., and Tsao, Y. A reduced variable neighborhood search-based hyperheuristic for the shelf space allocation problem. *Computers and Industrial Engineering 143* (2020).