OPSEARCH
https://doi.org/10.1007/s12597-023-00636-1

APPLICATION ARTICLE

Heuristics for the shelf space allocation problem

Kateryna Czerniachowska[1] · **Krzysztof Michalak**[2] · **Marcin Hernes**[3]

Accepted: 18 March 2023

Abstract

The retailers' goals to maximize the profit of the products in stores are realized on the planogram shelves. In this paper, we investigated a practical shelf space allocation model with a visible horizontal and vertical grouping of products into categories, which takes into account the number of facings, capping and nesting of a product. The result is four groups of constraints, such as shelf constraints, product constraints, multi-shelves constraints, and category constraints that are used in the model. We proposed 6 heuristics to solve the planogram profit maximization problem. The developed techniques on which heuristics are based may be applied to other category of management shelf space allocation problems because all of them share the same nature of the problem, i.e., the initial step of creating the allocation of products on the shelf and steps in which shelves are combined. Experiments were based on data sets generated according to contemporary real retail conditions. The efficiency of the designed heuristics has been estimated using the CPLEX solver.

Keywords Heuristics · Retailing · Revenue management · Merchandising · Shelf space allocation

JEL Classification C61 · C63 · C88 · L81

✉ Kateryna Czerniachowska
Kateryna.Czerniachowska@ue.wroc.pl

Krzysztof Michalak
Krzysztof.Michalak@ue.wroc.pl

Marcin Hernes
Marcin.Hernes@ue.wroc.pl

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

[2] Department of Information Technologies, Wroclaw University of Economics and Business, Wroclaw, Poland

[3] Department of Process Management, Center for Intelligent Management Systems, Wroclaw University of Economics and Business, Wrocław, Poland

Published online: 07 April 2023

1 Introduction

For successful merchandising decisions in traditional space management, a specific tool called a planogram is widely used by retailers. It illustrates the layout of the products on the fixtures. Creating effective visual plans is very important for supporting the retailers' decisions related to shelf space allocation. It allows for maximizing the store profits [5, 30] and, consequently, more effective functioning of the entire organization. The retail shelf space allocation problem (SSAP) is well-known in the literature. The examination of software applications in assortment and shelf space management, as well as quantitative models, has been presented by Hübner and Kuhn [27].

In this research, we formulate the SSAP according to the merchandising rules that specify vertical and horizontal groups of products and the division of the products based on sales potentials. Merchandising generalizes any techniques or practices of visual representation of products in the retail store. Dependencies between the customers' buying behaviour, product categorization, shelf location, and visual representation of them can be found in the research of Anic et al. [1], Desrochers and Nelson [17], Elbers [19].

The research by Gabrielli and Cavazza [22] deepened the knowledge about placing products at the end of an aisle. They studied brand appreciation on low, medium, and high levels. They concluded that the location of the product to be exposed is very powerful for brand recognition. Bianchi-Aguiar et al. [6] studied the categorization of product families and allocating them into blocks as well as a multi-level hierarchical structure. They proposed a method of decomposing the main problem into sub-problems and applying specific methods to solve them. Düsterhöft et al. [18] studied various shelf segments of different attractiveness with the aim of appropriately assigning available shelf space between lots of brands, considering optimum replenishment decisions. The concept of shelf segments of flexible size (named "virtual") was defined by Czerniachowska [10], Czerniachowska and Hernes [12] and Czerniachowska et al. [15, 16]. Several types of segments with regard to product types included local or regional and convenience or complementary products. Other types of segments allocated near the aisles included products, taking into account store customer traffic flow. The research by Czerniachowska [10] most widely explains the evidence regarding shelf levels and shelf segment usage in the SSAP models and gives a wide variety of solution tricks specifically to this problem. Metaheuristics [10, 15, 16] and hyper-heuristics [12] were adopted to solve the SSAP with shelf segments.

The objective of this paper is to present a real retail shelf space allocation model and to develop efficient heuristic algorithms that can allow for obtaining good practical results in a very short time, the ideas of which can also be used in other shelf space allocation models. The mathematical model, which was first presented in Czerniachowska and Hernes [13] includes vertical product allocation constraints as well as horizontal categories grouping, which are very important in visual merchandising.

One of the key limitations of the SSAP literature is not taking into account merchandising rules based on the frequency of product movement and product price. In the model by Czerniachowska and Hernes [13], a method of dividing products and allocating them to different vertical shelf levels based on their sales potential is proposed. Valenzuela and Raghubir [36] concluded that cheaper brands should be located on the bottom shelves, and the best place for luxury brands is on the top shelves. Therefore, in the model, Czerniachowska and Hernes [13] differentiated products based on their sales potential in order to locate them correctly on the bottom or top shelves.

In many studies, different approaches have been developed for shelf space allocation. Because of the NP-hard nature of the shelf space allocation problem, heuristic algorithms are needed for the solution of large instances of real-world problems. On this basis, many approaches have been proposed, including:

- Simulated annealing [2, 4, 7, 12, 37],
- Genetic algorithms [8–11, 15, 16, 20, 21, 24–26, 28, 29, 31–35],
- Gradient search heuristic [3, 4, 28],
- Dynamic programming [10, 14, 23].

There are two concepts taken into account in this paper: capping and nesting. Capping means an allocation of a product on top of another product with a rotation of the product on the top in order to push in more products on the shelf if there is no space to duplicate a product row one more time on the first product orientation. Nesting means an allocation of a product into another product, such as a basket or a plate.

2 Paper contributions

We develop heuristics and test them on different practical problem instances of different sizes and compare our solution with the CPLEX solver. The proposed techniques may be used to allow retailers to make category management decisions faster and to obtain higher profits. The proposed methods construct solutions incrementally, which has the following advantages:

- Only feasible candidate solutions are considered. Hence, no repair operator is required as opposed to metaheuristics (e.g. [8–11, 15, 16, 20, 21, 24–26, 28, 29, 31–35]), such as evolutionary algorithms, in which random local search, crossover, and mutation operations can turn a correct solution into an infeasible one.
- If constraints cannot be satisfied, the user can be notified to change the product grouping method, increase the minimum category size and tolerance, reassign sales potentials, etc.

- Each step filters out infeasible candidate solutions, so in the following steps, the search space is smaller than in the case of non-deterministic algorithms, such as evolutionary algorithms.
- The proposed heuristics are better than the neighbourhood search approaches (e.g. [4], low-level heuristics (e.g. [2] or simulated annealing (e.g. [2, 4, 7, 12, 37] because they do not need an initial solution to start with or neighbourhood preferences for constructing a possible solution.
- The proposed approach presents considerable competitiveness over metaheuristics and hyper-heuristics proposed by other researchers, as in the proposed approach, the total number of product allocations could be calculated and compared to the general case. This is extremely valuable in case of time-saving while processing large datasets.

In the experiments, the proposed method was tested on data sets that were generated on the basis of real-life data. In the tests, the heuristics found the solution for 11 SSAP instances for which CPLEX was not able to find one, which motivates the use of heuristics and suggests further development towards hybrid methods.

The paper is organized as follows. Problem formulation and its mathematical model are presented in Sect. 3. A detailed description of proposed heuristics is presented in Sect. 4. Next, in Sect. 5, the results of computational experiments are presented. The article is concluded in Sect. 6.

3 Statement of the problem

3.1 Creation of a planogram

The Shelf Space Allocation Problem (SSAP) consists in constructing a planogram that is divided horizontally into categories and vertically into subcategories. The problem can be formulated as follows [13]: there is a given number of products P that must be displayed on S shelves of a planogram. The products $1, ..., P$ are assigned to K product categories. On the planogram, there are also initially assigned spaces for the K product categories, i.e., the minimum possible size of the category allowing it to be visually attractive for the customers. The problem is to define the appropriate shelf space for each product category K with regard to the quantity of each product with the objective of maximizing retailers' profit.

Products $1, ..., P$ are divided into K categories based on their types (product families) and into G subcategories based on their sales potential. Each category is vertical, i.e., the products are placed on all shelves i ($i = 1, ..., S$) within one category k ($k = 1, ..., K$). In this paper, we assume that more general categories (such as milk) can be extended to the vertical spaces allocated to more specific categories (such as yoghurt and cheese). This follows the practice observed in real-life applications.

The sales potential subcategory g ($g = 1, ..., G$) is horizontal, i. e. the products are placed on one shelf i ($i = 1, ..., S$) within one subcategory g ($g = 1, ..., G$), but the same sales potential subcategory g ($g = 1, ..., G$) exists in all categories k ($k = 1, ..., K$). The higher the sales potential subcategory of the product, the more expensive it is and,

Table 1 Categories and sales potential subcategory allocation rules

	Category	A (milk)			B (yoghurts)			C (cheese)		
Category	Subcategory	10	20	30	10	20	30	10	20	30
A (milk)	10 (cheap products)	●	●	●	●	●	●	●	●	●
	20 (more expensive products)		●	●		●	●		●	●
	30 (brand products)			●			●			●
B (yoghurts)	10 (cheap products)				●	●	●	●	●	●
	20 (more expensive products)					●	●		●	●
	30 (brand products)						●			●
C (cheese)	10 (cheap products)							●	●	●
	20 (more expensive products)								●	●
	30 (brand products)									●

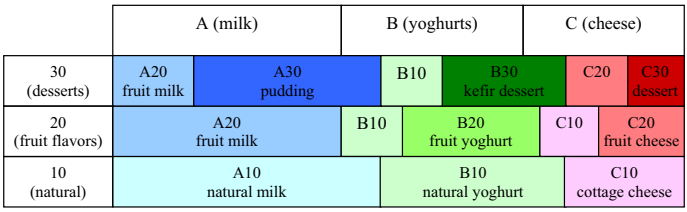


Fig. 1 Possible categories and sales potential subcategory allocation with the flexible border between vertical categories

therefore, the better level position on the shelf it receives. Obviously, the eye-level shelf for branded or expensive products is the best option.

The products with the lowest sales potential are located on the lowest shelf but can also be located on higher shelves. On the other hand, the products with the highest sales potential are located on the highest shelf (at eye level) and can't be located on other shelves. Products of the same category k ($k = 1, ..., K$) must be located together. This means that the category can't be split or interrupted with products from another category.

3.2 Planogram rules for categories and sales potential subcategories

Table 1 shows the rules for allocating products on shelves. Products from category A and subcategory 10 can be allocated to the shelves for categories A, B, C and subcategories 10, 20, 30. Products from category C and subcategory 30 can't be allocated to the shelves for other categories and subcategories. Another example, B20 can be placed in B20, B30, C20, C30. Sales potentials are assigned with the interval 10 for practical reasons. For example, if in the general assortment set, products are assigned to 10 and 20 sales subcategories, but one seasonal or promotional shelf is added only for 2-week intervals, it is easier to assign seasonal or promo products to 15 sales potential without the reassignment of the general assortment. Figures 1 and 2 present possible category sizes on the shelves with different sales potentials. The borderline between the categories may be flexible (Fig. 1) or strict (Fig. 2).

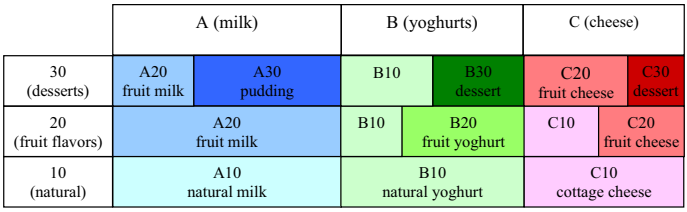


Fig. 2 Possible categories and sales potential subcategory allocation with the strict border between vertical categories

3.3 Explanation of other parameters used in the model

In the remaining part of the paper, the following notation is used. Due to a large number of problem parameters, in the given paper, subscripts are used for variable indexes. Superscripts are set for a variable's clarification and mustn't be interpreted as indexes.

A parameter defining the minimum category size is the m_k per cent of the shelf length, i.e., each category must exist on the planogram, and the products in this category must be noticeable. The coefficient t_k per cent of the shelf length limits the maximum category size tolerance between shelves in the category. This parameter is used to create visually attractive vertical category shapes with a well-defined border between the categories.

Each shelf i $(i = 1, ..., S)$ has a different length capacity s_i^l, shelf height s_i^h, shelf depth capacity s_i^d, product weight bound s_i^b, and shelf sales potential subcategory s_i^g. In the analyzed model, all shelves are assigned to each of the categories; a subcategory g $(g = 1, ..., G)$ is assigned to the shelf, and products from each subcategory g $(g = 1, ..., G)$ are also allocated to a certain category k $(k = 1, ..., K)$ (Figs. 1, 2).

In the given problem, the product j has width p_j^w, height p_j^h, depth p_j^d, and weight p_j^b. Out-of-stock situations are avoided by the retailer with the control of the supply limit parameter of the product p_j^s, which indicates the maximum possible number of product items displayed. The product item profit p_j^u characterizes the profit gained from the product if it is displayed. The category of the product is expressed by p_j^k, and its sales potential subcategory is p_j^g.

Generally, each product is placed on the shelf in the front orientation, but for some products, the secondary side orientation (90 degrees) is also available. Binary parameters of a product $p_j^{o_1}$ and $p_j^{o_2}$ reflect if the front and side orientation are available for the product j. For the front orientation,$p_j^{o_1}$ we take product width p_j^w as the width of the product on the shelf and depth p_j^d as the depth of the product on the shelf. For side orientation,$p_j^{o_2}$ we take its depth p_j^d as the width of the product on the shelf and width p_j^w as the depth of the product on the shelf.

$$p_j^{o_1} = \begin{cases} 1, \text{ if front orientation is available for the product } j \\ 0, \text{ otherwise} \end{cases}, j = 1, ..., P.$$

$$p_j^{o_2} = \begin{cases} 1, \text{ if side orientation is available for the product } j \\ 0, \text{ otherwise} \end{cases}, j = 1, ..., P.$$

To ensure a substitution effect, the products are grouped into clusters p_j^l. If the product is out-of-stock or delisted from the assortment, the customer can choose a similar product (with the same characteristics, functions, and tastes) from another brand. Moreover, when one product is delisted, to prevent the loss of profit, retailers ensure the availability of another substitutive product on the shelf. Cluster parameter p_j^l means the similar substitution product group. Products from the same cluster must be placed together on the same shelf next to one another.

3.4 Product item representation

In this paper, we take into account not only the facings of the products but also cappings and nestings, which is an extension with respect to models typically used in the literature, which only consider facings. Capping means that products, such as rectangular packages, can be put on top of each other in a rotated orientation. Nesting means that products, such as buckets, can be put inside each other. Depending on the product package's physical characteristics, cappings can be placed above the facings of the product rotated 90 degrees (e.g., tea boxes), and nesting is placed inside the facings (e.g., bowls or plates). To describe the product nesting possibilities, the nesting coefficient $p_j^n \left(p_j^n < 1\right)$ signifies additional height for one nested item (e.g., the additional bowl height if bowls are placed inside each other). Regular products that can't be nested $p_j^n = 0$ because products that can't be nested are represented as another row of facings.

The lower $f_j^{\min}$ and upper $f_j^{\max}$ bounds of facings of the products that should be placed on the shelves are defined by the retailer at the store. Furthermore, the lower $c_j^{\min}$ and upper $c_j^{\max}$ bounds of side cappings per position can also be restricted for each product. The parameter $c_j^{\max}$ means the maximum number of cappings that can actually be placed above the facings sequence of items without destroying them because of the weight of the capping items. The mandatory number of sequences of facings in one capped group is $\left\lceil p_j^h / p_j^w \right\rceil$ for products with front orientation and $\left\lceil p_j^h / p_j^d \right\rceil$ for products with secondary side orientation, which describes the possibility of holding up the cappings above. Let's consider an example. If more than $c_j^{\max}$ tea boxes are put above the tea box facings sequence, the boxes on the bottom level may be broken or destroyed, or tea boxes on the top level could fall off the shelf.

The lower $n_j^{\min}$ and upper $n_j^{\max}$ bounds of the side nestings per position can also be restricted for each product. The parameter $n_j^{\max}$ indicates the maximum number of nestings that can actually be placed above one facing item without destroying it. Let's consider another example. If more than $n_j^{\max}$ the bowls are put inside the lower bowl on the shelf, the lowest bowl may be broken or destroyed, or the bowls on the top level could fall off the shelf. The multi-shelf product placement possibility is

Fig. 3 Cappings allocation

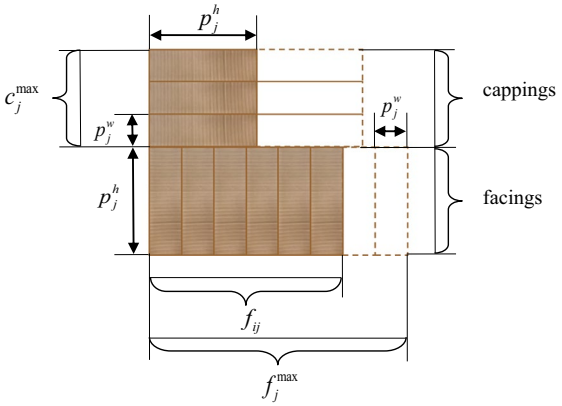
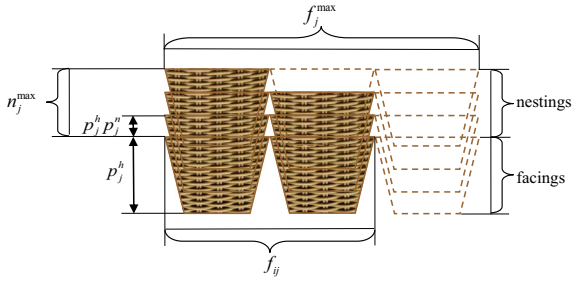


Fig. 4 Nestings allocation



defined by the minimum $s_j^{\min}$ and the maximum $s_j^{\max}$ number of shelves on which it can be placed. Based on the introductory definitions of facings, cappings and nestings above, the total number of items of the product is the sum of facings f_{ij}, cappings c_{ij} and nesting numbers n_{ij} of a product. All of the aforementioned parameters (f_{ij},c_{ij},n_{ij}) represent the number of product items. Figures 3 and 4 represent the cappings and nestings allocation methods.

3.5 The idea of a solution to the problem

In this paper, the number of facings in the vertical dimension is not considered. It is assumed that shelf height and product supply limit is given only for one topmost sequence of facings with capping or nesting on it, i.e., only one horizontal sequence of facings is analyzed. The same case goes with the depth of the products and the depth of the shelf, which is also given for one front sequence of facings. If the depth of the investigated part of the shelf is exceeded, and if the product could be placed on both front and side orientations, it is possible to rotate it so that it could fit the depth limitations.

Based on the above-mentioned suppositions, in order to solve the problem, the task is to calculate the number of facings f_{ij}, cappings c_{ij} and nestings n_{ij} of a product j which is allowed to be placed on the shelf i with regard to different constraints which, for clearness, can be grouped into 4 classes: shelf constraints, product constraints, multi-shelves constraints, and category constraints, in order to maximize the total retailer's profit.

3.6 Problem formulation

Given the assumptions stated above, the shelf space management problem can be summarized as a following non-linear integer programming model. The model was first presented by Czerniachowska and Hernes [13].

$$\max \sum_{j=1}^{P} \sum_{i=1}^{S} x_{ij} p_j^u (f_{ij} + c_{ij} + n_{ij}) \tag{1}$$

$$x_{ij} = \left\{ \begin{cases} 1, & \text{if product } j \text{ is put to the shelf } i \\ 0, & \text{otherwise} \end{cases} \right\}, j = 1, \ldots, P, \ i = 1, \ldots, S.$$

f_{ij}—the number of facings of the product j ($j = 1, ..., P$) on the shelf i ($i = 1, ..., S$).c_{ij} —the number of cappings of the product j ($j = 1, ..., P$) on the shelf i ($i = 1, ..., S$) .n_{ij}—the number of nestings of the product j ($j = 1, ..., P$) on the shelf i ($i = 1, ..., S$).

$$y_{ij}^{o_1} = \left\{ \begin{array}{l} 1, \text{ if product } j \text{ is put to the shelf } i \text{ on front orientation} \\ 0, \text{ otherwise} \end{array} \right\}, \ j = 1, ..., P, i = 1, ..., S.$$

$$y_{ij}^{o_2} = \left\{ \begin{array}{l} 1, \text{ if product } j \text{ is put to the shelf } i \text{ on side orientation} \\ 0, \text{ otherwise} \end{array} \right\}, \ j = 1, \ldots, P, \ i = 1, \ldots, S.$$

The formula (1) multiplies the profit from one product item p_j^u by the number of the product items ($f_{ij} + c_{ij} + n_{ij}$), thereby representing the total profit gained by the retailer.

3.7 Shelf constraints

$$\sum_{j=1}^{P} f_{ij} \left(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d \right) \leq s_i^l, i = 1, ..., S \tag{2}$$

$$x_{ij} \left(p_j^h + \left\lceil \frac{c_{ij} x_{ij}}{\max \left(\left\lfloor \frac{f_{ij} (y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d)}{p_j^h} \right\rfloor, 1 \right)} \right\rceil \cdot \left(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d \right) + \left\lceil \frac{n_{ij} x_{ij}}{\max \left(f_{ij}, 1 \right)} \right\rceil \cdot p_j^h p_j^n \right) \leq s_i^h,$$

$$j = 1, ..., P, i = 1, ..., S \tag{3}$$

$$x_{ij}\left(y_{ij}^{o_1}p_j^d + y_{ij}^{o_2}p_j^w\right) \le s_i^d, j = 1, ..., P, i = 1, ..., S \tag{4}$$

$$\sum_{j=1}^{P}\left(f_{ij} + c_{ij} + n_{ij}\right)p_j^b \le s_i^b, i = 1, ..., S \tag{5}$$

Constraint (2) restricts the shelf length. Constraint (3) assures that the height of each product on a shelf satisfies the shelf height limit. Capped products with front orientation ($p_j^{o_1} = 1$) are placed in a way that their width results in additional height to the facings of the product. Capped products with side orientation ($p_j^{o_2} = 1$) are placed in such a way that their depth results in additional height to the facings of the product. The total height equals the sum of the product height, cappings (Fig. 3) height and nestings (Fig. 4) height. Constructions $x_{ij}/\max(f_{ij}, 1)$ are used for omitting cases with the division by 0 if a product is not placed on the shelf. Constraint (4) restricts the shelf depth limit. Constraint (5) imposes the shelf weight limit.

3.8 Product constraints

$$\sum_{i=1}^{S} x_{ij} \ge s_j^{\min}, j = 1, ..., P \tag{6}$$

$$\sum_{i=1}^{S} x_{ij} \le s_j^{\max}, j = 1, ..., P \tag{7}$$

$$\sum_{i=1}^{S}\left(f_{ij} + c_{ij} + n_{ij}\right) \le p_j^s,\ j = 1, ..., P \tag{8}$$

$$\sum_{i=1}^{S} f_{ij} \ge f_j^{\min}, j = 1, ..., P \tag{9}$$

$$\sum_{i=1}^{S} f_{ij} \le f_j^{\max}, j = 1, ..., P \tag{10}$$

$$c_{ij} \ge c_j^{\min}, j = 1, ..., P, i = 1, ..., S \tag{11}$$

$$c_{ij} \le c_j^{\max} \cdot \left\lfloor \frac{f_{ij}\left(y_{ij}^{o_1}p_j^w + y_{ij}^{o_2}p_j^d\right)}{p_j^h} \right\rfloor,\ j = 1, \ldots, P,\ i = 1, \ldots, S \tag{12}$$

$$n_{ij} \geq n_j^{\min}, j = 1, ..., P, i = 1, ..., S \tag{13}$$

$$n_{ij} \leq n_j^{\max} f_{ij}, j = 1, ..., P, i = 1, ..., S \tag{14}$$

$$c_{ij} \cdot n_{ij} = 0, j = 1, ..., P, i = 1, ..., S \tag{15}$$

Constraints (6) and (7) restrict for the multi-shelf products, the minimum and the maximum number of shelves where these products should be placed. Constraint (8) imposes the available supply limit for a product. Constraints (9) and (10) impose the product facings lower and upper bound. Constraints (11) and (12) provide the lower and upper bounds of cappings per position. Constraint (12) ensures that the number of product facings is enough for the capping, i.e., the product may be capped. Constraints (13) and (14) guarantee the lower and upper bounds of nestings per facing position. Constraint (15) ensures the presentation of either cappings or nestings for a product, but not both of them at the same time.

3.9 Multi-shelves constraints

$$y_{ij}^{o_1} \cdot y_{ij}^{o_2} = 0, j = 1, ..., P, i = 1, ..., S \tag{16}$$

$$y_{ij}^{o_1} + y_{ij}^{o_2} = 1, j = 1, ..., P, i = 1, ..., S \tag{17}$$

$$\max_{i=1,...,S}(y_{ij}^{o_1}) \neq \max_{i=1,...,S}(y_{ij}^{o_2}), j = 1, ..., P \tag{18}$$

$$y_{ij}^{o_1} \leq p_j^{o_1}, j = 1, ..., P, i = 1, ..., S \tag{19}$$

$$y_{ij}^{o_2} \leq p_j^{o_2}, j = 1, ..., P, i = 1, ..., S \tag{20}$$

$$\forall(a, b : |a - b| \neq 1 \wedge a < b, a, b = 1, ..., S)[x_{aj} \cdot x_{bj} = 0], j = 1, ..., P \tag{21}$$

$$\forall\left(a, b : p_a^l = p_b^l, a, b = 1, ..., P\right)\left[x_{ia} = x_{ib}\right],\ i = 1, ..., S \tag{22}$$

For multi-shelf products, the requirement is to put the product on all shelves in the same orientation in order to make it noticeable in the same way on all the shelves. Constraints (16)–(17) indicate that only one orientation (front or side) is available for each product. Constraint (18)ensures the same orientation of the product on all shelves. Constraints (19)–(20) ensure the possibility of the front or side orientation, respectively. Constraint (21) allows the product to be placed only on adjacent shelves. Constraint (22) ensures that the products in the same cluster are placed together on the same shelf.

3.9.1 Category constraints

$$\left(\sum_{\substack{j=1,\\ p_j^k=k}}^{P} f_{ij}(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d) \geq \left[s_i^l \cdot m_k\right]\right) \vee \left(\sum_{\substack{j=1,\\ p_j^k=k}}^{P} f_{ij} = 0\right),\ i = 1, ..., S, k = 1, ..., K \quad (23)$$

$$\max_{i=1,...,S}\left(\sum_{\substack{j=1,\\ p_j^k=k}}^{P} f_{ij}\left(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d\right)\right) - \min_{i=1,...,S}\left(\sum_{\substack{j=1,\\ p_j^k=k}}^{P} f_{ij}\left(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d\right)\right) \leq \left[\max_{i=1,...,S}(s_i^l) \cdot t_k\right], k = 1, ..., K \quad (24)$$

$$x_{ij} \leq \min(\max(s_i^g - p_j^g, 0), 1), j = 1, ..., P, i = 1, ..., S \quad (25)$$

Constraint (23) guarantees the minimum category size if the products of this category exist on the shelf. Constraint (24) restricts the category tolerance between different shelves of the category in order to form the border between neighbouring categories. [...] denotes rounding to the nearest integer. Constraint (25) ensures the allocation of the products based on their sales potential subcategories. Products with lower sales potential can be placed on shelves with greater sales potential. Products with greater sales potential cannot be placed on shelves with lower sales potential.

3.9.2 Integrity constraints

$$x_{ij} \cdot s_i^l \cdot \left(\frac{y_{ij}^{o_1}}{p_j^w} + \frac{y_{ij}^{o_2}}{p_j^d}\right) \geq f_{ij},\ j = 1, ..., P,\ i = 1, ..., S \quad (26)$$

$$x_{ij} \leq f_{ij} \cdot \left(y_{ij}^{o_1} + y_{ij}^{o_2}\right),\ j = 1, ..., P,\ i = 1, ..., S \quad (27)$$

$$c_{ij} \leq x_{ij} \cdot c_j^{\max} \cdot \left\lfloor \frac{f_{ij}(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d)}{p_j^h} \right\rfloor, j = 1, ..., P, i = 1, ..., S \quad (28)$$

$$n_{ij} \leq x_{ij} \cdot n_j^{\max} \cdot f_{ij}, j = 1, ..., P, i = 1, ..., S \quad (29)$$

Constraints (26) and (27) ensure that when a product is placed on a shelf, the number of facings is non-zero and vice versa. Constraint (28) signifies capping relationships. Constraint (29) guarantees nesting relationships.

3.9.3 Decision variables

$$x_{ij} \in \{0, 1\} j = 1, ..., P, i = 1, ..., S \tag{30}$$

$$f_{ij} = \{f_j^{\min} .. f_j^{\max}\} j = 1, ..., P, i = 1, ..., S \tag{31}$$

$$c_{ij} = \left\{ c_j^{\min} ... c_j^{\max} \cdot \left\lfloor \frac{f_{ij}(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d)}{p_j^h} \right\rfloor \right\}, \; j = 1, ..., P, \; i = 1, ..., S \tag{32}$$

$$n_{ij} = \{n_j^{\min} ... n_j^{\max} \cdot f_j^{\max}\} j = 1, ..., P, i = 1, ..., S \tag{33}$$

$$y_{ij}^{o_1} \in \{0, 1\} j = 1, ..., P, i = 1, ..., S \tag{34}$$

$$y_{ij}^{o_2} \in \{0, 1\} j = 1, ..., P, i = 1, ..., S \tag{35}$$

The binary decision variable (30) shows if the product j is placed on the shelf i. Decision variables (31)–(33) represent the number of facings, cappings and nestings of the product j on the shelf i, which are integers bounded by the given upper bound values. Binary decision variables (34)–(35) show if the product j is placed on the shelf i in the front or side orientation accordingly.

4 New SSAP heuristics

In this section, we propose heuristics that can be applied to the variety of SSAP instances with different product and shelf constraints specific to each shelf.

4.1 Main steps

The proposed algorithm can be represented along these lines. The main goal of this algorithm is to create all possible product allocations with regard to all constraints. In the beginning, the algorithm does not specify the number of facings, cappings and nestings. It only determines if the product can be placed on the shelf. Next, the numeric values for facings, cappings and nestings are found based on the allocations received from the first procedures of the algorithm. Among the received allocations with facings, cappings and nestings specified, the most profitable solution is selected.

GenerateProductOnShelfAllocations()
{
  for $k \in \{k : k = 1,...,K\}$
    for $j \in \{j : p_j^k = k,\ j = 1,...,P\}$
      n:=$GenerateTernaryValues(k, j)$
    end
    n:=$GenerateProductSequences(k, n)$
    for $i \in \{S\}$
      n:=$GenerateShelfValues(k, i, n)$
    end
    n:=$GenerateShelfSequences(k, n)$
  end
  n:=$GenerateProductWidthSequences(n)$
  n:=$GenerateShelfCategorySequences(n)$
  n:=$GetMaxProfitSolution(n)$
}

Below the following notations are used. "Product sequence" means the sequence of products on one shelf with no regard as to which level this shelf will be situated (s_1, s_2 or s_3). "Shelf sequence" means the simultaneous sequences for all of the shelves (s_1, s_2 and s_3), created on the basis of the product sequences, the level of each shelf is known.

4.1.1 Main procedure GenerateProductOnShelfAllocations

This procedure selects the products from each category $\{j : p_j^k = k, j = 1, ..., P, k = 1, ..., K\}$ and generates possible product to shelf allocations (whether a product is placed on the shelf or not) with regard to the constraints. All the sequential steps of the remaining procedures are explained below.

4.1.2 GenerateTernaryValues procedure

This procedure constructs all sequences consisting of those product orientations which are allowed by product orientation constraints. The number of sequences is limited because, while generating, many allocations are excluded by the constraints which determine if a given product can be placed on the shelf or not. The allocations that do not satisfy all constraints can be excluded, which substantially reduces the number of generated sequences with respect to all possible ternary sequences.

According to the constraint (19) for the products with possible front orientation ($p_j^{o_1} = 1$) it generates ternary allocations using '0's and '1's. The number of such sequences is $\left|\{j : p_j^k = k,\, p_j^{o_1} = 1, j = 1, ..., P, k = 1, ..., K\}\right|$. E.g., 01,010—there are 5 products in the category, the 2nd and the 4th product is put on the shelf in front orientation. In this sequence, the following notations are used: 0—not placed, 1—front orientation, 2—side orientation.

Because the side orientation is the secondary one, all products have a front orientation ($p_j^{o_1} = 1$), but only a few of them have the additional side orientation ($p_j^{o_2} = 1$). Next, the procedure generates ternary allocations using '0's and '2's for products with possible side orientation ($p_j^{o_2} = 1$) according to the constraint (20). The number of such sequences is $\left|\{j : p_j^k = k,\, p_j^{o_2} = 1, j = 1, ..., P, k = 1, ..., K\}\right|$. For example, if the sequence is 20110—there are 5 products in the category, the 1st product is put on the shelf in the side orientation, the 3rd and the 4this put on the shelf in the front orientation.

Obviously, this procedure generates sequences in a way that excludes duplications, i.e., ternary sequences must be generated for each product considering its orientation.

4.1.3 GenerateProductSequences procedure

According to the constraint (22), all cluster products must be put together on the shelf, so this procedure excludes the incorrect sequences from the set of ternary sequences. Only correct sequences are left so that.

$$\forall(k{:}k=1,...,K)\left[\sum_{\substack{j,a=1,...,P,\\ p_j^k=k,\\ p_j^l=p_a^l,\\ j\neq a}} x_{ij}=\left|\{j{:}p_j^k=k,\ p_j^l=p_a^l,\ j\neq a,j,\ a=1,...,P\}\right|\right]\vee$$

$$\vee\forall(k{:}k=1,...,K)\left[\sum_{\substack{j,a=1,...,P,\\ p_j^k=k,\\ p_j^l=p_a^l,\\ j\neq a}} x_{ij}=0\right], i=1,...,S, \tag{36}$$

Which means that cluster products may be placed on the shelf together or may not be placed on the shelf at all.

In order to differentiate cluster and non-cluster products, the next parameter is defined.

$$z_j=\min_{j,a=1,...,P}\left(\left|\{j:p_j^k=k,p_j^l=p_a^l,j\neq a\}\right|-1,1\right)$$

$$z_j=\left\{\begin{array}{l}1,\ \text{if product } j \text{ is in a cluster}\\ 0,\ \text{otherwise}\end{array}\right\},j=1,...,P.$$

Claim. In this step, the number of possible product on shelf allocations equals:

$$\forall(k : k = 1, ..., K)\left(\prod_{\substack{j=1,...,P,\\ p_j^k=k,\\ z_j=0}}\left(p_j^{o_1}+p_j^{o_2}+1\right)\cdot\max_{a=1,...,P}\left(\prod_{\substack{j=1,...,P,\\ p_j^k=k,\\ z_j=1,\\ p_j^l=p_a^l,\\ j\neq a}}\left(p_j^{o_1}+p_j^{o_2}+1\right)\right)\right)$$

$\uparrow\uparrow$

non - cluster products cluster products

(37)

Proof. For non-cluster products, take the total possible variations with repetitions. For cluster products, calculate variations with repetitions only for one product in a cluster (because cluster products cannot be placed separately).

4.1.4 GenerateShelfValues procedure

This procedure constructs sequences for products that must be allocated on shelves taking into account the sales potentials subcategory constraint (25). It generates sequences specifically for each shelf on the basis of the sequences from the procedure GenerateProductSequences, i.e., for each shelf i ($i = 1, ..., S$), only ternary sequences are left so that ($p_j^g \leq s_i^g$).

4.1.5 GenerateShelfSequences procedure

From the sequences generated by the procedure GenerateShelfValues, this procedure excludes the ones that do not satisfy the minimum and maximum shelf number limit (constraints (6)–(7)). In order to take into account how many shelves the product can be allocated on, a parameter is provided.s_j^{real}—the real number of shelves to which a product can be allocated considering the shelf and the product sales potential subcategory s_i^g and p_j^g, the maximum number of shelves, which it can be allocated on is $s_j^{\max}$.

$$s_j^{real} = \min\left(\sum_{\substack{i=1,...,S\\ p_j^g\leq s_i^g}} 1, s_j^{\max}\right), j = 1, ..., P$$

Claim. The number of possible product on shelf allocations with regard to the minimum and the maximum number of shelves (constraints (6)–(7)) for the multi-shelf products and sales potentials subcategory (constraint (25)) equals:

$$\forall(k : k = 1, ..., K)\left(\prod_{\substack{j=1,...,P \\ p_j^k=k, \\ z_j=0}} (p_j^{o_1} + p_j^{o_2} + 1)^{s_j^{real}} \cdot \max_{a=1,...,P}\left(\prod_{\substack{j=1,...,P \\ p_j^k=k, \\ z_j=1, \\ p_j^l=p_a^l, \\ j\neq a}} \left(p_j^{o_1} + p_j^{o_2} + 1\right)^{\min\left(s_j^{real}\right)}\right)\right)$$

$$\uparrow\uparrow$$

non - cluster products cluster products

(38)

Proof. For non-cluster products, take all the possible variations with repetitions from (37), leave only those where the product sales potential subcategory does not exceed the shelf sales potential subcategory ($p_j^g \le s_i^g$). For cluster products, leave only those where the minimal sales potential subcategory in the cluster does not exceed the shelf sales potential subcategory:

$$\forall(k : k = 1, ..., K)\forall\left(j, a = 1, ..., P : p_j^k = k, z_j = 1, p_j^l = p_a^l, j \neq a\right)$$

$$\left[\begin{array}{ll} s_j^{real} = \min\limits_{\substack{j,a=1,...,P, \\ p_j^k=k, \\ z_j=1, \\ p_j^l=p_a^l \\ j\neq a}} (s_j^{real}) = \min\limits_{\substack{j,a=1,...,P, \\ p_j^k=k, \\ z_j=1, \\ p_j^l=p_a^l \\ j\neq a}} \left(\min\left(\sum\limits_{\substack{i=1,...,S \\ p_j^g\le s_i^g}} 1, s_j^{\max}\right)\right) \end{array}\right].$$

If the products in a cluster have different sales potential subcategories, they must be put on the shelf with the sales potential subcategory equal or greater than the maximum sales potential subcategory in the cluster. If a product is placed on the shelf, it cannot be placed on another shelf, so we sum up only those sequences where $\sum_{\substack{i=1,\ldots,S \\ p_j^g \le s_i^g}} 1 \le s_j^{\max}$.

Claim. The total number of possible product on shelf allocations considering all categories equals:

$$\prod_{k=1}^{K} \left(\prod_{\substack{j=1,\ldots,P \\ p_j^k=k, \\ z_j=0}} \left(p_j^{o_1} + p_j^{o_2} + 1 \right)^{s_j^{real}} \cdot \max_{a=1,\ldots,P} \left(\prod_{\substack{j=1,\ldots,P \\ p_j^k = k, \\ z_j = 1, \\ p_j^l = p_a^l, \\ j \neq a}} \left(p_j^{o_1} + p_j^{o_2} + 1 \right)^{s_j^{real}} \right) \right) \tag{39}$$

↑ ↑

non - cluster products cluster products

Proof. The total number of possible products on shelf allocations considering all categories is a multiplication of previously found values for each category (formula (38)).

Claim. The total number of product allocations in a general case is

$$(3^P)^S = 3^{PS} \tag{40}$$

Number 3 represents the possible allocations of the product: (1) if it is not placed on the shelf, (2) if it is placed on the shelf in the front orientation, (3) if it is placed on the shelf in the side orientation. All products may be placed on all shelves simultaneously.

Claim. The total number of product allocations in a general case that takes categories into account is:

$$\left(\prod_{\substack{k=1,\ldots,K \\ p_j^k=k}} 3^{\left| \left\{ j : p_j^k=k, j=1,\ldots,P \right\} \right|} \right)^S = \prod_{\substack{k=1,\ldots,K \\ p_j^k=k}} 3^{\left| \left\{ j : p_j^k=k, j=1,\ldots,P \right\} \right|} = 3^{PS} \tag{41}$$

In this formula, the total number of products P equals the sum of the number of products in each category, i.e. $P = \sum_{\substack{k=1,...,K \\ p_j^k = k}} \left| \{ j : p_j^k = k, j = 1, ..., P \} \right|$.

Claim. The total number of product allocations in our case, which takes the current constraints into account, can be reduced to:

$$\prod_{k=1}^{K} \left(\prod_{\substack{j=1,...,P \\ p_j^k = k, \\ z_j = 0}} \left(p_j^{o_1} + p_j^{o_2} + 1 \right)^{s_j^{real}} \cdot \max_{a=1,...,P} \left(\prod_{\substack{j = 1, ..., P \\ p_j^k = k, \\ z_j = 1, \\ p_j^l = p_a^l, \\ j \neq a}} \left(p_j^{o_1} + p_j^{o_2} + 1 \right)^{s_j^{real}} \right) \right) \tag{42}$$

The next steps of the procedure are as follows:

1. Exclude from the generated sequences the sequences with an incorrect orientation (constraints (16)–(18)).
2. Exclude from the generated sequences the sequences where the shelf depth limit is exceeded (constraint (4)).
3. Generate sequences with regard to the next shelf constraint (21). After this step, shelf permutations may be reduced. The number of shelf permutations may be left the same if there are no products allocated to multiple shelves in the given category.
4. From the generated set of sequences, exclude ones where the total width of facings lower bound on one shelf exceeds the total width of facings upper bound. Leave only those sequences that satisfy category tolerance between different shelves (constraint (24)).

$$\forall (k : k = 1, ..., K) \max_{\substack{i,a=1,...,S, \\ i \neq a \\ p_j^k = k}} \left(x_{ij} f_j^{\min} \left(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d \right) \right) - x_{aj} f_j^{\max} \left(y_{aj}^{o_1} p_j^w + y_j^{o_2} p_j^d \right) \leq \tag{43}$$

$$\leq \left[\max_{i=1,...,S} (s_i^l) \cdot t_k \right]$$

5. Based on the $\{f_j^{\min} ... f_j^{\max}\}$ if, in the result, the empty shelves are not expected within the category, at this step, the sequences with all '0's (empty shelf) and

consequently the sequences with all '1's or '2's (all products are put on this shelf) can be excluded.

6. Based on the assumption provided in step (5), get single shelf sequences and exclude empty and full shelves from them. In this step, we obtain the shelf sequences, which combine sequences for all shelves.

4.1.6 GenerateProductWidthSequences procedure

This procedure calculates the product sequence. The main steps of such a procedure are as follows:

```
GenerateProductWidthSequences(n)
{
  for i = 1 to n
    FindSpaceCategory(i)
    CreateItems(i)
    FindWidthLB(i)
    GenerateVariableWidths(i)
    CalculateProfit(i)
    CalculateTotals(i)
    OutputWidths(i)
  end
}
```

4.1.7 FindSpaceCategory procedure

In this procedure, the remaining length for a category is calculated based on the constraint (23). The same actions are done for weight.

The following notations will be used.b_j^{bin}—binary (0/1) value for the product j ($j = 1, ..., P$) in the current category in the sequence.b_j^{ter}—ternary (0/2) value for the product j ($j = 1, ..., P$) in the current category in the sequence.$c_k^{l\min}$—the minimum length of the category k ($k = 1, ..., K$).$c_k^{l\max}$—the maximum length of the category k ($k = 1, ..., K$).$c_k^{b\min}$—the minimum weight of the category k ($k = 1, ..., K$).$c_k^{b\max}$—the maximum weight of the category k ($k = 1, ..., K$).c_k^h—the height of the category k ($k = 1, ..., K$).c_k^d—the depth of the category k ($k = 1, ..., K$).r_j—the number of remaining available items of the product j ($j = 1, ..., P$).

$$x_{ij} = \max\left(b_j^{bin}, b_j^{ter}, 1\right)\sqrt{b^2 - 4ac},\ j = 1, \ldots, P,\ i = 1, \ldots, S \tag{44}$$

$$y_{ij}^{o_1} = \left\{ \begin{array}{l} 1,\ \max\left(b_j^{bin}, b_j^{ter}\right) = 1 \\ 0,\ \text{otherwise} \end{array} \right\},\ j = 1, \ldots, P,\ i = 1, \ldots, S \tag{45}$$

$$y_{ij}^{o_2} = \left\{ \begin{array}{l} 1,\ \max\left(b_j^{bin}, b_j^{ter}\right) = 2 \\ 0,\ \text{otherwise} \end{array} \right\},\ j = 1, ..., P,\ i = 1, ..., S \tag{46}$$

$$c_k^{l\min} = \sum_{\substack{j=1, \\ p_j^k = k}}^{P} x_{ij} f_j^{\min}\left(y_{ij}^{o_1} p_j^w + y_{ij}^{o_2} p_j^d\right),\ k = 1, ..., K \tag{47}$$

$$c_k^{l\max} = s_i^l - (K-1) \cdot \left[s_i^l \cdot m_k\right],\ i = 1, \ldots, S,\ k = 1, \ldots, K \tag{48}$$

$$c_k^{b\min} = \sum_{\substack{j=1, \\ p_j^k = k}}^{P} x_{ij}\left(f_j^{\min} + c_j^{\min} + n_j^{\min}\right)p_j^b,\ k = 1, ..., K \tag{49}$$

$$c_k^{b\max} = \left[\frac{c_k^{l\max} \cdot s_i^b}{s_i^l}\right],\ i = 1, ..., S,\ k = 1, ..., K \tag{50}$$

$$c_k^h = s_i^h,\ i = 1, \ldots, S,\ k = 1, \ldots, K \tag{51}$$

$$c_k^d = s_i^d,\ i = 1, \ldots, S,\ k = 1, \ldots, K \tag{52}$$

The formula (44) indicates if the product is put on the shelf in the investigated category in the current product sequence. Formulas (45) and (46) show if the product is put on the shelf in the investigated category in the front (45) or side (46) orientation. Formulas (47) and (48) represent the minimum and maximum category length on the shelf calculated on the facings from the given product sequence. Formulas (49) and (50) calculate the minimum and the maximum category weight. The category's height (51) and depth (52) equal the corresponding shelf values.

4.1.8 CreateItems procedure

We model the problem as a knapsack with internal sections, each section of which is dedicated to the product of the defined type. Because all the products, according to the generated sequence, must be placed into the knapsack, each section must exist. The goal is to define the number of items of each product ($f_j^{\max} + c_j^{\max} + n_j^{\max}$) and in consequence, the size of the section.

In this procedure, the items which will be later put in the knapsack are created in the following way. We have a bounded 0–1 knapsack, the set and the number of items ($f_j^{\max} + c_j^{\max} + n_j^{\max}$) which can be put into it is known. For each product in the current product sequence, we check if it is capped ($c_j^{\max} > 0$) or nested ($n_j^{\max} > 0$). If not, we add all $f_j^{\max}$ items to the set. The number of items that are added to the set is calculated with regard to the supply limit (constraint (8)), lower and upper bound of product facings (constraints (9) and (10)), cappings per position (constraints (11) and (12)), nestings per facing (constraints (13) and (14)). The available number of nests per face is also under the control of the height limit (constraint (3)). The products that are added to the items are marked with a *groupID* given at the step of item creation. One capped (nested) item is assigned a *groupID*. The separate items which are included in this capped (nested) group have the same *groupID*. This means that separate items and capped (nested) items cannot be taken simultaneously. Therefore, the items which can be taken simultaneously have different *groupID*, otherwise capped (nested) items, and corresponding not capped (not nested) items have the same *groupID*, which means that we can take either capped (nested) or not capped (not nested) items in the same group, but not both of them.

4.1.9 FindWidthLB procedure

We represent the knapsack as a knapsack with sections inside; in each section, different number of product items of the given type can be put. The total knapsack size is fixed; we modify only the size of the interior sections. In order to solve the knapsack problem later, this procedure defines the step $step_p$ with which the section size will be increased or decreased, i.e., the greatest common divisor *gcd* for each product in the category, i.e., the smallest item width of the given product type in the item set. From the previous procedure where items have been created, it is known that items may be represented as single product facing or merged from some facings with capping above or nestings inside. So, the width of an item (both for capped and nested ones) and weight (only for capped items) will be different. Therefore, in this case,*gcd* equals the minimum width of the product in the group with the same index j, i.e., all products with the same index j have width divisible by the minimum width in the group.

In the next listing, the following notations are used:$step_j$—*gcd* of items in the group with the same j ($j = 1, ..., P$).$start_j$—start width of the knapsack for items with index j ($j = 1, ..., P$).end_j—end width of the knapsack for items with index j ($j = 1, ..., P$).t—input parameter, a common multiple for $step_p$, used to take not one *gcd* but $t \cdot gcd$ of the product j ($j = 1, ..., P$) in order to perform local search faster.

The future width dimension for the product j will be $[0, start_j, start_j + step_j \ldots end_j]$.

4.1.10 GenerateVariableWidths procedure

In this procedure, we define the multisection knapsacks variants of the total section number $\left|\{j : p_j^k = k, j = 1, ..., P, k = 1, ..., K\}\right|$. The widths of sections vary from $start_j$ to end_j with the step $step_j$. All generated solutions are saved for later analysis.

4.1.11 CalculateProfit procedure

In this procedure, we sort items by profit in non-ascending order. Next, we add items to the knapsack section one by one if its width does not exceed the section size, supply limit p_j^s of the product j (also summing up cappings and nestings) is satisfied, and *groupID* for capped and nested items is the same.

4.1.12 CalculateTotals procedure

After the items have been chosen for each section of the knapsack, this procedure calculates facings, cappings, nestings, profit, width, weight, height, depth of each section of the knapsack.

4.1.13 OutputWidths procedure

In this procedure, we create a quality report for each width of the sections of the knapsack. Obviously, for each knapsack section size, different variants of totals will appear. Thus, for each binary b_j^{bin} or ternary b_j^{ter} value of product allocations, for each total knapsack width and weight, take items set with maximum profit.

4.1.14 GetMaxProfitSolution procedure

This procedure selects the sequence with the appropriate numbers of facings f_{ij}, cappings c_{ij} and nestings n_{ij} for the given product, which gives the maximum total profit. This is the best solution for the given product.

4.2 Proposed heuristics explanation

All heuristics described in this paper combine generated product sequences from 4.1.3 (GenerateProductSequences procedure), shelf sequences from 4.1.5 (GenerateShelfSequences procedure) and category totals (profit, width, weight, binary or ternary values of product allocations) from 4.1.14 (GetMaxProfitSolution procedure). All heuristics search only for an appropriate solution based on a formula (42). Figure 5 illustrates common and distinct steps in all proposed heuristics.

5 Computational Experiment

The computational experiments evaluate the performance of the solutions of the developed heuristics for the presented shelf space allocation problem. As there is neither real-world data available due to commercial confidentiality nor any benchmark found in literature or available from open sources, simulated test problems were generated. All of their data is based on real-life data.

The computational experiments were implemented in Visual C# 2015 and MS SQL Server 2014.

Fig. 5 Heuristics steps

Language: Visual C# 2015
Microsoft Visual Studio Community 2015
Version 14.0.25431.01 Update 3
Microsoft.NET Framework
Version 4.6.01055
Microsoft SQL Server Management Studio12.0.2269.0

An optimal (or maximum feasible in some cases) solution to compare with has been found using commercial solver IBM ILOG CPLEX Optimization Studio Version: 12.7.1.0.

5.1 Heuristics performance

Some of the advantages of the presented algorithms are:

- Lack of randomly selected elements, so there is no need to run it several times;
- Data is visible in each step and can be easily checked;
- The coefficient parameters that can also be easily estimated while watching the data.

Table 2 summarizes the number of solutions found by all heuristics where the heuristics solution was the best (most profitable) feasible solution among those that were found by a given heuristic. Heuristic H_3 was the best among all of them, finding the maximum solution in 20 cases. But as we see in Table 3, other heuristics found the maximum feasible solution also in cases where H_3 cannot achieve this. So there is definitely a necessity for other implemented ones. Compared to the CPLEX solver, heuristics found the solution in 34 cases while CPLEX in 23 cases only. In Table 3, we do not differentiate which of these solutions was better. Table 3 presents the total number of feasible solutions received. This observation proves the necessity of the heuristics because, clearly, instances for which CPLEX did not find a feasible solution do exist.

In order to estimate the performance of the heuristics, the following notations will be used.

U_h—the profit of the heuristics (calculated in formula (1)).
U_o—the profit of the CPLEX solution.

Table 2 Number of solutions found by Heuristics where Heuristics solution is a maximum feasible solution

Heuristics solution is a maximum feasible solution	Heuristic H_1	Heuristic H_2	Heuristic H_3	Heuristic H_4	Heuristic H_5	Heuristic H_6
Number of solutions	4	18	20	5	5	9

Table 3 Number of solutions found by Heuristics and by CPLEX Solver

Solution exists	Heuristics	CPLEX
Number of solutions	34	23

U_h/U_o—the profit ratio of the heuristics to the optimal (or maximum feasible in some cases) solution.

Table 4 presents the quality of solutions found by heuristics compared to the CPLEX solver and to the results reported by Czerniachowska and Hernes [13]. It could be observed that the average profit ratio of the best heuristic is 95.25%, with minimal and maximal values of 87.78% and 99.84%, respectively. Not bad, but this situation should be improved in the future. As we remember from 4.1.13 we took items set with maximum profit for each equal width and weight. At this step, the less profitable sequences were excluded, only the best ones for each width and weight were taken, but later these best ones were excluded because of not satisfy other constraints. As we see, the CPLEX solver created a better solution on the basis of a not profitable sequence. So in future research, the method of including the worst sequences without increasing their number so much should be proposed. Nevertheless, as we see, there are some advantages of the proposed approach. In 11 cases, heuristics found the solution, but CPLEX did not because there were few sequences to be analyzed, so taking the best sequences in 4.1.13 helped to create a feasible solution fast. It could be noticed that there was no reverse situation where CPLEX found the solution, but heuristics did not. In 11 cases, the solution was found neither by heuristics nor by CPLEX. This case illustrates a situation from the real world where retailers order something that cannot be implemented, e.g., there are too many products in the category that they cannot be placed in beautiful columns, it is better to divide them in some other way, minimum category size or category tolerance should be changed, the shelf weight or length does not fit, or the sales potentials of the products are incorrect so that on there are too many products one shelf and too few on another. All these errors are visible and could be analyzed at each described step. As we remember, if product sequences (4.1.3) or shelf sequences (4.1.5) could not be generated, there is no reason to waste time trying to find the solution to this data. In such a situation, a hint should be given to space planners explaining what constraint is not met and why and what should be corrected. The same thing happens if we receive too many sequences in one category and too few in another. As we see, the profit ratio does not depend on product number or on the shelf length, compared to the previous research. Based on the nature of the problem, CPLEX could find the solution neither for small instances (15 products) nor for large instances (50 products). This depends on the category parameters, such as tolerance and minimum category size, as well as the number of possible product and shelf sequences to be generated. If the input data is too complicated, the solution could not be found by CPLEX, if the input data is less complicated, the solution could be found, but in this case, heuristics would be slightly worse.

Comparing the obtained quality results with the results by Czerniachowska and Hernes [13], we can observe that the minimum and average profit ratios of the proposed heuristics are a bit higher. The minimum profit ratio is 87.78% (Table 4) compared to 86.80% of the mentioned research. The average profit ratio is 95.25% (Table 4) compared to 94.57% of the mentioned research. The maximum profit ratio is the same and equals 99.84%. The standard deviation of the proposed heuristics is 3.93% (Table 4) compared to 4.17%, which makes them more stable.

Table 4 The quality of the heuristics solution compared to CPLEX and the latest research

Products	Shelf width	Profit ratio	Profit ratio reported by Czerniachowska and Hernes [13]	Solution exists
10	250	99.00%	98.30%	Heuristics, CPLEX
	375	98.52%	98.52%	Heuristics, CPLEX
	500	98.60%	97.93%	Heuristics, CPLEX
	625	97.49%	97.01%	Heuristics, CPLEX
	750	97.74%	97.31%	Heuristics, CPLEX
15	250			Heuristics
	375			Heuristics
	500			Heuristics
	625			Heuristics
	750			Heuristics
20	250			
	375	99.84%	99.84%	Heuristics, CPLEX
	500	97.72%	97.72%	Heuristics, CPLEX
	625	92.68%	92.68%	Heuristics, CPLEX
	750	88.07%	88.07%	Heuristics, CPLEX
25	250	99.20%	97.95%	Heuristics, CPLEX
	375	99.40%	98.77%	Heuristics, CPLEX
	500	90.19%	89.58%	Heuristics, CPLEX
	625	90.17%	89.16%	Heuristics, CPLEX
	750	93.67%	93.60%	Heuristics, CPLEX
30	250			
	375			
	500			
	625			
	750			
35	250			
	375	97.02%	97.34%	Heuristics, CPLEX
	500	91.49%	86.80%	Heuristics, CPLEX
	625	94.76%	90.87%	Heuristics, CPLEX
	750	93.30%	93.09%	Heuristics, CPLEX
40	250			
	375	99.20%	97.40%	Heuristics, CPLEX
	500	90.88%	91.30%	Heuristics, CPLEX
	625			Heuristics
	750			Heuristics
45	250			
	375	87.78%	87.87%	Heuristics, CPLEX
	500	95.86%	95.86%	Heuristics, CPLEX
	625			Heuristics
	750	98.04%	98.08%	Heuristics, CPLEX

Table 4 (continued)

Products	Shelf width	Profit ratio	Profit ratio reported by Czerniachowska and Hernes [13]	Solution exists
50	250			
	375			
	500			Heuristics
	625			Heuristics
	750			Heuristics
Min		87.78%	86.80%	
Avg		95.25%	94.57%	
Max		99.84%	99.84%	
St.Dev		3.93%	4.17%	

The processing time for searching for the maximum feasible solution, which on average varies from 16.79 up to 54.55 s comparing all heuristics. The heuristic H_5 was the fastest. The heuristic H_6 was the slowest.

Table 5 compares the total time of execution of heuristics in the proposed approach and in the research by Czerniachowska and Hernes [13]. Total time for our approach means the sum of execution times of all 6 heuristics. The total time for the research by Czerniachowska and Hernes [13] means the sum of execution times of both 2 heuristics that they proposed. We used such a measure of the total time because we needed to run all 6 heuristics before we got the answer or decided which heuristics was the best.

Comparing the achieved time results with the results by Czerniachowska and Hernes [13], we can notice that proposed heuristics, on average, found the solution slower (15.36 min), compared to 10.82 min in previous research. But total time combines the time of all 6 heuristics. On average, the execution time of one heuristics is less than a minute, so it is faster than the average execution time of a single heuristics in the previous research. The time limit for CPLEX was set to 2 min because it is slightly more than the average execution time of a single heuristics in our research. But in most cases, CPLEX spent for solution significantly less than 2 min.

The heuristic H_1 is used for initial space searching when neither the solution range nor approximate profit for each category is known. The heuristic H_2 is used for the improvement of the solution, which was found by the heuristic H_1. Steering parameters of the heuristic H_2 help to improve the total solution because they improve the partition solutions for each category. The heuristic H_3 is also used for the improvement of the solution, which was found by heuristic H_1, but it is used when there are too many results to check that heuristic H_2 cannot do this in a reasonable time. The heuristics H_4, H_5, H_6 are appropriate for large instances and can

Table 5 Total time of execution of heuristics in the proposed approach and in the research by Czerniachowska and Hernes [13]

Products	Shelf width	Total time [min] of the proposed 6 heuristics	Total time [min] of the 2 heuristics reported by Czerniachowska and Hernes [13]
10	250	4.23	3.17
	375	6.23	1.68
	500	5.59	2.33
	625	9.81	4.66
	750	4.82	4.44
15	250	6.01	2.06
	375	5.27	1.49
	500	7.35	1.50
	625	20.63	1.45
	750	10.30	1.07
20	250	12.41	0.00
	375	28.41	21.96
	500	10.59	12.27
	625	31.56	7.78
	750	31.25	16.59
25	250	10.00	19.55
	375	28.38	35.62
	500	10.53	3.22
	625	7.71	2.91
	750	22.38	18.63
30	250	10.48	0.00
	375	24.79	0.00
	500	46.59	0.00
	625	8.47	0.00
	750	5.34	0.00
35	250	9.88	0.00
	375	14.89	50.32
	500	14.19	4.42
	625	15.70	19.41
	750	12.26	6.03
40	250	5.96	0.00
	375	12.37	46.26
	500	8.51	3.86
	625	11.54	6.13
	750	33.74	25.35
45	250	20.02	0.00
	375	22.92	5.07
	500	26.52	15.79
	625	22.15	13.23
	750	27.51	25.40

Table 5 (continued)

Products	Shelf width	Total time [min] of the proposed 6 heuristics	Total time [min] of the 2 heuristics reported by Czerniachowska and Hernes [13]
50	250	7.15	0.00
	375	9.96	0.00
	500	25.43	5.45
	625	11.75	8.39
	750	9.70	89.29
Min		4.23	0.00
Avg		15.36	10.82
Max		46.59	89.29
St.Dev		9.78	17.03

Table 6 The total number of product allocations in the general case compared to the proposed solution

Products	Number of product allocations in the general case	Authors' proposed number of product allocations
10	$2.06 \cdot 10^{14}$	$7.68 \cdot 10^{2}$
15	$2.95 \cdot 10^{21}$	$3.69 \cdot 10^{4}$
20	$4.24 \cdot 10^{28}$	$9.83 \cdot 10^{4}$
25	$6.08 \cdot 10^{35}$	$2.36 \cdot 10^{6}$
30	$8.73 \cdot 10^{42}$	$2.83 \cdot 10^{7}$
35	$1.25 \cdot 10^{50}$	$4.72 \cdot 10^{6}$
40	$1.80 \cdot 10^{57}$	$4.35 \cdot 10^{10}$
45	$2.58 \cdot 10^{64}$	$9.66 \cdot 10^{9}$
50	$3.70 \cdot 10^{71}$	$6.68 \cdot 10^{13}$

process a large amount of data very fast. They differ from each other by steering parameters, grouping methods, and sorting order.

Table 6 shows the total number of product allocations in the general case (formulas (40) or (41)) and the total number of product allocations of the proposed solution by authors (formula (42)) calculated on the basis of the test data. As can be observed, there is a huge difference between the general case and our method. It should be highlighted that in our proposed heuristics, only data that satisfy all the constraints is taken in each step. That is why the reduction in the number of solutions is so visible and so high.

6 Conclusion

This paper focuses on the SSAP in order to maximize the total profit. Retailers are looking for expert solutions or advice on how to differentiate and allocate products while minimizing lost revenue, maximizing profit, find a balance between meeting customer needs and multiple inputs. Based on the NP-hard nature of the problem examined, which is an extension of a knapsack problem and is difficult to be solved, there is a growing interest of scientists in developing heuristics and metaheuristics for the SSAP.

We proposed 6 heuristics and gave detailed practical algorithms with enough explanation, which are ready to be implemented in other category management problems. The first 3 heuristics are better for small instances. The last 3 ones are designed especially for large instances, as they process large amounts of data in a very short time. Examples of implementation and changing steering parameters are also included. The new practicable mathematical model was developed with the objective of maximizing the retailer's profit, which takes into account merchandising visual products display, i.e., horizontal products grouped into categories and vertical shelf levels based on the products' values and sales potentials.

To examine the performance of the proposed heuristics, 45 test cases were investigated. Among them, heuristics found solutions in 34 cases while CPLEX in 23 only. The profit ratio of the developed heuristics in the best case on average is 95.25%, with its minimal and maximal values 87.78% and 99.84%, accordingly.

The main characteristics which differentiate the proposed solution techniques from the previous ones are the lack of randomly generated or randomly selected elements and the fact that data in all described steps is visible and can be easily analyzed or adjusted by the coefficient steering parameters. In 11 cases where a solution was not found, the bits of advice of changing data could be given to category planners, i.e., changing the product grouping method, increase the minimum category size and tolerance, and reassign sales potentials. Very frequently, it occurs that the category planner cannot estimate if the solution exists. In the simplest case, the proposed steps of product and shelf sequence generation help him with this task. This detail also differentiates our research from the previous ones because it has a method of checking constraints quickly. The next steps could show which category does not meet constraints if the solution is not found. Heuristics execute in a couple of seconds or minutes, depending on the instance and heuristic type. A shorter calculation time is achieved because, in the beginning, we reduce the search space, generating only the possible product and shelf sequences, and then only the correct single parts of the future solution are taken, excluding the combinations which do not satisfy the constraints. So in all steps of the proposed algorithm, only the correct parts of the constructed solutions are considered. In most metaheuristics, there are random local searches, crossovers, and mutation operations that can make a correct solution infeasible, and thus, repair procedures have to be used. For example, Czerniachowska [10] developed a correction and solution improvement procedure in order to obtain the appropriate solution after each GA step. She also excluded a shelf with large packaged products from GA implementation and adopted dynamic programming to

solve it optimally. Thus, different methods were used for solution correction. But in the current research, the heuristics generate only appropriate solutions which do not need to be corrected.

As for the limitation of the proposed heuristic approach, there is no solution component evaluation while composing the whole solution. For example, in a genetic algorithm, highly rated components can be selected for crossover and solution creation at further steps. In the proposed approach, we can control the number of solutions generated and evaluate the total solution.

The results of the research can be used, for example, in order to implement a shelf space allocation module in retail information systems. There are several interesting, practically relevant questions and directions available for future investigation. We recommend the shelf space allocation model with included constraints of store layout for future research. It would be interesting and practically relevant to expand the proposed mathematical model with the goal of maximizing customer traffic. In addition, the model should consider clockwise or anti-clockwise customer direction.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12597-023-00636-1.

Acknowledgements The project is financed by the Ministry of Science and Higher Education in Poland under the program "Regional Initiative of Excellence" 2019–2022, project number 015/RID/2018/19, total funding amount 10,721,040.00 PLN. Krzysztof Michalak acknowledges the support by the Polish National Science Centre under grant no. 2015/19/D/HS4/02574.

Author contributions Conceptualization: Kateryna Czerniachowska; Krzysztof Michalak; Marcin Hernes. Methodology: Kateryna Czerniachowska. Formal analysis and investigation: Kateryna Czerniachowska.Writing—original draft preparation: Kateryna Czerniachowska. Writing—review and editing: Krzysztof Michalak; Marcin Hernes. Funding acquisition: Krzysztof Michalak; Marcin Hernes. Resources: Kateryna Czerniachowska;Krzysztof Michalak; Marcin Hernes. Supervision: Marcin Hernes. All authors have read and agreed to the published version of the manuscript.

Funding The project is financed by the Ministry of Science and Higher Education in Poland under the program "Regional Initiative of Excellence" 2019–2022, project number 015/RID/2018/19, total funding amount 10,721,040.00 PLN. Krzysztof Michalak acknowledges the support by the Polish National Science Centre under grant no. 2015/19/D/HS4/02574.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Anic, I.D., Radas, S., Lim, L.K.S.: Relative effects of store traffic and customer traffic flow on shopper spending. Int. Rev. Retail Distrib. Consum. Res. **20**(2), 237–250 (2010). https://doi.org/10.1080/09593961003701841
2. Bai, R., Kendall, G.: An investigation of automated planograms using a simulated annealing based hyper-heuristic. Oper. Res. Comput. Sci. Interfaces Seri. **32**, 87–108 (2005). https://doi.org/10.1007/0-387-25383-1_4
3. Bai, R., Kendall, G.: A model for fresh produce shelf-space allocation and inventory management with freshness-condition-dependent demand. INFORMS J. Comput. **20**(1), 78–85 (2008). https://doi.org/10.1287/ijoc.1070.0219
4. Bai, R., van Woensel, T., Kendall, G., Burke, E.K.: A new model and a hyper-heuristic approach for two-dimensional shelf space allocation. 4OR **11**(1), 31–55 (2013). https://doi.org/10.1007/s10288-012-0211-2
5. Bianchi-Aguiar, T., Silva, E., Guimarães, L., Carravilla, M.A., Oliveira, J.F., Amaral, J.G., et al.: Using analytics to enhance a food retailer's shelf-space management. Interfaces **46**(5), 424–444 (2016). https://doi.org/10.1287/inte.2016.0859
6. Bianchi-Aguiar, T., Silva, E., Guimarães, L., Carravilla, M.A., Oliveira, J.F.: Allocating products on shelves under merchandising rules: multi-level product families with display directions. Omega (United Kingdom) **76**, 47–62 (2018). https://doi.org/10.1016/j.omega.2017.04.002
7. Borin, N., Farris, P.W., Freeland, J.R.: A model for determining retail product category assortment and shelf space allocation. Decis. Sci. **25**(3), 359–384 (1994). https://doi.org/10.1111/j.1540-5915.1994.tb01848.x
8. Borumand, A., Beheshtinia, M.A.: A developed genetic algorithm for solving the multi-objective supply chain scheduling problem. Kybernetes **47**(7), 1401–1419 (2018). https://doi.org/10.1108/K-07-2017-0275
9. Choubey, N.: Floor layout optimization using genetic algorithm. Int. J. Curr. Res. **9**(07), 53529–53533 (2017)
10. Czerniachowska, K.: A genetic algorithm for the retail shelf space allocation problem with virtual segments. Opsearch (2021). https://doi.org/10.1007/s12597-021-00551-3
11. Czerniachowska, K., Hernes, M.: A genetic algorithm for the shelf-space allocation problem with vertical position effects. Mathematics **8**(11), 1–20 (2020). https://doi.org/10.3390/math8111881
12. Czerniachowska, K., Hernes, M.: Simulated annealing hyper-heuristic for a shelf space allocation on symmetrical planograms problem. Symmetry **13**(7), 1182 (2021). https://doi.org/10.3390/sym13071182
13. Czerniachowska, K., Hernes, M.: A heuristic approach to shelf space allocation decision support including facings, capping, and nesting. Symmetry **13**(2), 1–18 (2021). https://doi.org/10.3390/sym13020314
14. Czerniachowska, K., Lutosławski, K.: Dynamic programming approach for solving the retail shelf-space allocation problem. Procedia Comput. Sci. **192**, 4320–4329 (2021). https://doi.org/10.1016/J.PROCS.2021.09.208
15. Czerniachowska, K., Lutosławski, K., Kozina, A., Mateńczuk, K., Markowska, A., Kozierkiewicz, A., Pietranik, M., Roemer, I., Schieck, M.: Shelf space allocation problem with horizontal shelf division. Procedia Comput. Sci. **192**, 1550–1559 (2021). https://doi.org/10.1016/J.PROCS.2021.08.159
16. Czerniachowska, K., Sachpazidu-Wójcicka, K., Sulikowski, P., Hernes, M., Rot, A.: Genetic algorithm for the retailers' shelf space allocation profit maximization problem. Appl. Sci. **11**(14), 6401 (2021). https://doi.org/10.3390/app11146401
17. Desrochers, D.M., Nelson, P.: Adding consumer behavior insights to category management: improving item placement decisions. J. Retail. **82**(4), 357–365 (2006). https://doi.org/10.1016/j.jretai.2006.08.009
18. Düsterhöft, T., Hübner, A., Schaal, K.: A practical approach to the shelf-space allocation and replenishment problem with heterogeneously sized shelves. Eur. J. Oper. Res. **282**(1), 252–266 (2020). https://doi.org/10.1016/j.ejor.2019.09.012
19. Elbers, T.: The effects of in-store layout-and shelf designs on consumer behaviour. Available at http://edepot.wur.nl/369091 (2016)

20. Esparcia-Alcázar, A.I., Martínez-García, A.I.: Linear shelf space allocation using a multi objective evolutionary algorithm. Technical Report ITI-SAC-027, Instituto Tecnologico de Informatica, pp. 1–23. (2008)
21. Esparcia-Alcázar, A.I., Lluch-Revert, L., Albarracín-Guillem, J.M., et al.: An evolutionary algorithm for the product to shelf allocation problem. In: *2006 IEEE Congress on Evolutionary Computation, CEC 2006*, pp. 3197–3203. https://doi.org/10.1109/cec.2006.1688714 (2006)
22. Gabrielli, V., Cavazza, N.: The influence of in-store product holders on orientation towards the product and on purchase intention. Int. Rev. Retail Distrib. Consum. Res. **24**(3), 311–327 (2014). https://doi.org/10.1080/09593969.2013.862507
23. Gajjar, H.K., Adil, G.K.: A piecewise linearization for retail shelf space allocation problem and a local search heuristic. Ann. Oper. Res. **179**(1), 149–167 (2010). https://doi.org/10.1007/s10479-008-0455-6
24. Ghazavi, E., Lotfi, M.M.: Formulation of customers' shopping path in shelf space planning: a simulation-optimization approach. Expert Syst. Appl. **55**, 243–254 (2016). https://doi.org/10.1016/j.eswa.2016.01.043
25. Hansen, J.M., Raut, S., Swami, S.: Retail shelf allocation: a comparative analysis of heuristic and meta-heuristic approaches. J. Retail. **86**(1), 94–105 (2010). https://doi.org/10.1016/j.jretai.2010.01.004
26. Heydari, M., Yousefli, A.: A new optimization model for market basket analysis with allocation considerations: a genetic algorithm solution approach. Manag. Mark. **12**(1), 1–11 (2017). https://doi.org/10.1515/mmcks-2017-0001
27. Hübner, A.H., Kuhn, H.: Retail category management: state-of-the-art review of quantitative research and software applications in assortment and shelf space management. Omega **40**(2), 199–209 (2012). https://doi.org/10.1016/j.omega.2011.05.008
28. Hwang, H., Choi, B., Lee, M.J.: A model for shelf space allocation and inventory control considering location and inventory level effects on demand. Int. J. Prod. Econ. **97**(2), 185–195 (2005). https://doi.org/10.1016/j.ijpe.2004.07.003
29. Kim, G., Moon, I.: Integrated planning for product selection, shelf-space allocation, and replenishment decision with elasticity and positioning effects. J. Retail. Consum. Serv. **58**, 102274 (2021). https://doi.org/10.1016/j.jretconser.2020.102274
30. Lim, A., Rodrigues, B., Zhang, X.: Metaheuristics with local search techniques for retail shelf-space optimization. Manag. Sci. **50**(1), 117–131 (2004). https://doi.org/10.1287/mnsc.1030.0165
31. Ozcan, T., Esnaf, S.: A discrete constrained optimization using genetic algorithms for a bookstore layout. Int. J. Comput. Intell. Syst. **6**(2), 261–278 (2013). https://doi.org/10.1080/18756891.2013.768447
32. Pinto, F., Soares, C.: Space allocation in the retail industry: A decision support system integrating evolutionary algorithms and regression models. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) Vol. 8190 LNAI, pp. 531–546. https://doi.org/10.1007/978-3-642-40994-3_34(2013)
33. Pinto, F., Soares, C., Brazdil, P.: Combining regression models and metaheuristics to optimize space allocation in the retail industry. Intell. Data Anal. **19**(S1), 149–162 (2015). https://doi.org/10.3233/IDA-150775
34. Rabbani, M., Salmanzadeh-Meydani, N., Farshbaf-Geranmayeh, A., Fadakar-Gabalou, V.: Profit maximizing through 3D shelf space allocation of 2D display orientation items with variable heights of the shelves. Opsearch **55**(2), 337–360 (2018). https://doi.org/10.1007/s12597-018-0335-z
35. Urban, T.L.: An inventory-theoretic approach to product assortment and shelf-space allocation. J. Retail. **74**(1), 15–35 (1998). https://doi.org/10.1016/S0022-4359(99)80086-4
36. Valenzuela, A., Raghubir, P.: Center of orientation: effect of vertical and horizontal shelf space product position. In: NA-Advances in Consumer Research Volume 36, Eds. Ann L. McGill and Sharon Shavitt, pp. 100–103. Association for Consumer Research, Duluth, MN (2009).
37. Van Nierop, E., Fok, D., Franses, P.H.: Interaction between shelf layout and marketing effectiveness and its impact on optimizing shelf arrangements. Mark. Sci. **27**(6), 1065–1082 (2008). https://doi.org/10.1287/mksc.1080.0365