Maximizing Revenue Potential of Your Retail Shelf Space

—— A Solution to Shelf Space Allocation Problem

Group 25

Guanrong QIU	A0198358J
Yihe Wang	A0262735B
Xiaohan Shen	A0262717B
Manan Lohia	A0262838U
Bhargav Sagiraju	A0262798J
Yuxue YAO	A0262702M

Table of Contents

1. Introduction

The increasing efficiency of supply chains has seen the retail sector grow by leaps and bounds, cementing its role as a major entity connecting consumers to suppliers and while the advent of e-commerce has impacted the significance of brick-and-mortar shops, supermarkets continue to enjoy a beneficial position as a one-stop shop for a variety of items, especially for daily household items and groceries.

The need to stay competitive in an ever-changing environment has spurred the use of Information Technology, with large chains such as Walmart being one of the greatest drivers of innovation and implementation of technology to boost revenue.

One significant application of Operations Research in the sector is termed the Shelf Space Allocation Problem (SSAP). The problem is motivated by studies suggesting that in-store factors such as product placement play a role in a consumer's on-site decision to purchase an item, more so in cases of unplanned purchases or when an item of their choice is unavailable.

Given the increase in the number of brands and SKUs (Stock Keeping Units) for each product category, the limited shelf space in a physical store is an extremely valuable resource and efforts in the area have been directed toward building optimization models to capture the effects of product placement on purchases and identify optimal shelf space allocation for a variety of products, under the constraints of merchandising rules and internal policies.

In our project, we propose a simple model to determine shelf space allocation and discuss the various factors that can affect the decision. Section 2 expands on the business problem and in Section 3 we expound on our approach to developing a model, explaining any assumptions made along the way. Section 4 describes how we have tested our model using sample data. Section 5 concludes by summarizing our study and suggesting avenues for further development of the SSA problem.

2. Problem Discussion

Traditionally, store managers manually create planograms using specialized software to design the layout of shelves. The distribution of shelves and categories within a store are upstream problems commonly termed store space planning that is handled separately, and planograms are specifically designed for individuals or a small group of categories. They illustrate the placement of individual products on the shelf in terms of the number of facings and orientation of the product. Figure 1 provides an example of a planogram and its corresponding implementation in a store

Figure 1. Planogram Example.

A major shortcoming of most existing software solutions, however, is that they do not implement optimization of any kind due to the complexities involved. Hence, the aim of a model is to replicate the planogram construction process and to automatically determine the optimal distribution of products over shelves for maximum profit.

2.1 Basic Shelf Space Allocation Problem

At a very basic level, the SSA problem resembles a knapsack problem wherein we wish to select a group of products with different lengths and different revenue potentials for each shelf (having a constant length/capacity) to maximize the revenue or profit from the sale of such products. However, there are several factors that make the problem more complex than the knapsack problem due to the nature of the constraints affected by real business problems. Understanding the context in which the SSA problem needs to be solved is therefore important as it affects the aforementioned factors which directly impact the parameters to be used in a model.

The primary factor is that shelf space is limited, and various products have different dimensions which take up different amounts of space. The amount of space taken, and the location of the product impacts the revenue potential of that product. Hence, balancing the location and space occupied by a product and the expected profit from the sale is a primary concern to the problem. It is often the case that these effects are non-linear, so expressing them in the objective function and constraints requires certain assumptions and simplifications in the model to maintain tractability. While all three dimensions of a product, namely the height, length, and width are relevant in assessing the shelf space taken up by the product, to simplify the model, we only consider the length, or the horizontal space occupied by the product as we believe that it is the most valuable of the three dimensions when assigning shelf space.

For the sake of simplicity, we ignore any invisible inventory, such as products placed behind other products in the front row on a shelf. We believe this is a fair assumption as inventory that cannot be seen does not influence a consumer's decision to buy a product. This also ties into our assumption that the height or width of a product is not as important as its length in assessing its value on the shelf. Besides, Yang and Chen [1999] concluded that retailers can prevent stockout occurrences by building effective logistics systems, which means we can ignore the invisible inventory on the shelf.

2.2 Space Elasticity

Space elasticity was originally defined by Curhan [1972] as "the ratio of relative change in unit sales to relative change in shelf space." Experiments have concluded that products' demand increases as more space are allocated to them. However, the increasing rate slows down until a steady point, resembling an "S" shape. If we model the space elasticity exactly in the "S" shape, the objective function will become nonlinear. Therefore, instead of using the "S" shaped space elasticity. We followed Lim et al. [2004], which states that retailers prefer to operate on the linear portion of the "S" shaped curve of marginal returns. This assumption will simplify our model formulation as it means doubling the space allocated to a product will double the revenue of the product.

2.3 Row Effectiveness

When placing products on the shelves, different levels may have different effectiveness. For example, rows that are at eye level are easier to attract customers' attention than rows at the top or at ground level. Therefore, we also incorporate row effectiveness in our model by assigning different rows with different revenue multiples.

2.4 Contiguous Rectangular Placement

It is also desirable that in the case of a product being allotted multiple shelves, it occupies a "rectangular" block of space, i.e., it occupies the same amount of space on each shelf and the shelves are adjacent. This requirement stems from the desire to ease inventory management and restocking and improve the customer experience. Intuitively, it makes sense that it is better for all SKUs of a product to be in the same general location rather than be scattered across shelves.

2.5 Synergy Effect

Secondly, merchandising agreements or policy decisions often dictate that a set of products of a certain nature be placed together. For example, an agreement with a brand might require that all the products of the brand occupy a contiguous region on one or more shelves, or across cabinets. Similarly, it is common for a store to place products of the same category in the same cabinet for a synergy effect to ease the consumer experience, and this would enable customers to find their desired product quickly by locating their category.

2.6 Cross-Elasticity

Finally, products placed on the same shelf can exhibit a cross-elasticity effect, especially in cases where products are complements or substitutes. This suggests that the placement of one product can in fact affect the demand for another product, adding another level of complexity to our problem as the demands for different products are no longer independent of each other.

While there are several other factors to be considered when formulating the SSA problem, such as the limited shelf life of products, as well as the upstream problems of supply chain and inventory management, product assortment selection, etc., we believe that the aforementioned factors are crucial in building the most basic but nonetheless effective model to tackle the SSA Problem. Even within the limited scope of these factors, the model formulation and solution prove to be an NP-Hard problem, suggesting that arriving at an optimal solution may not always be feasible.

3. Model Formulation

To simplify model formulation and solution, we take a stepwise approach in our solution. Initially, we only consider the problem of products having different face lengths. This is expanded upon in section 3.1. In the subsequent sections, we sequentially add constraints and modify the objective to account for the condition of contiguous rectangular allotment of products (Section 3.2), for synergy effect (Section 3.3), and then for cross-elasticity effects between products (Section 3.4).

We consider the case where there are multiple cabinets, and each cabinet has several shelves. The problem is to allocate an assortment of said products to various shelves across the cabinets to maximize revenue while meeting certain limitations put forth by requirements of contiguous allocation of products and product categories.

3.1 Model 1 (Base Model Considering Size and Row Effectiveness)

Our first model simplifies the comprehensive case into a linear objective function, and only considers different product sizes and row effectiveness under the multiple cabinets' situation.

In this case, there are K cabinets (labeled as $k = 1, 2, ..., K$) with row length T available in the retail store, and the k^{th} cabinet has R_k rows (labeled as $i = 1, 2, ..., R_k$). The retailer wishes to display M products (labeled as $d = 1, 2, ..., M$), and the length for each product d is denoted as W_d. Moreover, we introduce a_i as the row effectiveness of row i, which captures the effect of the row on the potential revenue from the product placed in that row. The row effectiveness reaches its maximum in the middle rows, which are at eye level, and gradually decreases for higher or lower rows.

Figure 2 explains the above parameters visually.

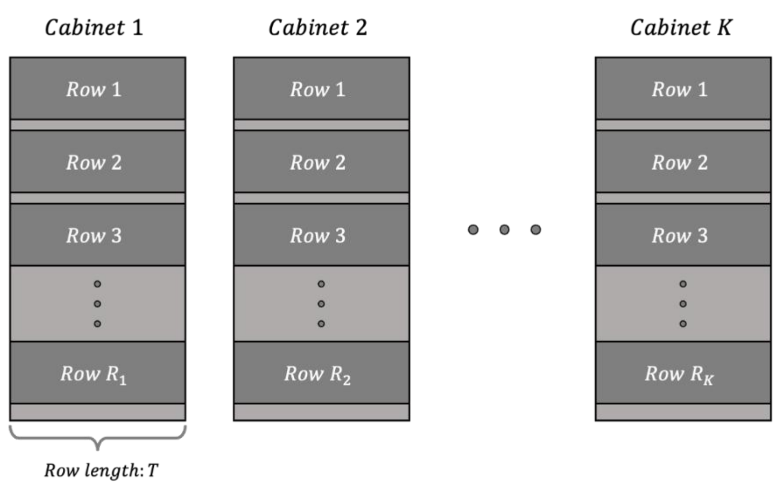


Figure 2. Illustration of Parameters

We summarize the parameters of the products and cabinet accordingly as shown below:

- K: the number of cabinets, denoted by $k = 1, 2, ..., K$
- T: the row length of each shelf for all cabinets
- i: the index for row in each cabinet k, $i = 1, ..., R_k$
- R_k: the total number of rows for each cabinet k
- a_i: the row effectiveness of row i
- M: the number of products to be displayed, denoted by $d = 1, ..., M$
- W_d: the face length for each product d
- r_d: the revenue potential for each product d, $d = 1, ..., M$

- U_d : the upper bound on the number of facings for product $d, d = 1, \ldots, M$ (U_d is calculated as the percentage of the sum of all revenue potentials that the product's revenue potential represents, $r_d/\sum_{h=1}^{M} r_h$)
- L_d: the lower bound on the number of facings for product $d, d = 1, \ldots, M$

With these settings, the decision variable is defined as X_{id}^k, which is the number of product d allocated to shelf i of cabinet k. The objective function is defined as follows:

$$\max_{x_{id}^k} \sum_{d=1}^{M} r_d \sum_{k=1}^{K} \sum_{i=1}^{R_k} a_i X_{id}^k \#(1)$$

subject to

$$\sum_{d=1}^{M} W_d X_{id}^k \leq T, i = 1,2, \ldots R_k, k = 1,2, \ldots K \#(2)$$

$$\sum_{k=1}^{K} \sum_{i=1}^{R_k} X_{id}^k \geq L_d, d = 1,2, \ldots M \#(3)$$

$$\sum_{k=1}^{K} \sum_{i=1}^{R_k} X_{id}^k \leq U_d, d = 1,2, \ldots M \#(4)$$

$$X_d^k \geq 0, \ integer, d = 1, \ldots, M, k = 1, \ldots, K \#(5)$$

It can be noted that in this formulation, we do not solve for the horizontal location of the product. While a complete solution would require accountability for horizontal product location, we choose to remove it from our consideration to simplify the problem, since the effect of the horizontal location of the product is minuscule in comparison to its vertical location.

We can see that the coefficient $r_d a_i$ together provide an estimate of the potential revenue by placing product d in row i.

Constraint (2) ensures that the products assigned to a shelf do not exceed the length of that shelf, whereas Constraints (3) and (4) ensure that the lower and upper bound constraints for the number of facings are met.

The problem involves assigning M products to $\sum_{k=1}^{K} R_k$ shelves to maximize the total revenue, with respect to the row length constraints as well as the number of facings constraints. According to Geismar et al. (2015), this formation is NP-hard and requires demanding computation power and memory for solving any problem of practical scale. Therefore, we break our problem into two subproblems with a two-stage model:

Stage 1 (SP1): Allocate products to cabinet
Stage 2 (SP2): Arrange units within cabinet

This decomposition enables us to divide the time-consuming multi-cabinet optimization issue into multiple more manageable optimization problems, one for each cabinet, which makes it a more efficient process.

3.1.1 Subproblem 1: Assign Products to Cabinets

The purpose of subproblem 1 is to decide how many facings each product should have in each cabinet based on their revenue potentials, without regarding the internal assignments within each cabinet. The objective function tries to maximize the number of units assigned to a cabinet for products with larger revenue potentials and minimize the units for products with smaller potentials while allocating products with similar revenue potentials equally across the cabinets. To solve this subproblem, we use the following decision variables:

- $Z_d^k = \begin{cases} 1, \ if\ product\ d\ is\ allocated\ to\ cabinet\ k; \\ 0, \ otherwise \end{cases}$
- X_d^k: the number of units of product d allocated to cabinet k.

With the above notation, our model formulation is as follows:

$$\max_{x_d^k,\ z_d^k} \sum_{d=1}^{m} \sum_{k=1}^{k} r_d\, x_d^k \ \#(6)$$

subject to:

$$\sum_{k=1}^{K} Z_d^k = 1 \ \#(7)$$

$$x_d^k \geq Z_d^k\, L_d \#(8)$$

$$x_d^k \leq \ Z_d^k\, U_d \#(9)$$

$$\sum_{d=1}^{M} x_d^k \leq \ TR_k, k = 1,2,3 \ldots K\#(10)$$

$$Z_d^k \in \{0, 1\},, d = 1, \ldots, M, k = 1, \ldots, K\#(11)$$

$$X_d^k \geq 0,\ integer, d = 1, \ldots, M, k = 1, \ldots, K \ \#(12)$$

The objective function (1) represents the total revenue of the display strategy without considering row effectiveness, which shows that each product's revenue potential is multiplied by its number of facings. Constraint (2) ensure that all the units for a specific product will be stored in exactly one cabinet. Constraints (3) and (4) enforce lower bounds and upper bounds for the number of slots allocated to a particular product respectively. Constraint (5) ensures that the sum of units of all products allotted to a particular cabinet does not exceed to shelf space available on that cabinet. Constraints (6) show the binary variable condition for Z_d^k and constraint (7) ensures that X_d^k has non-negative integers.

We denote the optimal solution as C_d, which is the optimal number of units of product d allocated to cabinet k. This solution will be used later as a parameter in Subproblem 2.

3.1.2 Subproblem 2: Arrange Units within Cabinets

Since the vertical location at which a product is displayed may be as significant as the amount of shelf space it occupies, we introduce the row effectiveness a_i to quantify the effect of vertical display location. The row effectiveness values reach the highest in the middle rows and decrease for rows closer to the top or bottom. Then the weighted revenue from displaying one unit of product d on row i for one period is defined as $a_i r_d$, where the concept "weighted" shows the influence of the row effectiveness values a_i.

Some previous papers on shelf-space allocation propose decreasing marginal revenue on the number of units displayed for a given product (Van Nierop et al. 2008, Murray et al. 2010). Our objective function for Subproblem 2 employs the row effectiveness values a_i to approximate this concavity while maintaining a linear objective to make the model tractable. Once the number of one product becomes larger, it should be placed over more rows, away from the middle rows to rows with lower effectiveness. Therefore, the product's total weighted revenue per unit displayed decreases.

Based on the illustration above and each product assigned to a cabinet in Subproblem 1, the purpose of Subproblem 2 is to arrange units within cabinets to ensure all slots of each cabinet are filled and the total weighted revenue is maximized.

To solve this subproblem for cabinet k, we use the following decision variables:

- X_{id}: the number of units of product d allocated on row i.

With the above notation, our model formulation is as follows:

$$\max_{X_{id}} \sum_{d=1}^{M} r_d \sum_{i=1}^{R_k} a_i X_{id} \#(13)$$

subject to:

$$C_d - 1 \leq \sum_{i=1}^{R_k} X_{id} \leq C_d + 1, d = 1,2,3\ldots M \#(14)$$

$$\sum_{d=1}^{m} w_d X_{id} \leq T, for\ i = 1, \ldots, R_k \#(15)$$

$$X_{id} \geq 0,\ integer, d = 1, \ldots, M, i = 1, \ldots, R_k \ \#(16)$$

The objective function (13) represents the weighted revenue of the display strategy for each cabinet considering row effectiveness. Constraint (14) ensures that the number of products we allocate in a cabinet for Subproblem 2 is equal to the optimal number of products we solved in Subproblem 1, but we add a relaxation to the constraint by converting it into an inequality in order to avoid cases where a fraction of a product is stored on the shelf. For example, consider the product has a facing length of 5, and the shelf has a length of 8. If SP1 suggests the optimal number of facings to be 3, it is therefore infeasible to place 3 units of products on 2 shelves. Instead, placing 2 or 4 units of products is more feasible. Constraint (15) ensures that the total face length for products in one row should not exceed the row length of each cabinet. Constraint (16) ensures that X_{id} is non-negative integers.

3.2 Model 2 (+ Contiguous Rectangle)

The display shown in Figure 2 is one solution that satisfies the constraints in Model 1 and maximizes total weighted revenue simply by placing the product with the largest revenue potential in the rows with the highest row effectiveness. However, this solution may lead to a cluttered display of products, making inventory management difficult and making it difficult for the customer to find the products they want. Hence, retailers would prefer to neatly arrange the same products in contiguous rectangles as shown in Figure 3.

C	C	C	B
D	A	A	A
D	C	C	B
B	B	C	A

Figure 2

C	C	C	D
C	C	C	D
B	B	A	A
B	B	A	A

Figure 3

Therefore, we introduce the concept of a *Contiguous Rectangle,* which is shown above in Figure 3, and construct Model 2 as an improvement on our Sub-Problem 2 (SP2) to account for it. The purpose of this formulation is to arrange each product assigned to a cabinet in a contiguous rectangle, so that all slots of each cabinet are filled, and the total weighted revenue is maximized.

We introduce binary variables Z_{id} to simplify our formulation, where Z_{id} is 1 if product *d* is assigned to shelf *i* and 0 otherwise. We use the following constraints to enforce this definition,

$$x_{id} \leq H \times Z_{id}, i = 1,2,3 \dots R_k, d = 1,2,3 \dots M\#(17)$$
$$x_{id} \geq Z_{id}, i = 1,2,3 \dots R_k, d = 1,2,3 \dots M\#(18)$$

where H is some large number.

We consider two steps to set the constraints for Contiguous Rectangle. Firstly, the number of facings on each shelf should be the same for each product:

$$X_{id} - X_{jd} \leq H \times (2 - Z_{id} - Z_{jd}), i, j = 1,2,3 \dots R_k\#(19)$$
$$X_{id} - X_{jd} \geq -H \times (2 - Z_{id} - Z_{jd}), i, j = 1,2,3 \dots R_k\#(20)1111\#(1)$$

Secondly, if a product is placed on multiple shelves, it should be on adjacent shelves:

$$i - j \leq \sum_i Z_{id} - 1 + H \times (2 - Z_{id} - Z_{jd}), d = 1,2,3 \dots M, i, j = 1,2,3 \dots R_k\#(21)$$

$$i - j \geq -\left[\sum_{i} Z_{id} - 1 + H \times \left(2 - Z_{id} - Z_{jd}\right)\right], d = 1,2,3 \ldots M, i, j = 1,2,3 \ldots R_k \#(22)$$

3.3 Model 3 (+ Synergy Effect)

In the retail setting, it is generally appreciable to allocate products of the same category to the same cabinet, as customers can perceive a clear category segmentation across cabinets. However, this may not be a stringent requirement as it can also be accounted for manually during the process of restocking simply by shuffling products on the same shelf. Therefore, it would be ideal to add a soft constraint to enforce such rules, without imposing a high negative impact on the total revenue.

The model is still formulated in a two-stage manner, with a slight alteration to the objective of SP1:

$$\max_{x_d^k,\, z_d^k} \sum_{d=1}^{M} \sum_{k=1}^{k} r_d\, x_d^k + \sum_{d_1=1}^{M} \sum_{d_2=1}^{M} \sum_{k=1}^{K} \frac{V_{d_1 d_2}^k}{2} e_{d_1 d_2} \#(23)$$

In this model, $V_{d_1 d_2}^k = \min\{X_{d_1}^k, X_{d_2}^k\}$. We divide this figure by 2 to avoid double counting. $e_{d_1 d_2}$ measures the synergy effect of product d_1 and product d_2, which is positive when d_1 and d_2 are from the same category and is zero when d_1 and d_2 are from the different category. Therefore, the objective function encourages, to the extent of the scale of $e_{d_1 d_2}$, products of the same category to be put in the same cabinet but does not penalize the model for not doing so. Hence, this implicitly acts as a soft constraint.

While this formulation of the model is not linear due to the existence of the $min()$ function, it can be made so by adding suitable inequalities to the constraints, the proof of which is left to the reader of this paper as an exercise.

The SP2 formulation remains the same as the formulation discussed for model 2 in section 3.2, which maximizes the total revenue on each shelf and ensures the products are placed in contiguous rectangles.

3.4 Model 4 (+ Cross-elasticity Effect)

Based on the previous models, if a different product that competes with the original product is given greater shelf space, the total revenue remains unchanged. But in reality, products within a section of retail shelf that are either complements or substitutes may have an influence on the demand of the other products, known as the *Cross-elasticity Effect*. Therefore, apart from the synergy effect discussed in Model 3, we suggest a new variable in Model 4 to take the cross-elasticity effect into consideration.

The model is also formulated in a two-stage manner, with an additional element to the objective function of Model 3:

$$\max_{x_d^k,\ z_d^k} \sum_{d=1}^{M}\sum_{k=1}^{k} r_d\, x_d^k + \sum_{d_1=1}^{M}\sum_{d_2=1}^{M}\sum_{k=1}^{K}\frac{V_{d_1 d_2}^k}{2} e_{d_1 d_2} + \sum_{d_1=1}^{M}\sum_{d_2=1}^{M}\sum_{k=1}^{K}\frac{V_{d_1 d_2}^k}{2} f_{d_1 d_2} \#(24)$$

where $f_{d_1 d_2}$ measures the cross-elasticity effect of product d_1 and product d_2, which is positive when d_1 and d_2 are complementary and is negative when d_1 and d_2 are substitutable.

4. Results

We considered the dataset based on the information regarding shelves and products. We utilize information on 7 shelves and since the cabinets are the same, we use the fact that they may contain 1 or more shelves. We proceeded to use this dataset which provided sufficient information on the products while maintaining the uniformity of the shelves. In summary, our dataset contains 7 cabinets and each of them contains 7 shelves. For the products side, we have 196 products which belong to 9 categories.

As mentioned previously, since we do not have enough time series data to estimate the cross elasticity accurately, we only consider the row effectiveness, synergy effect, and contiguous rectangle constraint in the implemented model.

The first stage takes 4 hours to get a result with a 2% gap, while the 7 second-stage models take 1 hour in total to get results with a gap of smaller than 0.5%. The computation times prove the need to break the original problem into two stages due to its complexity. While the detailed results can be found in the Appendix, we summarize the result from our stage 1 and a sample result from one of the stage 2 problems. For example, for category131, the model assigns 259 units to cabinet 5 and 10 units to cabinet 1. For each category, the products are placed in 1 or 2 cabinets, suggesting that the synergy effect in our model works as we expected, and the results can be further improved with longer optimization time and better estimation of the synergy effect coefficients.

category_id	cabinet_0	cabinet_1	cabinet_2	cabinet_3	cabinet_4	cabinet_5	cabinet_6
131	0	10	0	0	0	259	0
132	0	8	259	0	0	0	0
133	0	188	0	0	0	0	0
134	0	32	0	0	0	0	0
135	43	0	0	0	0	0	0
136	92	0	0	0	0	0	0
137	25	0	0	333	271	0	0
138	76	0	0	0	0	0	0
139	0	13	0	0	0	0	316

Figure 4. Number of Products of Different Categories to be Placed on Each Cabinet

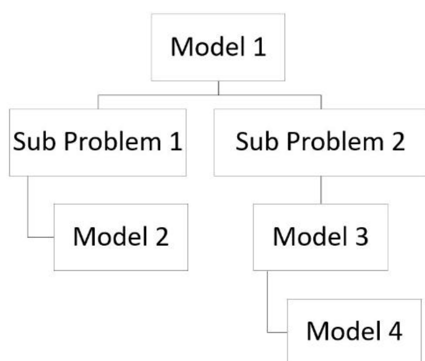
In the solution of the second stage problem for cabinet 3, we see that several products are assigned adjacent shelves, and in all such cases, the number of facings in each shelf is equal. For example, product 16 is assigned to 6 shelves, with 12 units per shelf. In this sense, all units of product 16 are placed in a contiguous rectangle. It implies that we can generate the desired optimal solution using the current model formulation.

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6	width	revenue	min_facing	max_facing	category_id
0	15	39	0	0	0	0	0	0	62	257	2	40	137
1	16	12	12	12	12	12	12	0	93	719	2	73	137
2	22	0	0	0	0	0	0	20	64	122	2	20	137
3	26	0	16	16	16	16	0	0	59	408	2	65	137
4	27	0	0	0	0	0	30	30	71	445	2	60	137
5	59	0	18	18	18	18	0	0	84	655	2	73	137
6	140	0	0	0	0	0	3	0	95	15	2	4	137

Figure 5. Number of Products to be Placed on Cabinet 3

5. Conclusion and Future Scope

In conclusion, we started with a complete model (Model 1) to capture the essence of the Shelf Space Allocation Problem. Facing its NP-hard formation, we broke it down into 2 stages, the computation of each of which is less demanding. From the basic two-stage formation, we began incorporating more realistic design in the retailing context, e.g., placing all units of a product in the contiguous rectangle (Model 2), placing all products of the same category on one cabinet (Model 3), considering the cross-elasticity between different products (Model 4).



We then discuss the feasibility and implementation of this model in Section 4, showing the results with a sample dataset. While we do not run model 4 due to computational constraints, results from the implementations of models 1 through 3 suggest that model 4 is also solvable given sufficient computing power and time.

However, the use of a heuristic algorithm to search for an optimal solution can help improve the implementation and help reach a solution easier. There have also been several simplifying assumptions made in our approach, which can be modified and improved upon using appropriate datasets and preprocessing techniques. Real data can be used to estimate cross elasticity, and additional constraints and models can be made to consider other factors such as horizontal elasticity, supply chain and inventory cycles, time horizons, etc. We can also consider alternative approaches, such as using a maximum-weight independent set (MWIS) problem framework on a network to model the contiguous rectangle constraints, as suggested by Geismar et al. [2014], which may be more efficient that the current formulation.

6. References

[1] Chandon, P., Hutchinson, J. W., Bradlow, E. T., & Young, S. H. (2009). Does In-Store Marketing Work? Effects of the Number and Position of Shelf Facings on Brand Attention and Evaluation at the Point of Purchase. Journal of Marketing, 73(6), 1–17. https://doi.org/10.1509/jmkg.73.6.1

[2] Van Nierop, E., D. Fok, P. H. Franses. 2008. Interaction between shelf layout and marketing effectiveness and its impact on optimizing shelf arrangements. *Manage. Sci*. 27(6): 1065–1082.

[3] Murray, C. C., D. Talukdar, A. Gosavi. 2010. Joint optimization of product price, display orientation and shelf-space allocation in retail category management. *J. Retail*. 86(2): 125–136.

[4] Geismar, H. N., Dawande, M., Murthi., B. P. S., Sriskandarajah, C. 2014. Maximizing Revenue Through Two-Dimensional Shelf-Space Allocation. *Production and Operations Management, 24*(7), 1148-1163. https://doi.org/10.1111/poms.12316.

[5] M.-H. Yang and W.-C. Chen. A study on shelf space allocation and management. Interna tional Journal of Production Economics, 61(510):309–317, 1999.

[6] R. C. Curhan. The Relationship Between Shelf Space and Unit Sales in Supermarkets. Journal of Maketing Research, 9(4):406–412, 1972.

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

Appendix

November 19, 2022

The full dataset can be found in https://github.com/gamma-opt/ShelfSpaceAllocation.jl

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
import os
os.getcwd()
os.chdir("/content/drive/MyDrive/DBA5103 final project")
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

```
Mounted at /content/drive
```

[26]:
```
print("The number of products of different categories to be placed on each␣
 ↪cabinet")
first_stage=pd.read_excel("/content/drive/MyDrive/DBA5103 final project/
 ↪result_imputed_data/first_res.xlsx")
first_stage.groupby("category_id").agg({
    "cabinet_0":sum,
    "cabinet_1":sum,
    "cabinet_2":sum,
    "cabinet_3":sum,
    "cabinet_4":sum,
    "cabinet_5":sum,
    "cabinet_6":sum,
    "product_id":"nunique"

})
```

```
The number of products of different categories to be placed on each cabinet
```

[26]:
```
             cabinet_0  cabinet_1  cabinet_2  cabinet_3  cabinet_4  cabinet_5  \
category_id
131                  0         10          0          0          0        259
132                  0          8        259          0          0          0
133                  0        188          0          0          0          0
134                  0         32          0          0          0          0
135                 43          0          0          0          0          0
136                 92          0          0          0          0          0
```

137	25	0	0	333	271	0
138	76	0	0	0	0	0
139	0	13	0	0	0	0

	cabinet_6	product_id
category_id		
131	0	58
132	0	57
133	0	9
134	0	3
135	0	4
136	0	5
137	0	28
138	0	6
139	316	23

[27]:
```
import warnings
warnings.filterwarnings('ignore')
for i in range(7):
  print("-"*200)
  print("-"*200)
  print("-"*200)
  print("-"*200)
  print("Number of products to be placed on cabinet_{} at different shelves".
 ↪format(i))
  pd.read_excel("/content/drive/MyDrive/DBA5103 final project/
 ↪result_imputed_data/cabinet_{}.xlsx".format(i)).drop(["Unnamed: 0"],1)
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
Number of products to be placed on cabinet_0 at different shelves.
```

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6 \
0	0	0	0	0	14	14	0	0
1	12	4	0	0	0	0	0	0
2	14	2	0	0	0	0	0	0
3	19	8	0	0	0	0	0	0

4	20	0	0	0	12	12	0	0
5	23	0	0	0	0	0	0	10
6	24	5	0	0	0	0	0	0
7	28	0	0	0	0	0	0	5
8	41	0	0	0	0	0	0	2
9	54	0	0	0	0	0	0	7
10	58	0	0	0	0	0	11	0
11	66	0	0	0	8	8	8	0
12	67	0	15	0	0	0	0	0
13	70	0	0	29	0	0	0	0
14	90	0	0	17	0	0	0	0
15	92	0	9	0	0	0	0	0
16	97	0	0	0	0	0	0	4
17	121	0	0	0	0	0	9	0
18	147	0	0	0	0	0	11	0
19	154	4	4	0	0	0	0	0
20	177	2	0	0	0	0	0	0

	width	revenue	min_facing	max_facing	category_id
0	145	418	2	29	136
1	140	57	2	6	137
2	149	41	2	5	137
3	135	79	2	8	135
4	81	200	2	25	135
5	136	109	2	10	137
6	146	67	2	7	137
7	137	64	2	7	137
8	147	39	2	5	135
9	97	48	2	7	136
10	75	73	2	11	138
11	69	165	2	24	138
12	136	188	2	15	138
13	62	179	2	29	136
14	101	156	2	17	136
15	103	90	2	10	138
16	140	33	2	5	138
17	103	88	2	10	136
18	114	104	2	11	138
19	145	81	2	8	135
20	140	36	2	5	137

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
Number of products to be placed on cabinet_1 at different shelves.
```

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6 \
0	11	0	7	7	0	0	0	0
1	29	0	0	0	0	0	0	8
2	30	0	0	0	0	0	0	9
3	33	0	0	0	0	0	14	0
4	34	0	0	22	0	0	0	0
5	35	0	0	0	49	0	0	0
6	65	0	16	16	0	0	0	0
7	81	2	0	0	0	0	0	0
8	99	3	0	0	0	0	0	0
9	106	2	0	0	0	0	0	0
10	116	2	0	0	0	0	0	0
11	118	2	0	0	0	0	0	0
12	128	0	0	0	0	0	0	2
13	130	0	0	0	0	0	0	2
14	151	2	0	0	0	0	0	0
15	161	8	0	0	0	0	0	0
16	168	2	0	0	0	0	0	0
17	169	2	0	0	0	0	0	0
18	178	0	0	0	0	8	8	0
19	179	0	12	0	0	0	0	0
20	180	0	0	0	0	0	10	0
21	181	0	0	0	0	0	0	9
22	182	0	0	0	0	21	0	0
23	190	2	0	0	0	0	0	0

	width	revenue	min_facing	max_facing	category_id
0	51	71	2	15	134
1	85	47	2	8	134
2	149	112	2	9	134
3	116	144	2	14	133
4	57	120	2	22	133
5	73	370	2	49	133
6	116	384	2	33	133
7	145	23	2	4	132
8	110	27	2	5	131
9	111	27	2	5	131
10	128	31	2	5	131
11	133	22	2	4	139

12	125	15	2	4	131
13	134	34	2	5	131
14	146	47	2	5	139
15	132	78	2	8	139
16	141	24	2	4	132
17	140	28	2	4	132
18	74	108	2	16	133
19	100	115	2	13	133
20	128	115	2	11	133
21	116	80	2	9	133
22	140	287	2	21	133
23	147	28	2	4	132

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
Number of products to be placed on cabinet_2 at different shelves.
```

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6 \
0	2	0	0	0	6	0	0	0
1	3	0	0	0	7	0	0	0
2	4	0	0	6	0	0	0	0
3	5	0	6	0	0	0	0	0
4	6	0	0	6	0	0	0	0
5	7	0	0	0	0	6	0	0
6	8	0	0	6	0	0	0	0
7	9	0	0	0	6	0	0	0
8	31	0	0	0	6	0	0	0
9	36	0	0	0	0	0	0	4
10	37	4	0	0	0	0	0	0
11	38	0	0	0	0	0	0	4
12	39	0	0	0	6	0	0	0
13	53	0	0	0	6	0	0	0
14	55	0	0	0	0	3	3	0
15	56	0	0	0	0	0	4	0
16	57	0	4	0	0	0	0	0
17	60	0	6	0	0	0	0	0
18	61	0	0	0	0	0	0	4

19	62	0	0	0	0	6	0	0
20	71	4	0	0	0	0	0	0
21	72	0	0	0	0	0	0	4
22	73	4	0	0	0	0	0	0
23	74	2	0	0	0	0	0	0
24	75	0	0	0	0	6	0	0
25	76	0	0	0	0	0	4	0
26	77	0	0	0	0	0	2	2
27	78	4	0	0	0	0	0	0
28	79	0	6	0	0	0	0	0
29	80	5	0	0	0	0	0	0
30	82	0	0	0	6	0	0	0
31	84	0	0	0	0	6	0	0
32	85	0	0	0	6	0	0	0
33	86	0	0	6	0	0	0	0
34	87	0	6	0	0	0	0	0
35	88	0	0	0	0	0	0	4
36	89	0	0	0	0	0	6	0
37	122	0	0	0	0	6	0	0
38	141	0	0	0	0	0	4	0
39	142	0	0	0	0	0	0	4
40	143	4	0	0	0	0	0	0
41	144	0	0	6	0	0	0	0
42	145	2	0	0	0	0	0	0
43	146	0	0	0	0	0	5	0
44	148	0	0	0	0	6	0	0
45	167	0	0	0	0	0	4	0
46	170	0	0	0	0	6	0	0
47	171	0	4	0	0	0	0	0
48	183	0	0	0	0	0	0	3
49	184	0	0	0	0	0	4	0
50	189	0	6	0	0	0	0	0
51	191	0	0	6	0	0	0	0
52	192	0	0	0	0	0	4	0

	width	revenue	min_facing	max_facing	category_id
0	63	144	2	23	132
1	64	144	2	23	132
2	77	84	2	12	132
3	86	60	2	9	132
4	127	132	2	12	132
5	108	108	2	12	132
6	121	96	2	10	132
7	122	156	2	14	132
8	77	105	2	15	132
9	124	43	2	6	132
10	142	44	2	5	132

11	133	55	2	6	132
12	62	100	2	17	132
13	88	132	2	16	132
14	97	72	2	9	132
15	116	70	2	8	132
16	146	70	2	7	132
17	98	60	2	8	132
18	100	36	2	6	132
19	70	54	2	10	132
20	135	48	2	6	132
21	116	36	2	5	132
22	86	32	2	6	132
23	118	25	2	4	132
24	55	48	2	11	132
25	69	40	2	8	132
26	142	60	2	6	132
27	129	48	2	6	132
28	87	60	2	9	132
29	112	48	2	6	132
30	51	60	2	13	132
31	81	72	2	11	132
32	58	72	2	14	132
33	122	96	2	10	132
34	98	72	2	9	132
35	131	43	2	6	132
36	93	58	2	8	132
37	55	60	2	13	132
38	95	48	2	7	132
39	114	44	2	6	132
40	135	48	2	6	132
41	86	72	2	10	132
42	140	38	2	5	132
43	69	48	2	9	132
44	59	48	2	10	132
45	63	32	2	7	132
46	121	112	2	11	132
47	104	48	2	7	132
48	143	40	2	5	132
49	108	48	2	7	132
50	61	45	2	9	132
51	55	50	2	11	132
52	75	51	2	9	132

```
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
Number of products to be placed on cabinet_3 at different shelves.
```

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6	\
0	15	39	0	0	0	0	0	0	
1	16	12	12	12	12	12	12	0	
2	22	0	0	0	0	0	0	20	
3	26	0	16	16	16	16	0	0	
4	27	0	0	0	0	0	30	30	
5	59	0	18	18	18	18	0	0	
6	140	0	0	0	0	0	3	0	

	width	revenue	min_facing	max_facing	category_id
0	62	257	2	40	137
1	93	719	2	73	137
2	64	122	2	20	137
3	59	408	2	65	137
4	71	445	2	60	137
5	84	655	2	73	137
6	95	15	2	4	137

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
Number of products to be placed on cabinet_4 at different shelves.
```

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6	\
0	1	0	0	19	19	0	0	0	
1	10	9	0	0	0	0	0	0	
2	13	6	0	0	0	0	0	0	
3	25	9	0	0	0	0	0	0	
4	32	0	0	0	0	0	0	15	

5	63	0	0	0	0	0	30	0
6	68	0	0	19	19	0	0	0
7	69	0	0	0	0	35	0	0
8	166	0	18	0	0	0	0	0
9	172	11	0	0	0	0	0	0
10	173	0	0	0	0	0	0	14
11	175	0	15	0	0	0	0	0
12	176	0	0	0	0	0	0	5
13	185	0	4	4	4	0	0	0
14	186	0	0	0	0	0	0	12

	width	revenue	min_facing	max_facing	category_id
0	114	453	2	39	137
1	86	73	2	10	137
2	100	56	2	8	137
3	123	90	2	9	137
4	82	111	2	15	137
5	118	349	2	30	137
6	61	236	2	38	137
7	99	350	2	35	137
8	126	218	2	18	137
9	99	87	2	11	137
10	64	78	2	14	137
11	64	87	2	15	137
12	112	65	2	8	137
13	55	59	2	12	137
14	73	75	2	12	137

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
```

Number of products to be placed on cabinet_5 at different shelves.

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6 \
0	17	0	0	6	0	0	0	0
1	18	0	0	0	0	6	0	0
2	21	3	0	0	0	0	0	0
3	40	0	0	0	0	0	6	0

4	42	0	0	0	0	0	2	2
5	43	0	4	0	0	0	0	0
6	44	0	0	0	0	6	0	0
7	45	0	0	0	0	6	0	0
8	46	0	0	6	0	0	0	0
9	47	0	0	6	0	0	0	0
10	48	0	5	0	0	0	0	0
11	49	0	3	3	0	0	0	0
12	50	0	0	0	0	0	0	5
13	51	0	0	0	0	0	4	0
14	52	4	0	0	0	0	0	0
15	95	0	0	0	0	3	3	0
16	96	0	0	6	0	0	0	0
17	100	0	0	0	0	0	0	2
18	101	4	0	0	0	0	0	0
19	102	0	0	0	6	0	0	0
20	103	0	6	0	0	0	0	0
21	104	0	0	0	0	0	0	3
22	105	0	0	0	0	6	0	0
23	107	0	0	0	0	0	6	0
24	108	4	0	0	0	0	0	0
25	109	0	0	0	0	0	0	4
26	110	0	0	0	0	0	4	0
27	111	4	0	0	0	0	0	0
28	112	0	0	0	6	0	0	0
29	113	0	0	0	0	0	6	0
30	114	0	0	0	0	0	0	4
31	115	3	0	0	0	0	0	0
32	117	0	0	0	6	0	0	0
33	119	0	0	0	0	0	4	0
34	120	0	6	0	0	0	0	0
35	123	0	4	0	0	0	0	0
36	124	0	0	6	0	0	0	0
37	125	0	0	0	0	5	0	0
38	126	0	0	3	3	0	0	0
39	127	4	0	0	0	0	0	0
40	129	0	0	0	0	6	0	0
41	131	0	0	0	0	0	0	4
42	132	0	0	0	6	0	0	0
43	133	0	4	0	0	0	0	0
44	134	0	0	3	3	0	0	0
45	135	4	0	0	0	0	0	0
46	136	0	0	0	3	3	0	0
47	137	0	0	0	0	0	0	3
48	138	0	0	0	0	0	0	4
49	139	4	0	0	0	0	0	0
50	155	4	0	0	0	0	0	0

51	187	0	0	0	6	0	0	0
52	188	0	0	0	0	0	0	5

	width	revenue	min_facing	max_facing	category_id
0	94	76	2	10	131
1	80	80	2	12	131
2	129	51	2	6	131
3	106	72	2	9	131
4	148	71	2	7	131
5	114	57	2	7	131
6	109	111	2	12	131
7	147	138	2	11	131
8	104	90	2	10	131
9	126	90	2	9	131
10	137	80	2	8	131
11	67	48	2	9	131
12	102	47	2	7	131
13	66	38	2	8	131
14	99	38	2	6	131
15	74	52	2	9	131
16	64	57	2	11	131
17	113	38	2	6	131
18	101	38	2	6	131
19	99	169	2	18	131
20	91	62	2	9	131
21	111	41	2	6	131
22	73	54	2	9	131
23	87	54	2	8	131
24	74	31	2	6	131
25	116	47	2	6	131
26	73	47	2	8	131
27	111	52	2	7	131
28	96	128	2	15	131
29	137	96	2	9	131
30	97	36	2	6	131
31	102	31	2	5	131
32	94	128	2	15	131
33	132	63	2	7	131
34	117	68	2	8	131
35	133	73	2	8	131
36	59	48	2	10	131
37	57	47	2	10	131
38	148	193	2	14	131
39	94	33	2	6	131
40	72	64	2	11	131
41	75	22	2	5	131
42	121	191	2	17	131

43	112	60	2	7	131
44	70	86	2	14	131
45	68	31	2	7	131
46	53	76	2	16	131
47	109	41	2	6	131
48	77	31	2	6	131
49	106	48	2	7	131
50	70	23	2	6	131
51	51	73	2	16	131
52	86	41	2	7	131

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
Number of products to be placed on cabinet_6 at different shelves.
```

[27]:

	product_id	shelf_0	shelf_1	shelf_2	shelf_3	shelf_4	shelf_5	shelf_6	\
0	64	0	0	0	0	0	25	0	
1	83	0	0	0	45	0	0	0	
2	91	0	0	15	15	0	0	0	
3	93	0	0	0	0	0	0	12	
4	94	0	0	0	0	0	0	2	
5	98	4	0	0	0	0	0	0	
6	149	5	0	0	0	0	0	0	
7	150	6	0	0	0	0	0	0	
8	152	0	0	0	0	0	0	3	
9	153	1	1	0	0	0	0	0	
10	156	12	0	0	0	0	0	0	
11	157	0	0	0	0	26	0	0	
12	158	0	0	31	0	0	0	0	
13	159	0	0	0	0	0	22	0	
14	160	0	0	0	0	0	0	11	
15	162	0	15	0	0	0	0	0	
16	163	0	14	14	0	0	0	0	
17	164	0	22	0	0	0	0	0	
18	165	7	0	0	0	0	0	0	
19	174	4	0	0	0	0	0	0	

	width	revenue	min_facing	max_facing	category_id
0	82	199	2	25	139
1	55	256	2	45	139
2	64	193	2	30	139
3	130	128	2	12	139
4	99	28	2	5	139
5	60	19	2	5	139
6	56	19	2	6	139
7	80	31	2	6	139
8	125	67	2	7	139
9	113	52	2	7	139
10	104	109	2	12	139
11	137	369	2	27	139
12	54	168	2	31	139
13	69	144	2	22	139
14	128	119	2	11	139
15	65	88	2	15	139
16	64	182	2	28	139
17	70	148	2	22	139
18	139	103	2	9	139
19	62	13	2	5	139

model

November 20, 2022

Detail instruction to install gurobipy in colabb https://support.gurobi.com/hc/en-us/articles/4409582394769-Google-Colab-Installation-and-Licensing

[2]:
```
import os
os.getcwd()
```

[2]:
```
'C:\\Users\\33321\\Desktop\\NUS\\Operation\\FinalProject'
```

[ ]:
```
from gurobipy import *
import numpy as np
import pandas as pd
pd.set_option('display.max_columns', None)
```

[ ]:
```
products=pd.read_csv("./data/products.csv")
shelf=pd.read_csv("./data/shelves.csv")
```

[ ]:
```
shelf.drop(["Unnamed:␣
↪0","product_min_unit_weight","product_max_unit_weight"],1,inplace=True)
```

[ ]:
```
shelf.level=7
shelf["total_length"]=shelf.total_width*shelf.level
```

[ ]:
```
shelf
```

[ ]:
```
products
```

[ ]:
```
# the min max facing provided by the dataset is too small
products["all_shelf_length"]=shelf["total_length"].sum()

products["revenue"]=products.monthly_demand*products.price

products["min_facing"]=2
products["max_facing"]=np.ceil(products["all_shelf_length"]*products["revenue"]/
↪products["revenue"].sum()/products.width)+2
products["product_id"]=products.index
```

[ ]:
```python
# id of cabinet are same, replace them
shelf["id"]="cabinet_"+shelf.index.astype("str")
cabinet_id=shelf["id"].values
total_length=shelf["total_length"].values
```

[ ]:
```python
shelf
```

[ ]:
```python
revenue_info=products[["product_id","width","revenue","min_facing","max_facing","category_id"]
revenue_info
```

[ ]:
```python
sum(revenue_info.width*revenue_info.max_facing)/3600/7
```

[ ]:
```python
product_id=revenue_info.product_id.values
revenue=revenue_info.revenue.values
min_facing=revenue_info.min_facing.values
max_facing=revenue_info.max_facing.values
product_width=revenue_info.width.values
```

[ ]:
```python
num_product=len(product_id)
num_cabinet=len(cabinet_id)
```

[ ]:
```python
# cross elasticity cofficients
e=np.array([np.array([1]*num_product)]*num_product)
for i in range(num_product):
    for j in range(num_product):
        if  revenue_info.loc[revenue_info.product_id==i,"category_id"].values␣
 ↪== revenue_info.loc[revenue_info.product_id==j,"category_id"].values:
            e[i,j]=(revenue[i]+revenue[j])/2*0.5
        else:
            e[i,j]=-(revenue[i]+revenue[j])/2*0.5
```

[ ]:
```python

```

[ ]:
```python
first = Model("First Stage_with_cross_elasticity",env=env)

# the number of facing of product i at cabinet k
x = first.addVars(num_product,num_cabinet,vtype=GRB.INTEGER , name = "x_ik")

# the Vijk of cross elasticity of product i and j at cabinet k
v = first.addVars(num_product,num_product,num_cabinet, name = "v_ijk")

# whether product i is placed on cabinet k
z = first.addVars(num_product,num_cabinet,vtype=GRB.BINARY, name = "z_ik")

first.setObjective(quicksum(revenue[i]*x[i,k] for i in range(num_product) for k␣
 ↪in range(num_cabinet))+\
```

```
    quicksum(v[i,j,k]*e[i,j]/2 for i in range(num_product) for j in␣
 ↪range(num_product) for k in range(num_cabinet)), GRB.MAXIMIZE)

#first.setObjective(quicksum(revenue[i]*x[i,k] for i in range(num_product) for␣
 ↪k in range(num_cabinet)), GRB.MAXIMIZE)

# make sure one product is only placed in one cabinet
first.addConstrs(( quicksum(z[i,k] for k in range(num_cabinet)) == 1 for i in␣
 ↪range(num_product) ))

# max facing
first.addConstrs((x[i,k]<=z[i,k]*max_facing[i] for i in range(num_product) for␣
 ↪k in range(num_cabinet)))

# min facing
first.addConstrs((x[i,k]>=z[i,k]*min_facing[i] for i in range(num_product) for␣
 ↪k in range(num_cabinet)))

# vijk
first.addConstrs((v[i,j,k]<=x[i,k] for i in range(num_product) for j in␣
 ↪range(num_product) for k in range(num_cabinet)))

first.addConstrs((v[i,j,k]<=x[j,k] for i in range(num_product) for j in␣
 ↪range(num_product) for k in range(num_cabinet)))


# the total length of products within one cabinet does not exceed its total␣
 ↪length
first.addConstrs((quicksum(x[i,k]*product_width[i] for i in␣
 ↪range(num_product))<=total_length[k] for k in range(num_cabinet) ))
```

[ ]:
```
# first = Model("First Stage")

# # the number of facing of product i at cabinet k
# x = first.addVars(num_product,num_cabinet,vtype=GRB.INTEGER , name = "x_ik")

# # whether product i is placed on cabinet k
# z = first.addVars(num_product,num_cabinet,vtype=GRB.BINARY, name = "z_ik")

# first.setObjective(quicksum(revenue[i]*x[i,k] for i in range(num_product) for␣
 ↪k in range(num_cabinet)), GRB.MAXIMIZE)

# # make sure one product is only placed in one cabinet
# first.addConstrs(( quicksum(z[i,k] for k in range(num_cabinet)) == 1 for i in␣
 ↪range(num_product) ))
```

```
# # max facing
# first.addConstrs((x[i,k]<=z[i,k]*max_facing[i] for i in range(num_product)␣
↪for k in range(num_cabinet)))

# # min facing
# first.addConstrs((x[i,k]>=z[i,k]*min_facing[i] for i in range(num_product)␣
↪for k in range(num_cabinet)))

# # the total length of products within one cabinet does not exceed its total␣
↪length
# first.addConstrs((quicksum(x[i,k]*product_width[i] for i in␣
↪range(num_product))<=total_length[k] for k in range(num_cabinet) ))
```

[ ]:
```
softlimit = 5
hardlimit = 60*60*4

def softtime(model, where):
    if where == GRB.Callback.MIP:
        runtime = model.cbGet(GRB.Callback.RUNTIME)
        objbst = model.cbGet(GRB.Callback.MIP_OBJBST)
        objbnd = model.cbGet(GRB.Callback.MIP_OBJBND)
        gap = abs((objbst - objbnd) / objbst)

        if runtime > softlimit and gap <= 0.1/100:
            model.terminate()

def save_model(m):
    gv = m.getVars()
    names = m.getAttr('VarName', gv)
    for i in range(m.SolCount):
        m.params.SolutionNumber = i
        xn = m.getAttr('Xn', gv)
        lines = ["{} {}".format(v1, v2) for v1, v2 in zip(names, xn)]
        with open('{}_{}.sol'.format(m.ModelName, i), 'w') as f:
            f.write("# Solution for model {}\n".format(m.modelName))
            f.write("# Objective value = {}\n".format(m.PoolObjVal))
            f.write("\n".join(lines))
```

first.setParam('TimeLimit', hardlimit) first.optimize(softtime)

[ ]:
```
first_res=pd.DataFrame({"Product_Id":product_id})
for i in cabinet_id:
    first_res[i]=0
```

[ ]:
```
for v in x:
    if x[v].x > 0:
        #print(x[v].VarName, x[v].x)
```

```
        first_res.loc[first_res.Product_Id==v[0],"cabinet_{}".
 ↪format(v[1])]=x[v].x
```

[ ]:
```
first_res=first_res.
 ↪merge(revenue_info[["product_id","category_id"]],left_on="Product_Id",right_on="product_id"
```

[ ]:
```
first_res.to_excel("./result_imputed_data_0.5/first_res.xlsx",index=False)
```

[ ]:
```
first_res=pd.read_excel("./result_imputed_data/first_res.xlsx")
```

[ ]:
```
first_res.groupby("category_id").agg({
    "cabinet_0":sum,
    "cabinet_1":sum,
    "cabinet_2":sum,
    "cabinet_3":sum,
    "cabinet_4":sum,
    "cabinet_5":sum,
    "cabinet_6":sum,
    "product_id":"nunique"

})
```

[ ]:
```
effectiveness={
    "cabinet_0":[1,2,3,4,3,2,1],
    "cabinet_1":[1,2,3,4,3,2,1],
    "cabinet_2":[1,2,3,4,3,2,1],
    "cabinet_3":[1,2,3,4,3,2,1],
    "cabinet_4":[1,2,3,4,3,2,1],
    "cabinet_5":[1,2,3,4,3,2,1],
    "cabinet_6":[1,2,3,4,3,2,1],
}
```

[ ]:
```
cabinet="cabinet_0"
product_within_cabinet=list(first_res[first_res[cabinet]!=0].Product_Id.values)
num_product_within_cabinet=list(first_res[first_res[cabinet]!=0][cabinet].
 ↪values)

num_shelf=int(shelf.loc[shelf["id"]==cabinet,"level"].values[0])
shelf_length=float(shelf.loc[shelf["id"]==cabinet,"total_width"].values[0])

row_eff=effectiveness[cabinet]

temp_num_product=len(product_within_cabinet)
num_product_within_cabinet=list(first_res[first_res[cabinet]!=0][cabinet].
 ↪values)
temp_product_width=product_width[product_within_cabinet]
temp_revenue=revenue[product_within_cabinet]
```

```
temp_max_facing=max_facing[product_within_cabinet]
temp_min_facing=min_facing[product_within_cabinet]

huge_number=sum(num_product_within_cabinet)
```

[ ]:
```
second = Model("Second Stage",env=env)

# the number of facing of product i at shelf j

x = second.addVars(temp_num_product,num_shelf, vtype=GRB.INTEGER,name = "x_ij")


# whether product i is placed on shelf j
z = second.addVars(num_product,num_cabinet,vtype=GRB.BINARY, name = "z_ij")

second.setObjective(quicksum(row_eff[j]*temp_revenue[i]*x[i,j] for i in␣
↪range(temp_num_product) for j in range(num_shelf)), GRB.MAXIMIZE)

# make sure z_ij works

second.addConstrs(( x[i,j]<=z[i,j]*huge_number for i in range(temp_num_product)␣
↪ for j in range(num_shelf) ))

# upper bound for a product on the same shelf?
#second.addConstrs(( x[i,j]<=15 for i in range(temp_num_product)  for j in␣
↪range(num_shelf) ))

second.addConstrs(( x[i,j]>=z[i,j] for i in range(temp_num_product)  for j in␣
↪range(num_shelf) ))


# The solutions generated by first stage may not be feasible in second stage,␣
↪allow 1 unit change
second.addConstrs(( quicksum(x[i,j] for j in␣
↪range(num_shelf))<=min(temp_max_facing[i],num_product_within_cabinet[i]+1)␣
↪for i in range(temp_num_product) ))

second.addConstrs(( quicksum(x[i,j] for j in␣
↪range(num_shelf))>=max(num_product_within_cabinet[i]-1,temp_min_facing[i]) ␣
↪for i in range(temp_num_product) ))

second.addConstrs(( quicksum(x[i,j]*temp_product_width[i] for i in␣
↪range(temp_num_product))<=shelf_length  for j in range(num_shelf) ))

#  contiguous rectangle
```

```
# if product j is placed on different shelves ,the number of facings at each␣
↪shelf is equal
second.addConstrs(( x[i,j]-x[i,j_2]<=(2-z[i,j]-z[i,j_2])*huge_number for i in␣
↪range(temp_num_product) for j_2 in range(num_shelf) for j in␣
↪range(num_shelf) ))
second.addConstrs(( x[i,j]-x[i,j_2]>=-(2-z[i,j]-z[i,j_2])*huge_number for i in␣
↪range(temp_num_product) for j_2 in range(num_shelf) for j in␣
↪range(num_shelf) ))



# make sure the rectangle is contiguous
second.addConstrs(((j-j_2<=(quicksum(z[i,shelf] for shelf in range(num_shelf))␣
↪)-1+huge_number*(2-z[i,j]-z[i,j_2]) \
    for j in range(num_shelf) for j_2 in range(num_shelf) for i in␣
↪range(temp_num_product)))

second.addConstrs(((j-j_2>=-((quicksum(z[i,shelf] for shelf in␣
↪range(num_shelf)))-1+huge_number*(2-z[i,j]-z[i,j_2])) \
    for j in range(num_shelf) for j_2 in range(num_shelf) for i in␣
↪range(temp_num_product)))
```

[ ]:
```
second.setParam('TimeLimit', hardlimit)
second.optimize(softtime)
```

[ ]:
```
second_res=pd.DataFrame({"Product_Id":product_within_cabinet})
for i in range(num_shelf):
    second_res["shelf_{}".format(i)]=0
```

[ ]:
```
for v in x:
    if x[v].x > 0:
        #print(x[v].VarName, x[v].x)
        second_res.loc[second_res.
↪Product_Id==product_within_cabinet[v[0]],"shelf_{}".format(v[1])]=x[v].x
second_res
```

[ ]:
```
def second_stage(cabinet):
    product_within_cabinet=list(first_res[first_res[cabinet]!=0].Product_Id.
↪values)
    num_product_within_cabinet=list(first_res[first_res[cabinet]!=0][cabinet].
↪values)

    num_shelf=int(shelf.loc[shelf["id"]==cabinet,"level"].values[0])
    shelf_length=float(shelf.loc[shelf["id"]==cabinet,"total_width"].values[0])

    row_eff=effectiveness[cabinet]
```

```
    temp_num_product=len(product_within_cabinet)
    num_product_within_cabinet=list(first_res[first_res[cabinet]!=0][cabinet].values)
    temp_product_width=product_width[product_within_cabinet]
    temp_revenue=revenue[product_within_cabinet]
    temp_max_facing=max_facing[product_within_cabinet]
    temp_min_facing=min_facing[product_within_cabinet]

    second = Model("Second Stage",env=env)

    # the number of facing of product i at shelf j

    x = second.addVars(temp_num_product,num_shelf, vtype=GRB.INTEGER,name = "x_ij")


    # whether product i is placed on shelf j
    z = second.addVars(num_product,num_cabinet,vtype=GRB.BINARY, name = "z_ij")

    second.setObjective(quicksum(row_eff[j]*temp_revenue[i]*x[i,j] for i in range(temp_num_product) for j in range(num_shelf)), GRB.MAXIMIZE)

    # make sure z_ij works

    second.addConstrs(( x[i,j]<=z[i,j]*huge_number for i in range(temp_num_product)  for j in range(num_shelf) ))
    second.addConstrs(( x[i,j]>=z[i,j] for i in range(temp_num_product)  for j in range(num_shelf) ))


    # The solutions generated by first stage may not be feasible in second stage, allow 1 unit change
    second.addConstrs(( quicksum(x[i,j] for j in range(num_shelf))<=min(temp_max_facing[i],num_product_within_cabinet[i]+1) for i in range(temp_num_product) ))

    second.addConstrs(( quicksum(x[i,j] for j in range(num_shelf))>=max(num_product_within_cabinet[i]-1,temp_min_facing[i])  for i in range(temp_num_product) ))

    second.addConstrs(( quicksum(x[i,j]*temp_product_width[i] for i in range(temp_num_product))<=shelf_length  for j in range(num_shelf) ))

    #  contiguous rectangle
    # if product j is placed on different shelves ,the number of facings at each shelf is equal
```

```
    second.addConstrs(( x[i,j]-x[i,j_2]<=(2-z[i,j]-z[i,j_2])*huge_number for i␣
↪in range(temp_num_product) for j_2 in range(num_shelf) for j in␣
↪range(num_shelf) ))
    second.addConstrs(( x[i,j]-x[i,j_2]>=-(2-z[i,j]-z[i,j_2])*huge_number for i␣
↪in range(temp_num_product) for j_2 in range(num_shelf) for j in␣
↪range(num_shelf) ))

    # make sure the rectangle is contiguous
    second.addConstrs((j-j_2<=quicksum(z[i,shelf] for shelf in␣
↪range(num_shelf))-1+huge_number*(2-z[i,j]-z[i,j_2]) \
        for j in range(num_shelf) for j_2 in range(num_shelf) for i in␣
↪range(temp_num_product)))

    second.addConstrs((j-j_2>=-(quicksum(z[i,shelf] for shelf in␣
↪range(num_shelf))-1+huge_number*(2-z[i,j]-z[i,j_2])) \
        for j in range(num_shelf) for j_2 in range(num_shelf) for i in␣
↪range(temp_num_product)))

    second.setParam('TimeLimit', hardlimit)
    second.optimize(softtime)

    second_res=pd.DataFrame({"product_id":product_within_cabinet})
    for i in range(num_shelf):
        second_res["shelf_{}".format(i)]=0

    for v in x:
        if x[v].x > 0:
            second_res.loc[second_res.
↪product_id==product_within_cabinet[v[0]],"shelf_{}".format(v[1])]=x[v].x
    second_res=second_res.merge(revenue_info,on="product_id",how="left")
    return second_res
```

[ ]:
```
softlimit = 5
hardlimit = 60*60*4

def softtime(model, where):
    if where == GRB.Callback.MIP:
        runtime = model.cbGet(GRB.Callback.RUNTIME)
        objbst = model.cbGet(GRB.Callback.MIP_OBJBST)
        objbnd = model.cbGet(GRB.Callback.MIP_OBJBND)
        gap = abs((objbst - objbnd) / objbst)

        if runtime > softlimit and gap <= 0.5/100:
            model.terminate()

def save_model(m):
```

```python
    gv = m.getVars()
    names = m.getAttr('VarName', gv)
    for i in range(m.SolCount):
        m.params.SolutionNumber = i
        xn = m.getAttr('Xn', gv)
        lines = ["{} {}".format(v1, v2) for v1, v2 in zip(names, xn)]
        with open('{}_{}.sol'.format(m.ModelName, i), 'w') as f:
            f.write("# Solution for model {}\n".format(m.modelName))
            f.write("# Objective value = {}\n".format(m.PoolObjVal))
            f.write("\n".join(lines))
```

[ ]:
```python
for i in range(7):
  print("-"*150)
  print("cabinet_{}".format(i))
  second_res=second_stage("cabinet_{}".format(i)).astype("int")
  second_res.to_excel("./result_imputed_data/cabinet_{}.xlsx".format(i))
  print(second_res)
```

[ ]:
```python

```