A Comprehensive Study of Assortment Optimization with Substitution and Uncertainty: Introducing a Machine Learning Heuristic

Harrasse Abir

A Comprehensive Study of Assortment Optimization with Substitution and Uncertainty: Introducing a Machine Learning Heuristic

Abir HARRASSE

June 2023

Contents

1 Introduction

1.1 Glossary

The following glossary, inspired by [11] and [4], provides definitions and explanations of key terms commonly used in the retail industry, offering a comprehensive understanding of concepts related to e-commerce, omnichannel strategies, inventory management, customer experience, and more.

E-commerce: The buying and selling of goods or services over the internet.

Omnichannel: A strategy that integrates various channels (e.g., online, mobile, physical stores) to provide a seamless and consistent customer experience.

Inventory Management: The process of efficiently tracking, organizing, and controlling inventory levels to meet customer demand while minimizing costs.

Point of Sale (POS): The location or system where a customer completes a transaction, typically referring to the checkout counter or the software used for sales transactions.

Merchandising: The activities and strategies involved in promoting and selling products to customers, including product selection, pricing, promotion, and presentation.

Supply Chain: The network of organizations, activities, resources, and processes involved in the production, distribution, and delivery of goods or services to customers.

Customer Experience: The overall interaction and impression a customer has with a retailer or brand throughout their buying journey, encompassing all touchpoints and interactions.

Retail Analytics: The practice of collecting, analyzing, and interpreting data to gain insights and make data-driven decisions in various areas of retail, such as sales, customer behavior, and inventory management.

Footfall: The number of people who enter or visit a retail store or location within a specific time period.

Conversion Rate: The percentage of visitors or potential customers who complete a desired action, such as making a purchase or signing up for a newsletter.

Upselling: The practice of encouraging customers to purchase additional or higher-priced items or services to increase the overall transaction value.

Cross-selling: The strategy of recommending or selling complementary or related products to customers based on their current purchase or interests.

Loyalty Program: A marketing initiative that rewards customers for their repeat business or loyalty with various incentives, discounts, or exclusive offers.

Out-of-Stock: When a desired product is not available for immediate purchase due to inventory depletion.

Markdown: A reduction in the original selling price of a product to stimulate sales, typically used for clearance or promotional purposes.

Return on Investment (ROI): A measure of the profitability and effectiveness of an investment, indicating the return or profit generated relative to the investment cost.

Customer Relationship Management (CRM): The practices, strategies, and technologies used to manage and nurture customer relationships, often involving the use of customer data and insights.

Private Label: Products that are manufactured or branded by a retailer and sold under their own brand name, rather than a third-party brand.

Sales Forecasting: The process of predicting or estimating future sales based on historical data, market trends, and other relevant factors.

SKU: Stock-keeping unit. This is the basic term for a piece of merchandise.

Price type: Regular, markdown, event, rain check, BOGO (buy one, get one free), or clearance.

Pack quantities:: Package quantities designate how many items will be packed in a single bundle.

Gross margin: The difference between cost and selling price (revenue minus cost of goods sold).

Case pack: Products are shipped in full cases (for example, 12, 24, or 36 units). These types of products cannot be broken down into smaller quantities.

1.2 Analytics in retail

Analytics has emerged as a powerful tool in various industries, and the retail sector is no exception. In fact and according to the June 2023 report of McKinsey Company[7], Retail industry is the 3rd top industry to benefit financially (240-390 $ billion) from generative AI, a new candidate in the analytics landscape. With the rapid growth of data and advancements in technology, analytics has become increasingly essential for retailers to understand consumer behavior, optimize operations, and foster business expansion. By leveraging data-driven insights, retailers can make informed decisions, improve customer experiences, and stay ahead of the competition.

It encompasses a wide range of applications, leveraging techniques from data analysis, mathematical programming, statistical modeling, machine learning, and artificial intelligence. In the following, we will explore how analytics is used in the retail industry.

- **Customer Analysis:** By analyzing customer data, retailers can understand customer preferences, segment their customer base, and personalize marketing strategies and product recommendations to improve customer satisfaction and drive sales.
- **Supply Chain Management:** Analytics assists retailers in optimizing supply chain operations by forecasting demand, optimizing inventory levels, and reducing stockouts, leading to improved efficiency and cost savings.
- **Market Basket Analysis:** Market basket analysis, a technique in analytics, helps retailers identify relationships between products frequently purchased together. This analysis allows retailers to implement cross-selling and upselling strategies, optimize product placement, and improve recommendations, ultimately increasing sales and enhancing the customer shopping experience.
- **Pricing and Promotions:** Retailers can use analytics to determine optimal pricing strategies based on factors such as competition and customer willingness to pay. Analytics also helps identify effective promotional activities and allocate resources efficiently to maximize return on investment.
- **Merchandising:** Effective merchandising strategies, supported by analytics, enable retailers to make informed decisions on product selection, pricing, and inventory management. By understanding customer preferences, market trends, and competitor analysis, retailers can optimize *product assortment*, placement, and product assortmentpresentation to drive sales and enhance customer experience.
- **Store placement:** By analyzing data on demographics, foot traffic, competitors, and market trends, retailers can strategically locate their stores for maximum visibility and accessibility. This data-driven approach helps retailers attract more customers, increase sales, and identify new market opportunities for expansion.
- **Market Trends and Competitive Analysis:** Analytics enables retailers to analyze market trends, monitor competitor performance, and identify opportunities for differentiation and competitive advantage, guiding strategic decision-making.
- **Inventory Management:** Through analyzing historical sales data and demand patterns, retailers can optimize inventory levels, reduce excess stock, minimize stockouts, and improve cash flow management.
- **Fraud Detection and Loss Prevention:** Analytics helps retailers detect patterns of fraudulent activities, identify potential risks, and implement preventive measures to minimize losses due to theft, fraud, or security breaches.

1.3 What is assortment optimization?

Assortment optimization is a strategic process employed by retailers and businesses to maximize their product offerings and meet the diverse needs of their customers. It involves analyzing vast amounts of data, such as sales history, market trends, and customer preferences, to determine the ideal assortment of products to offer. By understanding which

items perform well and appeal to the target market, businesses can make informed decisions about which products to stock, which to discontinue, and how to allocate limited shelf space effectively. Assortment optimization aims to strike a balance between variety and efficiency, ensuring that customers find the products they desire while minimizing costs and maximizing profitability for the retailer. This data-driven approach empowers businesses to tailor their assortments to specific customer segments, enhancing customer satisfaction, driving sales, and ultimately fostering long-term success.

1.4 Assortment optimization for retailers: rationale and implementation

The assortment optimization problem is gaining importance due to several reasons, including:

- An increasing number of SKUs.
- Limited shelf space.
- Growing supply chain complexity.
- Location-dependent dynamics of stores.
- An increasing number of clients.
- With a wide range of products and slim gross margins, precise optimization is essential to remain profitable

According to Target's study case, for each project aiming to implement assortment optimization methods, several steps are essential. The proof of concept stage is crucial to gain the retailer's confidence and develop methods tailored to the specific case. Following this, the insights from the proof of concept are utilized, developed further, and integrated into the internal functioning of the retailer, ensuring that analytical methods are effectively incorporated.

To maximize the added value of the developed models, collaboration with the client's business and marketing teams is essential, as it helps uncover the internal structure of the company, its aspirations, constraints, operational rules, and more. Finally, having access to data alone is not sufficient; the ability to manipulate and expand the data's scope by leveraging historical data is critical. This enables understanding of various trends and facilitates the proposal of highly customized models for the client (Marjane).

The significance of optimization methods applied to assortment stems from the continuous growth in data size (the number of SKUs). The ability to process and store this ever-increasing volume of data has become challenging. Thus, the importance of incorporating optimization.

1.5 Assortment Optimization: State of the Art

Research in assortment optimization, also known as "the assortment optimization problem," has seen numerous publications and advancements. This problem has been studied under different frameworks, and both exact and heuristic algorithms have been developed in this context. In this section, we will provide a brief overview of the literature on this topic.

In the last 60 years, extensive research has been conducted on the assortment problem, particularly in relation to sectors such as apparel [8] or grocery. This research can be grouped into different categories based on the specific problem framework and its definition, including:

- **Demand: deterministic vs. stochastic**.
- **Demand pattern: discrete vs. continuous:** The demand pattern in assortment optimization can be either discrete, characterized by distinct quantities or counts of demand at different time periods, or continuous, representing a smooth distribution of demand over time.
- **Dimensions: single vs. multiple:** The dimension in assortment optimization refers to the number of attributes used to describe products, with single dimension focusing on one attribute (e.g., size) and multiple dimensions considering multiple attributes (e.g., size, color, brand).
- **Number of sizes to stock: fixed vs. to be determined.**
- **Cost substitution structure: linear vs. non-linear**: The cost structure in assortment optimization can be either linear (proportional relationship between cost and quantity) or non-linear (complex or nonlinear cost patterns).
- **Inventory's model stability: stationary vs. non-stationary:** Inventory model stability in assortment optimization refers to whether the inventory levels exhibit consistent patterns (stationary) or fluctuate (non-stationary) over time.

Unfortunately, in the case of Marjane (the retailer studied), these types of results will be of limited use to us since they mainly focus on the case of size assortments in apparel, where the diversity and complexity are far from that of our case in which the depth/breadth level of each article category is very large.

Fortunately, research has also focused on our specific case, and we can find publications and studies within various frameworks:

- **Omni-channel structure**: Retailers are experiencing a trend where sales are conducted both in physical stores and online [2][6]. In the following sections, we will explore how this trend is well-justified and examine different works done in this direction to develop algorithms that solve these cases.
- **Time-based assortment optimization**: In this research framework, the focus is on the real-world scenario where the assortment can only be changed after specific time intervals (which can be interpreted as waiting for the shelves to empty before refilling them) [5] [6].

- **Consideration of synergy and cannibalization effects**: In this model, the strong effects between products in the same assortment are taken into consideration [6].
- **Incorporating uncertainty**: Certainty in the study of assortment problems is a significant simplifying assumption, hence the necessity of introducing the uncertainty factor into a model that aims to approximate reality [9].

In the following, we will begin with a simple formulation and its implementation. Then, the model will be developed based on the latest research mentioned above, and a dataset will be implemented to demonstrate the effectiveness of the algorithms.

We will work under the assumption that the overall optimal assortment is the same as the union of assortments made under each product category (e.g., Dairy products, Books...). Therefore, the following formulation only considers a single category, and the rest is simple: replicate the corresponding algorithm for different categories and combine the resulting assortments. **The final result is the assortment to adopt**.

2 Linear formulation: implementation and development avenues

After the discussion with Marjane, we concluded on the parameters to include in our model:

- Linear footage (1)
- Customer type (2)
- Minimum number of facings per product (3)
- List of essential items (4)

In the following, we will gradually build our model while introducing the different required parameters.

2.1 An initial formulation

For this formulation, we will only consider parameters (1) and (3). Hence, for each product, a minimum facing is provided. **We suppose that we only display the minimum facing for each product** for the sake of simplification.

Thus, our formulation is as follows:

$$\begin{aligned} \text{Max:} \quad & \sum_{k=1}^{N} p_k x_k \\ \text{S.t:} \quad & \sum_{k=1}^{N} w_k b_k x_k \leq C \end{aligned} \tag{1}$$

Where:

. p_k: is the gross profit of the product k.

. b_k: the minimum facing of the product k (The quantity that will be displayed).
. w_k: the volume per product i.
. C: The shelf constraint.
. x_k: A binary variable (equals 1 if the product is in the assortment, 0 otherwise)

We observe that this initial formulation is similar to the knapsack problem. Therefore, this problem is NP-hard, and obtaining an exact solution using an exact algorithm is not feasible for large products' sets. The use of a heuristic must be necessary.

This will be the focus of the next section!

2.2 Data: dairy products

First, we will start by generating fake data for dairy products in a store:

We generate 170 dairy products, each with its corresponding profit margin, volume per item, and the minimum number of items to include. Below are the first few lines of this data:

Products	**Gross proft (MAD)**	**Volume (ml)**	**Minimum facing**
XYZ Dairy Milk	2.8	1000	60
ABC Farms Yogurt	1.95	150	55
Dairy Delights Cheese	4.2	250	70
Golden Spread Butter	3.1	500	75
Sweet Treats Ice Cream	6.8	1000	80

Table 1: Dairy data

Therefore, we utilize the generated data to implement the aforementioned formulation in Python. We use the PuLP package to obtain a list of products to include in the assortment. This list maximizes the retailer's profit while adhering to the space constraint.

2.3 Exact solution: python implementation

As mentioned before, our problem is exponential in the size of the inputs. Therefore, we can expect the runtime, which did not exceed one second for our dataset of 170 products, to explode as we increase the number of products we choose from. To verify our hypothesis, we plot a curve representing the runtime against the input size. The result is indeed exponential, as we previously stated.

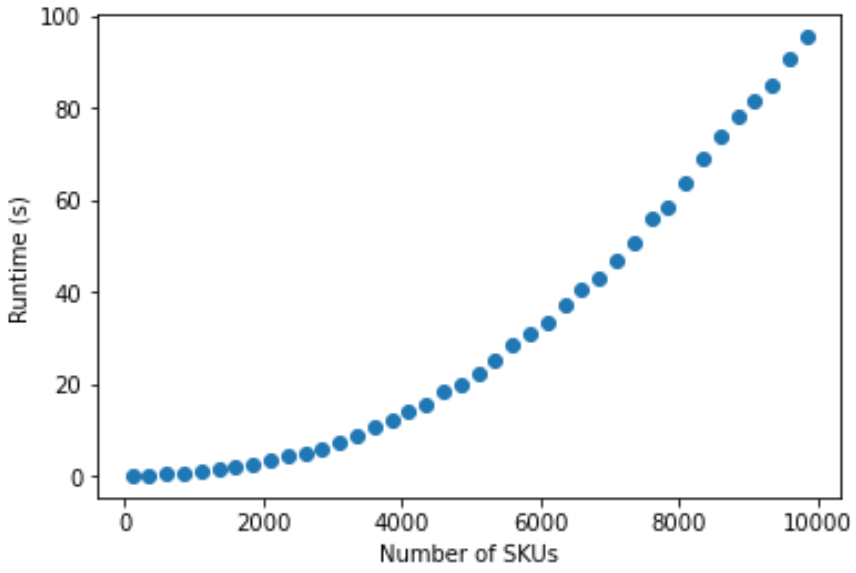


Figure 1: Runtime as a function of inputs size

We notice that the assortment for 10,000 products can be done within a reasonable time (approximately 1 minute). However, the runtime explodes to 23 days for 20,000 products!

The need for a heuristic and a powerful computer is evident. As for the computing, we decided to use a more efficient solver such as the *FICO Xpress optimizer*. The results improved as the following curves suggest:

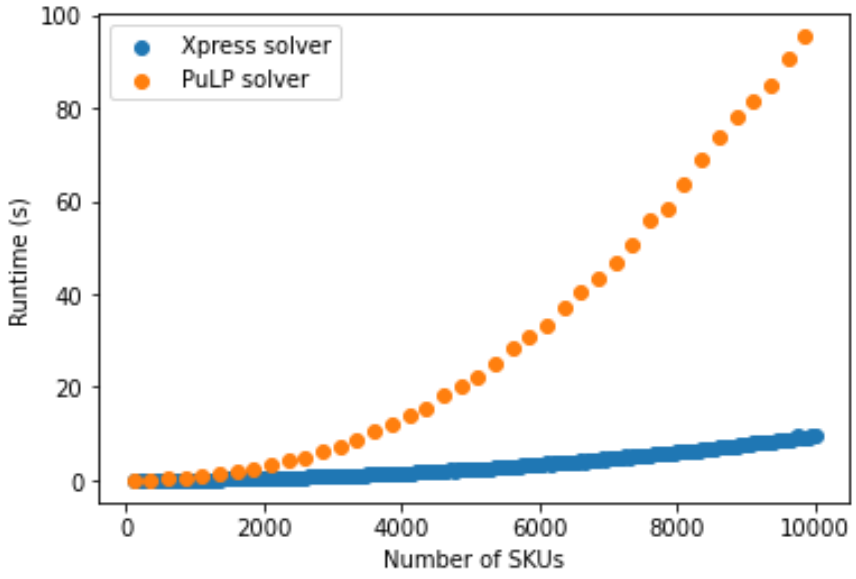


Figure 2: Runtime's comparison between Xpress and PuLP solutions

The runtime of the algorithm using the *FICO Xpress solver*for 20000 products improved to 2days (computed using an extrapolation) compared to 23days in the previous setting!

As for the heuristic approach, there are many choices to pick from, we can work with a metaheuristic such as the genetic algorithm, which is well-suited for the Knapsack Problem.

In this sense, the literature offers implementations that are well-suited for large inputs [10].

Another option is to implement a heuristic which is more "tailored" to the knapsack problem. In the following, we introduce a heuristic and we compare it to the previous exact resolution using *Xpress solver*.

2.4 The knapsack problem's relaxation: introduction and implementation

We solve the knapsack problem under its relaxed form, where the decision variables x_i are chosen in the interval [0, 1] rather than binary variables. The problem is then solved as an LP with x_i in the range [0, 1], and the list of x_i values is arranged in decreasing order. We include the products i based on the list of x_i values, while ensuring they satisfy the constraint C representing the weight limitation. By allowing x_i to take fractional values and sorting them in descending order, we prioritize the most valuable items for inclusion in the knapsack. Sequentially considering the products in this order, we add each product to the knapsack as long as the cumulative weight constraint C is not exceeded. This approach strikes a balance between computational efficiency and obtaining a close approximation to the optimal solution, as it optimizes the value of the knapsack while respecting the weight limitation.

Another equivalent approach is to classify the products i by their profit densities $\frac{p_k}{w_k b_k}$ in decreasing order, instead of solving the LP relaxation version. This alternative method offers the advantage of reducing both the runtime and complexity of the algorithm. By prioritizing items with higher profit densities, calculated as the ratio of profit p_k to weight $w_k b_k$ of the product, we can streamline the selection process. Sorting the products based on profit densities ensures that more valuable items are considered first. By focusing on this criterion, we can optimize the algorithm's efficiency, leading to improved performance and faster computation times as the following curve suggests:

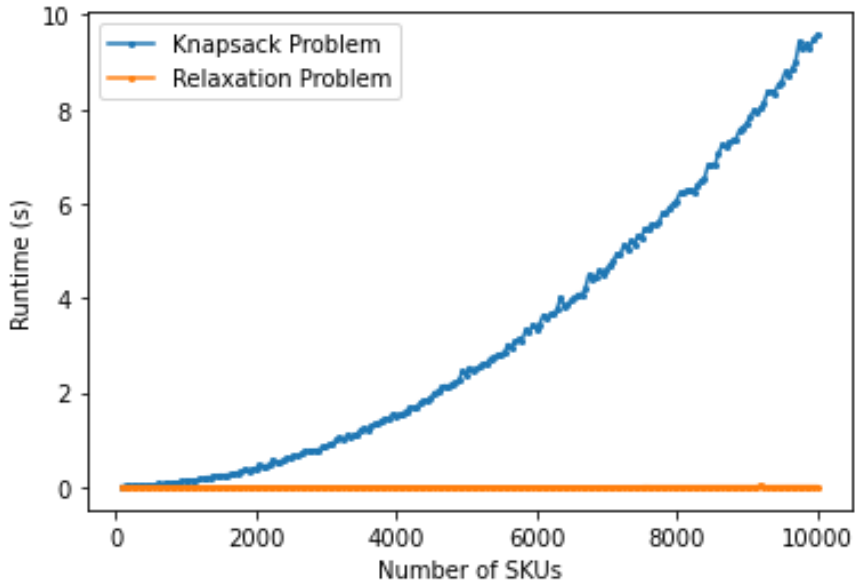


Figure 3: Runtime's comparison between the LP algorithm and the knapsack heuristic

Furthermore, employing the knapsack heuristic algorithm to solve the assortment problem yields a solution that is remarkably close to the actual value of the objective function. In fact, the deviation or error incurred by this relaxed approach is minimal, typically less than 1%. This indicates that the knapsack heuristic algorithm provides an accurate approximation, allowing us to efficiently and effectively optimize the assortment while maintaining a high level of fidelity to the optimal solution.

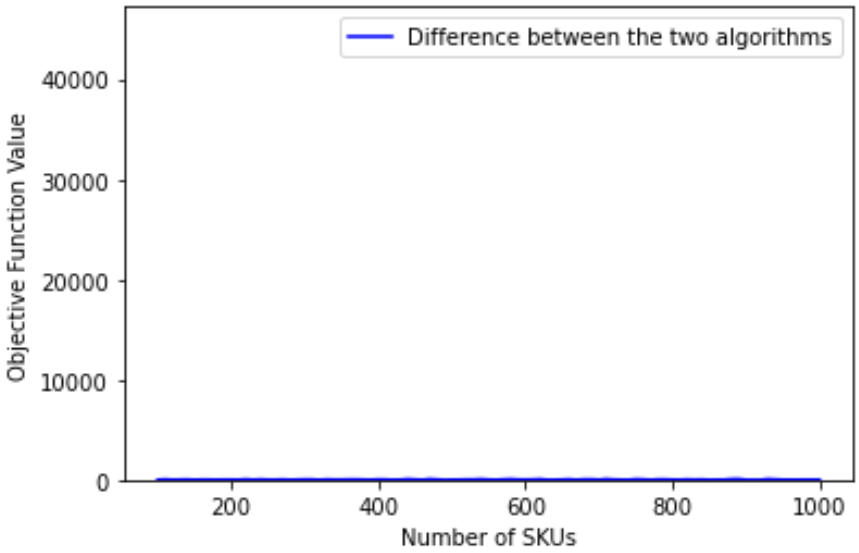


Figure 4: Error of the knapsack heuristic algorithm computing the objective function compared to the objective function's value

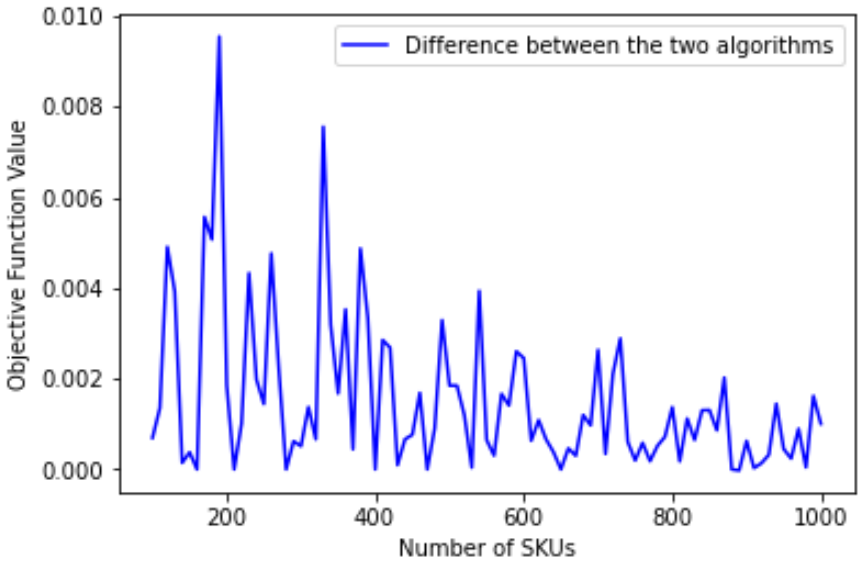


Figure 5: Error of the knapsack heuristic algorithm computing the objective function

2.5 Conclusion

So far, we have examined our problem from a very simplified perspective, considering only the space constraint. However, this framework involves many simplifying assumptions:

- The demand is not taken into account.
- The stochastic nature of the problem, mainly arising from uncertain demand, is omitted.
- The effects of substitution are not considered.

These simplifying assumptions overlook several important aspects of the problem. It would be wise to further develop our formulation by addressing these points! We can even consider working on more realistic paradigms by considering:

- The risk of stock-out and the fact that the retailer only has access to assortment change after a period T, where the assortment offered to customer n depends on both the choice of customer $n-1$ and the initial assortment offered by the retailer.
- The online and in-store activities of the retailer to consider both entities under a unified view and optimize the overall business operations.

3 Quadratic formulation and resolution

3.1 Assumptions

In our work, as mentioned in the previous paragraph, we will consider both the demand and its uncertain nature, as well as the effect of substitution.

Regarding **substitution**, it is well-known in the marketing literature [3] that substitution generally occurs in two forms: assortment-based and stock-out based:

- Assortment-based substitution: occurs when a customer substitutes their preferred product x with another product y when x is not available in the store's assortment.
- Stock-out based substitution: occurs when a customer substitutes their preferred product x with another product y due to the unavailability of x (stock-out).

It has been demonstrated by Campo et al. (2004) [3] that both types of substitution exhibit significant similarities. Initially, we can consider only one type of substitution, such as assortment-based substitution. Therefore, we present our working assumptions as follows:

- We work within a period T during which the retailer does not have access to their assortment. This allows us to consider substitution due to stock-outs.
- The number of customers K during period T is a random variable. Two approaches can be considered: either predict the number of customers using available data for a given period T, or consider K as a random variable and determine its distribution based on the data.
- The price of each product is fixed.
- Each customer makes only one substitution: if they decide to substitute their preferred product x with product y, and if y is also unavailable, the customer decides not to make a purchase.
- A customer decides to make a substitution with a probability δ, otherwise, they leave the store without making a purchase with a probability $1 - \delta$.

  In fact, it has been demonstrated [1] that multiple substitutions with a probability δ' are equivalent to a single substitution with a probability δ'' where $\delta'' > \delta'$ (if δ' is not too large).

3.2 Formulation

In this formulation, we will take into consideration only the **assortment based** substitution, the **stock-out** based substitution will be tackeled under uncertainty later on.
The following is the formulation we suggest:

$$\max_{\mathbf{S}\subseteq\mathbf{N}} \sum_{i\in N} K_T p_i d_i x_i + \sum_{i\in N} K_T p_i \sum_{j\neq i} \delta(\alpha_{ji} d_j (1 - x_j) x_i) - \sum_{i\in N} c_i u_i \tag{2}$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{i \in N} w_i u_i \leq C \\
& u_i \leq M x_i && \forall i \in N \\
& u_i \geq K_T d_i x_i + K_T \sum_{j \neq i} \delta \alpha_{ji} d_j (1 - x_j) x_i) && \forall i \in N \\
& x \text{ in } \{0,1\}^n,\ u \text{ in } \mathbb{N}^n
\end{aligned}
$$

Our objective function comprises two components: the original demand represented by the first part of the sum and the assortment-based substitution by the second part.

Below, we define the parameters of our model, where the index T represents the specific day or period being considered. We have:

Decision variables:

. x_i: Binary decision variable that takes a value of 1 if product i is included in the assortment and 0 otherwise.

. u_i: Integer decision variable indicating the quantity of product i stocked over the period T.

Model parameters:

. K_T: The number of customers who visit the store during the period T

. p_i: The unit price of product i

. d_i: The demand for product i.

. δ: The probability of considering substituting a preferred product.

. αij: The assortment substitution rate from i to j.

. c_i: The storage cost per unit for product i.

. w_i: Volume per product i.

. M: Big-M constraint.

Constraints:

The first constraint expresses the space limitation in the section reserved for this category. The second constraint uses the Big-M constraint technique, which states that the quantity of product i chosen will be **zero** if product i is not included in the assortment ($x_i = 0$).

To determine the optimal assortment, we will work incrementally. It should be noted that demand and the number of customers arriving at the store are random variables. We will start with the simple case where these variables are treated as deterministic (which is not far from reality as the number of customers and demand can be predicted using machine learning models). We will then study the more general stochastic case under both assortment and stock-out based substitution.

3.3 Assortment-based substitution: a thorough study

Our problem:

$$
\max_{\mathbf{S} \subseteq \mathbf{N}} \sum_{i \in N} K_T p_i d_i x_i + \sum_{i \in N} K_T p_i \sum_{j \neq i} \delta (\alpha_{ji} d_j (1 - x_j) x_i) - \sum_{i \in N} c_i u_i \tag{3}
$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{i \in N} w_i u_i \leq C \\
& u_i \leq M x_i && \forall i \in N \\
& u_i \geq K_T d_i x_i + K_T \sum_{j \neq i} \delta \alpha_{ji} d_j (1 - x_j) x_i) && \forall i \in N \\
& x \text{ in } \{0,1\}^n, \; u \text{ in } \mathbb{N}^n
\end{aligned}
$$

is a quadratic problem. This adds a layer of complexity to it. However, it will be interesting to study the convexity of the problem. If the problem is convex, the resolution becomes simplified!

3.3.1 Convexity

To study the convexity of the problem, it is sufficient to calculate the Hessian and check its positive definiteness. We define our objective function:

$$f(x) = \sum_{i \in N} K_T p_i d_i x_i + \sum_{i \in N} K_T p_i \sum_{j \neq i} \delta (\alpha_{ji} d_j (1 - x_j) x_i) - \sum_{i \in N} c_i u_i$$

We denote by:

$$\pi_i = K_T p_i d_i$$

A is the matrix whose coefficients are:

$$
\begin{gathered}
a_{ji} = K_T p_i \delta d_j \alpha_{ji} \; \forall i \in N \text{ si } i \neq j \\
a_{ii} = 0 \; \forall i \in N
\end{gathered}
$$

Thus, our objective function becomes::

$$f(x) = \pi \cdot \mathbf{x} + x^T \cdot A \cdot (e - x) - \mathbf{c} \cdot \mathbf{u}$$

Therefore, we define a as a new variables vector such that:

$$\mathbf{a} = \begin{bmatrix} x \\ u \end{bmatrix}$$

and:

$$\mathbf{d} = \begin{bmatrix} \pi \\ -c \end{bmatrix}$$

We also transform A, which was originally a $n \times n$ matrix to a $2n \times 2n$ matrix. We denote it as A. The first block of A will be equal to the original matrix A, and all other entries will be **zeros**. The objective function then becomes:

$$f(x) = \mathbf{d} \cdot \mathbf{a} + a^T \cdot A \cdot (e - a)$$

To study the convexity of the function, we first calculate the gradient of the function and then its Hessian. Let's first calculate the differential. We omit the linear term as it will disappear when calculating the Hessian. We have:

$$\begin{aligned} f(x+h) &= (x+h)^T \cdot A \cdot (e-(x+h)) \\ &= f(x) + (h^T \cdot A \cdot e - h^T \cdot A \cdot x - x^T \cdot A \cdot h) - h^T \cdot A \cdot h \end{aligned}$$

Thus, the gradient of f is:

$$\nabla f(x) = A \cdot e - A \cdot x - A^T \cdot x$$

We find the hessian of the objective function:

$$H_f(x) = -(A + A^T)$$

This matrix is not generally positive definite! Therefore, we need to check the Hessian for the objective function of our problem. Since the coefficients of the substitution are not fixed, it would be wise not to rely on the convexity of the problem!

3.3.2 The quadratic problem linearization

To facilitate the resolution of the quadratic problem, a common approach is to linearize the problem. We introduce the decision variable y_{ij} such that:

$$y_{ij} = (1 - x_j)x_i$$

The problem becomes:

$$\max_{\mathbf{S} \subseteq \mathbf{N}} \sum_{i \in N} K_T p_i d_i x_i + \sum_{i \in N} K_T p_i \sum_{j \neq i} \delta(\alpha_{ji} d_j y_{ij}) - \sum_{i \in N} c_i u_i \tag{4}$$

$$\begin{aligned} \text{s.t.} \quad & \sum_{i \in N} w_i u_i \leq C \\ & u_i \leq M x_i && \forall i \in N \\ & u_i \geq K_T d_i x_i + K_T \sum_{j \neq i} \delta \alpha_{ji} d_j y_{ij}) && \forall i \in N \\ & y_{ij} \leq x_i && \forall i, j \in N \\ & y_{ij} \leq (1 - x_j) && \forall i, j \in N \\ & y_{ij} \geq x_i - x_j && \forall i, j \in N \\ & x \text{ in } \{0,1\}^n, \, u \text{ in } \mathbb{N}^n \end{aligned}$$

3.3.3 Data: the model's parameters

Before implementing our model, we need to estimate the parameters of the demand d_i for all products under consideration, the substitution matrix $(\alpha_{ij})_{i,j\in N}$, the probability of considering a substitution δ, and finally, the number of customers: to be fixed in this first part.

With the data collected from Marjane, we can draw inspiration from A. Gürhan Kök et al. [1] to estimate all these parameters:

- **Demand d_i for a product i:** This parameter can be estimated using three other probabilities that can be calculated from the data[1]. We have:

$$d_i = \gamma b_i q_i$$

where: - γ: The incidence of sales, the probability that a customer visiting the store buys an item from the category under study.

$$\gamma = \frac{\text{Sales of the category under study}}{\text{Total store sales}}$$

- b_i: The purchase probability of product i.

$$p_i = \frac{\text{Number of product } i \text{ items sold}}{\text{Total category sales}}$$

- q_i: The average quantity purchased by a customer.

- **Substitution probability δ:** This probability, combined with the substitution probabilities $(\alpha_{ij})_{i,j\in N}$, replaces choice models (e.g., MNL) that we could have used. The advantage here, according to A. Gürhan Kök et al. [1], is to consider differences in substitution preferences between categories that may have the same initial demand. In this case, to maintain the model tractable, the customer makes only one substitution. If the product they want to substitute with is no longer available, they abandon their purchase.

  To estimate this parameter, we adopt the following approach assuming we have access to different stores of the same retailer:

  We group the different stores into **clusters** where stores with similar characteristics (customer demographics, sales volume, geographical location, store size, etc.) are grouped together. To estimate the substitution probability δ in a store h, we locate the cluster C to which this store belongs and identify the store h' in cluster C with the largest assortment in terms of inclusion. We assume it is complete (containing all the products from which we choose the assortment). Therefore, we have $S_h \subseteq S_{h'}$. We can define δ as follows:

$$\delta = \frac{\sum_{j \in S}(D_j - D'_j)}{D'_{NT} - D'_{ST}}$$

We notice that to establish the substitution probability of the category we are working on in store h, we used the demands (how much was purchased) in store h' under the assumption of similarity in terms of characteristics (hence, the same substitution probability a priori). However, we normalize the demands by the number of customers who visited the store: the demands D are per customer.

The formula is very simple. It states that under the assumption of similarity and since assortment S_h in store h is present in store h' with the largest assortment $S_{h'}$, δ is given by:

$$\delta = \frac{\text{Number of substitutions}}{\text{Unfulfilled demand}}$$

- **Number of substitutions:** It is the sum, for each product belonging to assortment S, of the demand in store h minus the demand in store h' for product j. Assuming that store h' contains all the products that consumers need, they will not make substitutions, so the observed demand is the true demand for product j. However, the demand D_j in store h includes the demand that results from substitution.
- **Unfulfilled demand:** This time, we only consider the demand in store h' with the largest assortment. The unfulfilled demand is given by the demand for assortment $S_{h'}$ (in the case where all products are available) minus the demand when only assortment S_h is presented.

- **Substitution matrix:** Here, too, there are two simplified approaches to tackle the problem. Of course, we can compute a substitution matrix that tackles the substitution relationship between every two products distinctly, but we think that the approaches we will represent are good enough for a start.

  - Random substitution: Here, we assume that customers generally have no preferences other than their favorite products. In this case:

$$\alpha_{ij} = \frac{1}{|S|}$$

  - Proportional substitution: In this case, we assume that a customer's preference, on average, is proportional to the demand for the product in the category. Therefore, we have:

$$\alpha_{ij} = \frac{d_j}{\sum_{l \neq i} d_l}$$

Now that we know how to find all the parameters of the model, we can start solving it using a solver (such as Xpress).

We suppose β_{ij} coefficients to be equal to α_{ij}, because as mentionned earlier, there's a very strong similarity between assortment based and stock-out based substitutions.

3.3.4 Exact solution

Our problem is a Mixed Integer Linear Programming Problem (MILP), and for inputs of limited size, we can simply use available solvers (such as Xpress in our case).
We start by implementing it in PuLP(with Xpress as a solver) and using fake data to feed the model.

To simplify the model, we might replace the decision variable y_{ij} that indicates whether product i is substituted by j by the variable z_{ij}, this time defined as follow:

$$z_{ij} = x_i x_j$$

This will reduce the number of decision variables y_{ij} by 2. Thus, our formulation becomes:

$$\max_{\mathbf{S}\subseteq\mathbf{N}} \sum_{i\in N} K_T p_i d_i x_i + \sum_{i\in N} K_T p_i \sum_{j\neq i} \delta(\alpha_{ji} d_j (x_i - z_{ij})) - \sum_{i\in N} c_i u_i \tag{5}$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{i\in N} w_i u_i \leq C \\
& u_i \leq M x_i && \forall i \in N \\
& u_i \geq K_T d_i x_i + K_T \sum_{j\neq i} \delta \alpha_{ji} d_j (x_i - z_{ij}) && \forall i \in N \\
& z_{ij} \leq x_i && \forall i, j \in N \\
& z_{ij} \leq x_j && \forall i, j \in N \\
& z_{ij} \geq x_i + x_j - 1 && \forall i, j \in N \\
& x \text{ in } \{0,1\}^n, z \text{ in } \{0,1\}^{n^2},\ u \text{ in } \mathbb{N}^n
\end{aligned}$$

By implementing the model in Xpress, and trying out different inputs' sizes, we end up with the following curve:

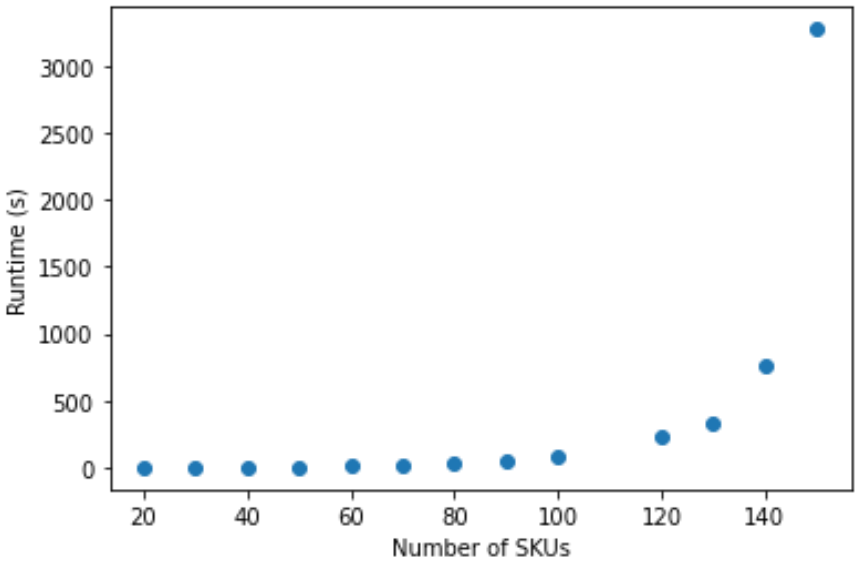


Figure 6: Runtime as a function of inputs size

The process of finding the optimal assortment from a set of 150 SKUs requires approximately **one hour**!, highlighting the significant time investment involved. This underscores the necessity for a heuristic approach to address this challenge efficiently. Consequently, our upcoming section will focus on exploring and implementing a heuristic method to solve this problem effectively.

3.3.5 Our heuristic: an ML based algorithm

In this section, we will introduce a heuristic approach to address our problem under the assortment-based substitution. The primary objective is to attain solutions that are close to optimal while ensuring that the computational time remains low. By leveraging this heuristic method, we aim to strike a balance between solution quality and efficiency in our problem-solving process.

Our approach draws inspiration from the knapsack heuristic algorithm we discussed earlier. The key concept is to assign a score to each product, reflecting its performance. These products are then ranked in descending order based on their scores. We include products with higher scores in the assortment, with quantities matching their respective demand, until the shelf space constraint is exceeded. When that happens, we remove the last product included in the assortment and replace it with the remaining quantity from the product with the highest performance score. This iterative process helps us optimize the assortment while considering both product performance and shelf space limitations.

The crux of the issue lies in determining a score for each product. Attempting to achieve this in a greedy manner, where we compare each product with all others and explore every conceivable combination to select an assortment, proves to be an infeasible task. The sheer number of combinations makes this approach impractical and computationally prohibitive.

To address this challenge, we devised a strategy to estimate the scores for a limited set of products. Specifically, we focused on studying 1000 products out of a total of 10000. With

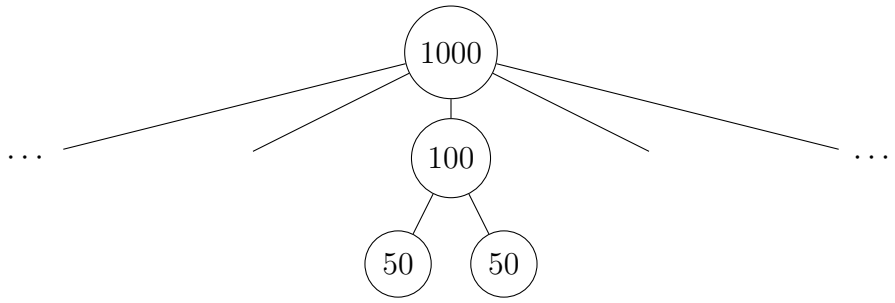
the scores obtained from this subset and considering the relevant features of each product (such as demand, inventory cost, and the substitution matrix $(\alpha_{ij})_{i,j\in N}$, we built a machine learning (ML) model. This model is designed to predict the score of a product i based on its specific features. By adopting this approach, we aim to significantly reduce the expected runtime while still obtaining reasonably accurate score estimates for the products in our assortment.

Our work will be structured into four distinct sections. Firstly, we will present our approach for determining the scores of a limited set of products, specifically 1000 items. After this, we will present the implementation of our method. Next, we will proceed with training a machine learning (ML) model using this set to establish the scores for each individual product. Finally, armed with the derived scores, we will proceed to construct our assortment.

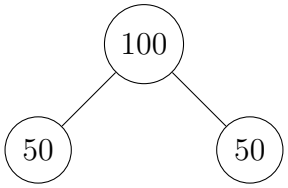
3.3.5.1 The scoring algorithm

Our reasoning will revolve around evaluating the scores for a set of 1000 products. To achieve this, we will divide the initial set of 1000 products into multiple subsets, following the rules outlined below:

1. Initially, we partition the initial set into ten subsets, each containing 100 elements.
2. Afterward, each subset of 100 elements will be further divided into two sets, each consisting of 50 elements.



Presently, our rationale revolves around every node comprising 100 elements:



Let's denote by:

S_{100}: every set of 100 elements
S_{50}: every set of 50 elements

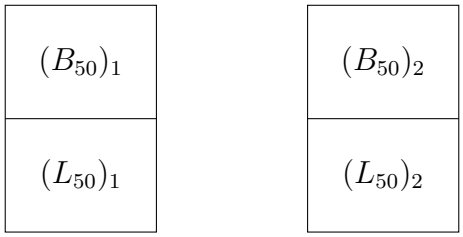
We adhere to the following guidelines for evaluating the elements in both the S_{50} and S_{100} type subsets:

1. For each S_{50} subset, we solve an optimization problem in its deterministic form to select a set of products. The solution of this problem, denoted as (x, u), allows us to assign a score to each product, referred to as the initial score s_d.

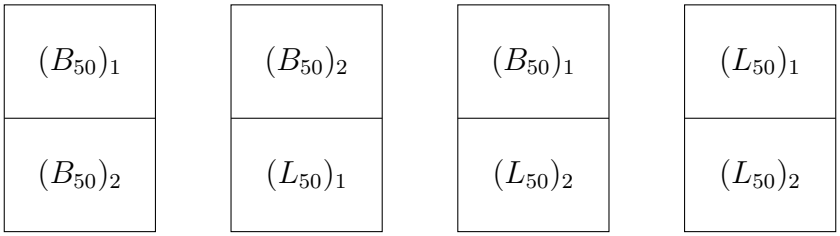
$$(s_d)_i = \frac{u_i}{\max(u_i, i \in N)}$$

2. After scoring the two S_{50} subsets, we proceed to establish a global score by comparing the elements of $(S_{50})_1$ with those of $(S_{50})_2$. This involves the following steps:

2.1. We set a threshold of 0.5 to partition each S_{50} subset into two parts: the most high-performing elements B_{50} and the less high-performing elements L_{50}.



2.2. We create new subsets T_{50} by forming various combinations from the initial sets $(S_{50})_1$ and $(S_{50})_2$. The resulting sets are defined as follows:



The concept involves comparing the top-performing elements within each subset S_{50} against each other, comparing the lower-performing elements against each other, and finally, comparing the top-performing elements of one subset S_{50} against the lower-performing elements of the other subset. This approach allows us to generate a score that considers the relationship between each pair of products comprehensively.

2.3. Afterward, a score is computed for each product i as a weighted average of the scores obtained from every set T_{50} along with the initial sets S_{50}. When comparing a product i with a high-performing set, a weight of 2 is assigned, while a weight of 1 is given for comparisons with low-performing sets. let's denote by s_d, the score we get in this step:

$$(s_m)_i = \frac{\sum_{i=1}^{3} w_i score_i}{\sum_{i=1}^{3} w_i} \tag{6}$$

Where:
-i: denotes the specific T_{50} or S_{50} set studied.
-w_i: denotes a score of 2 or 1 depending on the set of elements, i was put with.
-$score_i$: denotes the score that product i gets in the ith set of type T_{50} or S_{50} it belongs to.

It should be noted that the sum in (6) includes the initial score $s_{initial}$ introduced earlier!

With the computed scores s_m, we now have rankings for each of the ten sets S_{100}. Our next step is to partition each S_{100} set into two groups: the most high-performing products and the less high-performing products, as we did previously. Afterward, we will execute a similar **combination routine** on these ten sets, ensuring that we do not exceed a total size of 100 products in the deterministic optimization model.

Our objective is to ensure that each product i in either category L_{100} or B_{100} is compared to all other products from different categories in the other sets S_{100}. However, we also aim to minimize the number of combinations while obtaining the best possible results.
By carefully selecting the appropriate combinations and the best formulation of scores, we can efficiently evaluate the relationships between products across all sets and arrive at an optimal solution.

As before, we initially divide each S_{100} set into two parts using the same threshold as previously set: $p = 0.5$. We call these sets instance I_1

$(B_{100})_1$	$(B_{100})_2$	$(B_{100})_3$	$\cdots$	$(B_{100})_{10}$
$(L_{100})_1$	$(L_{100})_2$	$(L_{100})_3$	$\cdots$	$(L_{100})_{10}$

The subsequent step involves creating **four** instances: I_2, I_3, I_4, I_5 of sets following a 10-cycle that look like the following:

$(B_{100})_1$	$(B_{100})_2$	$(B_{100})_3$	$\cdots$	$(B_{100})_{10}$
$(B_{100})_2$	$(B_{100})_3$	$(B_{100})_4$	$\cdots$	$(B_{100})_1$
$(L_{100})_1$	$(L_{100})_2$	$(L_{100})_3$	$\cdots$	$(L_{100})_{10}$
$(L_{100})_2$	$(L_{100})_3$	$(L_{100})_4$	$\cdots$	$(L_{100})_1$
$(B_{100})_1$	$(B_{100})_2$	$(B_{100})_3$	$\cdots$	$(B_{100})_{10}$
$(L_{100})_2$	$(L_{100})_3$	$(L_{100})_4$	$\cdots$	$(L_{100})_1$

$(L_{100})_1$	$(L_{100})_2$	$(L_{100})_3$	$\cdots$	$(L_{100})_{10}$
$(B_{100})_2$	$(B_{100})_3$	$(B_{100})_4$	$\cdots$	$(B_{100})_1$

The goal is to compare each product to a maximum number of other products while adhering to the constraint of working with the 40 generated sets mentioned earlier. However, we notice that each L/B set is only compared to 3 other sets instead of 18 in the instances we suggested.
To address this issue, we leverage the 10-cycle structure inherent in the sets. For instance, the first set in I_1 generates a comparison between $(B_{100})_1$ and $(B_{100})_2$, and the second set generates a comparison between $(B_{100})_2$ and $(B_{100})_3$, resulting in a comparison between $(B_{100})_1$ and $(B_{100})_3$. Continuing this pattern, we find that $(B_{100})_1$ is compared to all the other sets $(B_{100})_i$, and similarly, all the B/L sets are compared to each other.

Our task is to devise a score that considers these relationships between different sets, even if they are not directly compared.
We denote by:

.$(sc_k)_i$: the average score of product i in the instance I_k. It's computed by averaging over the scores of product i obtained in the sets of instance I_k that include product i (namely 2 sets per product).
.$S(I_K)_l$: the lth set in the instance I_k.

To accomplish this, we introduce the following formula:

$$(s_f)_{ik} = \max\left((sc_k)_i, \left[(sc_k)_j + \frac{1}{1000} \mid \text{if } (sc_k)_j < (sc_k)_i \text{ and } (i,j) \in S(I_k)_l \, \forall l \in [1,10] \text{ where } k \in [1,4]\right]\right)$$

This formula ensures a connection between the various products and assigns an instance's score to product i that surpasses the instance's score of product j if i is found to perform better than j in any comparison conducted.

The overall score of product i is computed as follows:

$$(s_f)_i = \frac{\sum_{m \,\in\, ind_i} w_m.(s_f)_{i,m}}{\sum_{m \,\in\, ind_i} w_m}$$

Where:
.ind_i: is the set of the indices of instances where product i appeared (there's 4 instances for each product)

The final scores are given by the vector: s_f.

The algorithm is the following:

Algorithm 1 The Scoring Algorithm

Input: A list of 10000 items

Output: A list of overall scores for 1000 items

Function `Main`**:**

Step 1: Draw 1000 items out of 10000 randomly

Step 2: Divide the 1000 items into 10 sets S_{100_i} of 100 items each

for *each* i **do**

Step 3: Divide the set S_{100_i} into 2 subsets $S_{50_{i1}}$ and $S_{50_{i2}}$ of 50 elements each

Step 4: Run the optimization model on each set $S_{50_{i1}}$ or $S_{50_{i2}}$

Step 5: Establish a score for each element in these sets using the formula

$$(s_d)_i = \frac{u_i}{\max(u_i, i \in N)}$$

Step 6: Divide each set $S_{50_{i1}}$ or $S_{50_{i2}}$ into two parts $L_{50_{i1}}$, $B_{50_{i1}}$, $L_{50_{i2}}$, $B_{50_{i2}}$ using a threshold $p = 0.5$

Step 7: Form 4 new sets $T_{50_{i1}}$, $T_{50_{i2}}$, $T_{50_{i3}}$, $T_{50_{i4}}$ by merging $(L_{50_{i1}}, L_{50_{i2}})$, $(L_{50_{i1}}, B_{50_{i2}})$, $(B_{50_{i1}}, L_{50_{i2}})$, $(B_{50_{i1}}, B_{50_{i2}})$

for *each set* T_{50_i} **do**

Step 8: Run the optimization model and compute the scores as done previously

end

Step 9: Compute the total score for each element i in S_{100_i}

$$(s_m)_i = \frac{\sum_{i=1}^{3} w_i score_i}{\sum_{i=1}^{3} w_i}$$

end

Step 10: Generate the 4 instances of elements $I2$, $I3$, $I4$, $I5$ using the 10-cycle

for *each instance* **do**

for *each set in the instance* **do**

Step 11: Compute the average score for each element in each instance using the formula

$$(s_f)_{ik} = \max\left((sc_k)_i, \left[(sc_k)_j + \frac{1}{1000} \mid \text{if } (sc_k)_j < (sc_k)_i \text{ and } (i,j) \in S(I_k)_l\right]\right)$$

Step 13: Use the formula to compute the score inside the instance for each product

$$(s_f)_i = \frac{\sum_{m \in ind_i} w_m.(s_f)_{i,m}}{\sum_{m \in ind_i} w_m}$$

end

end

Step 14: Use the formula to compute the overall scores

return *The list of overall scores*

We utilize it to obtain a list of scores for 1000 elements randomly selected from a pool of 10000 products. This list of scores will be used for labeling the data that will be fed into the subsequent ML model.

3.3.5.2 The ML model

In this section, we perform predictions on a subset of 3000 products, as the process remains the same for 10000 products.

First, we preprocess the data by dropping unnecessary columns, such as product indices. Additionally, we break down the substitution vectors $(\alpha_{ji})_{i,i\in N}$ into $card(N)$ columns, where each column represents the substitution probability of product j by product i. With a total of 3000 products, this results in 3005 features to feed into our model. While this is a large number of features, reducing it would lead to a loss of valuable information. Using a metric that replaces the vector $(\alpha_{ji})_{i,i\in N}$ (e.g., its mean) is suboptimal, as it would result in information loss. Hence, we retain the individual substitution probabilities to preserve the richness of information.

After data pre-processing, we proceed with selecting the most suitable regression algorithm for our dataset. To achieve this, we leverage a Python library called PYCARET, which automates the process of running various regression algorithms and comparing their performances.

PYCARET simplifies the task by automating the algorithm selection and evaluation process. It efficiently tests multiple regression models and provides valuable insights into their respective performances, allowing us to identify the best-performing algorithm for our specific data.

After the comparison performed by PYCARET, we get the following results:

Model	MAE	MSE	RMSE	R2	RMSLE
rf	0.0319	0.0058	0.0752	0.4034	0.0604
et	0.0322	0.0060	0.0758	0.4019	0.0607
lightgbm	0.0359	0.0063	0.0775	0.3841	0.0625
knn	0.0309	0.0067	0.0804	0.2812	0.0643
gbr	0.0373	0.0072	0.0838	0.2370	0.0675
ada	0.0629	0.0078	0.0870	0.1992	0.0729

Table 2: Regression Model Performance

Despite the relatively low R2 values, which can be attributed to the large number of features and the moderate dataset size, we decide to proceed with the Random Forest Regressor. The choice is driven by our specific objective, as our primary goal is not to achieve high precision but rather to rank the products based on their scores.

As the final step before using our algorithm to predict the scores, we conduct hyperparameter tuning to optimize its performance. This results in increasing the R2 value to 0.54!

3.3.5.3 Final results

Now that we have the scores for each product in our list, we proceed with the following steps:
1. Rank the products in descending order based on their scores.
2. Select products i from the ranked list, each with a quantity $u_i = K_T d_i$, until the shelf constraint C is reached. The chosen products form the assortment set A, while the remaining products are in set B.
3. Update the values u_i to include the assortment-based substitution terms:

$$u_i = K_T d_i + \delta K_T \sum_{j \in B} \alpha_{ji} d_j$$

4. Compare the sum of u_i for products in set A with the constraint C. If it exceeds C, adjust sets A and B by removing products with the least scores until the constraint is met.
5. Repeat steps 3 and 4 until an equilibrium is reached.

By implementing these steps, we obtain the optimal assortment using our heuristic approach.

To evaluate the optimality of our solution, we compare the objective values obtained from the exact resolution and the heuristic. The results demonstrate that our heuristic performs well.

When comparing the runtimes of the two algorithms, especially on larger product sets (e.g., 10000 products), our heuristic significantly outperforms the traditional solution. For instance, the exact solution may take approximately **11 days** for 200 products, while our heuristic takes only around **1 hour** for 10000 products.

4 Stochastic model: modeling under demand uncertainty

In this section, our focus shifts towards a comprehensive exploration of a "complete" version of the problem. Here, we will delve into an enriched model that incorporates demand uncertainty, assortment base substitution, and stock-out based substitution. The decision to address stock-out based substitution within the stochastic resolution framework was deliberate, as stock-outs lack meaningful interpretation in a deterministic context. We will begin by introducing the parameters of our model, followed by the formulation of the problem itself. Then, we will explore different resolution methods, engaging in a comparative analysis to identify the most suitable approach. Once a resolution method is selected, we will proceed with its implementation and thorough testing.

4.1 The model's parameters

In the deterministic setting, we consistently rely on the established parameters of the assortment based substitution framework, which included the demand vector $\boldsymbol{d}$, the substitution probability , the substitution rate matrix $(\boldsymbol{\alpha_{ij}})_{i,jN}$, the price vector $\boldsymbol{p}$, and the cost vector $\boldsymbol{c}$.

However, when dealing with stochastic scenarios, the presence of demand uncertainty introduces a new dimension of stock-out based substitution. To comprehensively capture the customers' substitution behavior in this context, we introduce **the stock-out substitution matrix** $(\boldsymbol{\beta_{ij}})_{i,j\in N}$. This additional matrix complements the existing parameters and fully characterizes the dynamic nature of substitutions.

The decision variables x and u remain unchanged.

In the following, the number of customers $\boldsymbol{K_T}$ and their demand $\boldsymbol{d}$ are unknown. Instead, this information is randomly drawn from some finite space $\boldsymbol{\Omega}$.

We begin our exploration in the stochastic setting by focusing on the stochastic variable $\boldsymbol{K_T}$, representing the number of customers arriving at the store during period T. Subsequently, we advance to the next phase, where our attention shifts to the uncertainty surrounding the demand variable $\boldsymbol{d}$. This progression enables us to specifically address and account for the inherent variability and unpredictability in customer demand.

4.2 Customer Arrival Variability: Stochastic Variable $\mathbf{K}_T$

4.2.1 Stochastic formulation

We consider $\boldsymbol{K_T}$ to be a random variable drawn from our finite sample space $\boldsymbol{\Omega}$ of nonnegative integer values.

In this setup, we introduce two additional factors:

- **Minimum facing $\boldsymbol{f}$:** For each product, we establish a minimum quantity that should be displayed on the shelf.
- **Maximum product limit within a category MAX:** To prevent overwhelming customers with an extensive selection of products, we set a maximum threshold for the number of products allowed within a given category.

Our formulation becomes:

$$\max_{\mathbf{S}\subseteq\mathbf{N}} \mathbb{E}_{K_T}\left[\sum_{i\in N} p_i m_i(\omega) + \sum_{i\in N}\sum_{j\neq i} p_i\delta\alpha_{ji}q_{ji}(\omega) + \sum_{i\in N}\sum_{j\neq i} p_i\delta\beta_{ji}s_{ji}(\omega) - \sum_{i\in N} c_i u_i\right]$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{i \in N} w_i u_i \leq C \\
& \sum_{i \in N} x_i \leq \mathbf{MAX} \\
& u_i \leq M x_i && \forall i \in N \\
& u_i \geq \boldsymbol{f}_i x_i && \forall i \in N \\
& m_i(\omega) \leq u_i && \forall i \in N \ \forall \omega \in \Omega \\
& m_i(\omega) \leq K_T(\omega) d_i && \forall i \in N \ \forall \omega \in \Omega \\
& q_{ij}(\omega) \leq M x_j && \forall i \in N \ \forall j \in N \ \forall \omega \in \Omega \\
& q_{ij}(\omega) \leq M(1 - x_i) && \forall i \in N \ \forall j \in N \ \forall \omega \in \Omega \\
& s_{ij}(\omega) \leq M x_i && \forall i \in N \ \forall j \in N \ \forall \omega \in \Omega \\
& s_{ij}(\omega) \leq M x_j && \forall i \in N \ \forall j \in N \ \forall \omega \in \Omega \\
& m_i(\omega) + \sum_{j \neq i} q_{ij}(\omega) + \sum_{j \neq i} s_{ij}(\omega) = K_T(\omega) d_i && \forall i \in N \ \forall \omega \in \Omega \\
& m_i(\omega) + \sum_{j \neq i} q_{ji}(\omega) + \sum_{j \neq i} s_{ji}(\omega) \leq u_i && \forall i \in N \ \forall \omega \in \Omega \\
& x \text{ in } \{0,1\}^n,\ u,\ m \text{ in } \mathbb{N}^n,\ q,\ s \text{ in } \mathbb{N}^{n^2}
\end{aligned}
$$

In our updated formulation, the objective function consists of four components. Firstly, we have the original demand, which is represented by the first part of the sum. Secondly, we include assortment-based substitution, similar to the previous formulation, which is represented by the second part of the sum. Additionally, we introduce a new component related to stock-out based substitution, represented by the third term of the sum. Lastly, we continue to consider the inventory cost by incorporating the fourth term of the sum as before.

Below, we define the parameters of our model, where the index T represents the specific day or period being considered.
We have:

Decision variables:

. x_i: Binary decision variable that takes a value of 1 if product i is included in the assortment and 0 otherwise.
. u_i: Integer decision variable indicating the quantity of product i stocked over T.
. $m_i(\omega)$: equals $\min(K_T(\omega)d_i, u_i)$. in the scenario ω
. $q_{ij}(\omega)$: the quantity of product i substituted by j due to assortment based substitution in the scenario $\omega \in \Omega$.
. $s_{ij}(\omega)$: the quantity of product i substituted by product j due to stock-out based substitution in the scenario $\omega \in \Omega$.

Model parameters:

. $K_T(\omega)$: The number of customers who visit the store during the period T in the scenario $\omega \in \Omega$
. p_i: The unit price of product i
. d_i: The demand for product i.
. δ: The probability of considering substituting a preferred product.
. αij: The assortment substitution rate from i to j.
. βij: The stockout substitution rate from i to j.
. c_i: The storage cost per unit for product i.
. w_i: Volume per product i.
. $\boldsymbol{f}_i$: the minimum facing of the product i.
. **MAX**: the maximum number of products to consider for the category studied.
. M: Big-M constraint.

Constraints:

The first constraint captures the spatial limitations within the designated category section, ensuring that the available space is not exceeded. The second constraint enforces a maximum limit, ensuring that no more than a specified number of products are included in the assortment.

To incorporate the concept of exclusivity, the third constraint utilizes the Big-M constraint technique. It guarantees that if a product (let's call it "i") is not included in the assortment, its quantity (x_i) will be zero.

Moving on, the fourth constraint emphasizes that for each product chosen in the assortment, a minimum quantity (f_i) must be displayed to meet visibility requirements. This ensures that products receive sufficient exposure.

The fifth and sixth constraints impose minimum values on u_i and $K_T(\omega)d_i$ for $m_i(\omega)$, ensuring that they align with the respective product's requirements.

Continuing with the assortment-based substitutions, the seventh and eighth constraints establish the relationship between the assortment quantities $q_{ij}(\omega)$ and the presence of products "i" and "j" in the assortment. If "j" is included and "i" is not, then $q_{ij}(\omega)$ represents the quantity of product "i" substituted by product "j".

Similarly, the ninth and tenth constraints connect the quantity $s_{ij}(\omega)$ to the presence of both products "i" and "j" in the assortment, indicating the amount of product "i" substituted by product "j" when both are included.

To address the demand for a particular product, the last two constraints come into play. The first ensures that the total quantity provided, derived from either the direct items of product "i" or items of product "j" available for substitution, equals the demand for product "i".

Lastly, the final constraint safeguards against exceeding the available quantity u_i, ensuring that the quantity provided to fulfill the demand for product "i" remains within the specified limit.

4.2.2 Exploration of resolution methods

Having formulated our problem, the next step is to explore the various methods available for solving it within the stochastic programming framework. Fortunately, it provides a range of methods, each tailored to specific problem settings, offering us a diverse set of approaches to tackle our problem.

- **Stochastic Dynamic Programming (SDP):** enables sequential decision-making by decomposing the problem into smaller subproblems and solving them iteratively. In the specific context of assortment optimization, this approach, known as Sequential Decision Processes (SDP), offers the flexibility to dynamically adjust the assortment based on the fluctuating number of customers arriving. However, it should be noted that our adopted framework for this problem does not allow for changes to the assortment during the designated period T.

- **Scenario-based Methods:** These methods discretize the uncertainty of customer arrivals into scenarios or samples. By constructing deterministic optimization problems based on these scenarios, they provide insights into the selection and allocation of products within the assortment considering various customer arrival scenarios.

- **Stochastic Decomposition:** Assortment optimization problems can be decomposed into deterministic subproblems, facilitating independent optimization of product selection and inventory allocation. This approach allows for efficient decision-making by considering the stochastic nature of customer arrivals in each subproblem.

- **Robust optimization:** it prioritizes worst-case scenarios to ensure resilience. By considering a range of customer arrival patterns, these techniques aim to identify assortments that perform well under the most challenging conditions. The focus is on mitigating the impact of uncertainty by selecting robust assortments that can withstand unfavorable outcomes. This approach enhances the reliability and stability of the optimization process, providing safeguards against potential losses or negative impacts.

- **Simulation-based Optimization:** Simulation-based optimization leverages the stochastic nature of customer arrivals by using simulation models to generate outcomes based on different customer arrival scenarios. Optimization algorithms then find the optimal assortment configuration based on the simulated outcomes, accounting for customer variability.

We will use the scenario-based method to address our problem, with a potential inclusion of a robust optimization component towards the end of the resolution. This approach allows us to consider various customer arrival scenarios and gain insights into the optimal solutions. By incorporating robust optimization techniques, we can further enhance the resilience and feasibility of the obtained solutions, ensuring their effectiveness across different scenarios.

4.2.3 Implementation and testing

4.2.3.1 Generation of scenarios

Using the Poisson distribution for modeling the number of customers arriving at the store

during period T is a common choice in scenarios where discrete count data is encountered. The Poisson distribution is particularly suitable because it allows us to model events that occur with a known average rate, represented by the parameter λ. In this case, we set λ to 100, indicating an average of 100 customers expected during period T.

The Poisson distribution offers simplicity in implementation and is well-suited for handling rare events, which can be significant in customer arrival patterns. Additionally, it enables the generation of different scenarios by randomly drawing K_T values from the distribution.

Furthermore, the parameter λ can be adjusted based on the duration of period T, allowing customization to reflect different average rates of customer arrivals depending on the time interval. This adaptability makes the Poisson distribution a versatile choice for modeling customer flow in a store during varying time periods.

We set the parameter λ or the mean of the distribution to be equal to 100.

$$\lambda = 100$$

In this case, we assume that during period T, the expected number of customers is 100. It should be noted that the value of λ can be chosen differently depending on the duration of period T.

4.2.3.2 Runtime

To begin our analysis, we focus on the efficiency of our algorithm, which we evaluate using the runtime key performance indicator (KPI). It is important to note that this time, the runtime is influenced not only by the size of the set of SKUs we choose from but also by the number of scenarios considered.

As always, having a larger set of products to choose from can lead to increased profitability. The same principle applies to the number of scenarios: the more scenarios we consider, the closer we come to the actual optimal value, leveraging the law of large numbers. Therefore, our goal is to consider the largest number of products and scenarios available to us. However, it is essential to acknowledge that there is always a trade-off to be made due to computational constraints.

In the subsequent analysis, we will examine the relationship between runtime and the number of scenarios and products. This can be expressed as a function:

$$\text{runtime} = f(N_{\text{scenarios}}, N_{\text{products}})$$

However, it is important to note that for this investigation, we will be working with smaller quantities of products and scenarios compared to the earlier analysis. The reason lies in the fact that the runtime explodes going from a few minutes when dealing with a small number of products to over an hour, as observed with 40 products, for example.

We obtain the following result:

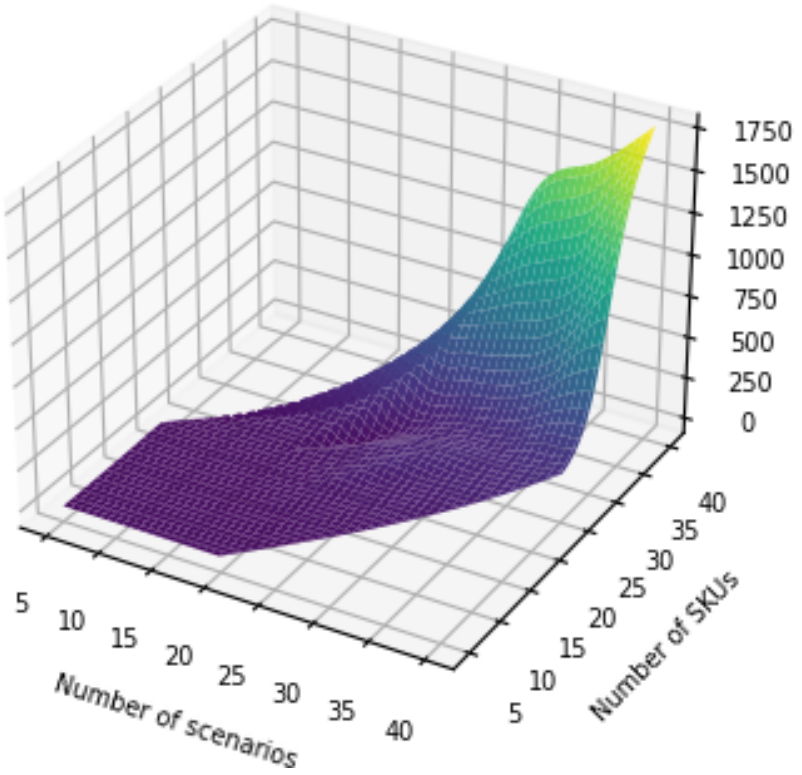


Figure 7: Runtime as a function of the number of scenarios and the number of scenarios

We conclude that a heuristic could be of great use when dealing with larger products' set.

4.2.3.3 Solution's stability

In the following analysis, we delve deeper into our investigation. A crucial aspect to explore is determining the number of scenarios at which the solution (profit, x, u) stabilizes. This knowledge is essential as it allows us to identify the ideal number of scenarios needed to achieve an actual optimal profit and assortment.

To accomplish this, we work with a predefined set of products and solve the stochastic program under various numbers of scenarios. For each value of N, we generate a set of scenarios S_N with a cardinal of N, drawn from the Poisson distribution mentioned earlier. We observe the evolution of the optimal solution as a function of the number of scenarios considered. The following are the results:

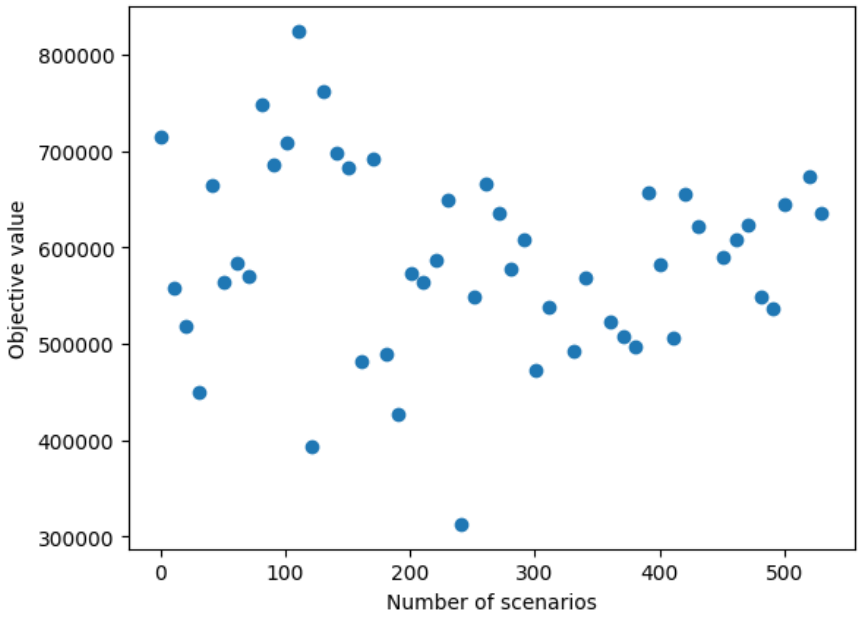


Figure 8: Objective value's stability with the number of scenarios

As the number of scenarios increases, we observe a reduction in the variance of the objective value. Due to computational constraints, we were unable to explore larger numbers of scenarios in-depth. However, based on theoretical insights, it is confirmed that the objective value will converge as the number of scenarios substantially increases.

Based on our analysis, we conclude that using 500 scenarios from this point forward in our stochastic program would be acceptable.

4.2.3.4 Solution's optimality

In this section, our goal is to assess the optimality of our solution. To achieve this, we will attempt to compute an upper bound against which we can compare our results. Let S represent the set of scenarios under consideration. A natural upper bound is defined as follows:

$$\Pi_{max}(x, u, S) = \frac{1}{|S|} \sum_{\omega \in S} \Pi(x, u, \omega)$$

To provide further clarity, we follow the process of generating the set of scenarios S and proceed to solve the problem deterministically for each individual ω within S. We calculate the optimal objective value $\Pi(x, u, \omega)$ for each scenario. By averaging over all these objective values $\Pi(x, u, \omega)$, we obtain our upper bound.

To ensure the validity of our analysis, we iterate over different sets of scenarios S_N, defined by their cardinal N. For each set S_N, we calculate the following value:

$$F(x, u, S) = \frac{\Pi(x, u, S)}{\Pi_{max}(x, u, S)}$$

This analysis demonstrates that the objective value reaches 35% of the upper bound for up to 500 scenarios.

It is important to highlight that the upper bound established earlier remains applicable regardless of any heuristics we may consider in the future.

4.2.3.5 Solution's robustness

The analysis conducted previously raises concerns regarding the suitability of the selected stochastic programming framework. It becomes apparent that we experience a loss in optimality without any discernible advantages. This situation arises mainly because we have not yet accounted for the robustness effects that the stochastic framework offers to our solution.

To address the robustness effect our scenario based method has on the solution, we may want to compare the deterministic solution to the stochastic one.

For such comparison to hold:

1. We generate a random set R of K_T always from the Poisson distribution defined earlier.

2. We solve the problem deterministically using a random generated K_T value. We save the solution (x_d, u_d) for later use

3. We solve the problem under the stochastic framework using a set of scenarios S. We save the solution (x_s, u_s) for later use

4. for each K_T value in the set R, we compute the profit generated and the lost sales under both the solutions (x_d, u_d) and (x_s, u_s) .

The results are as follows:

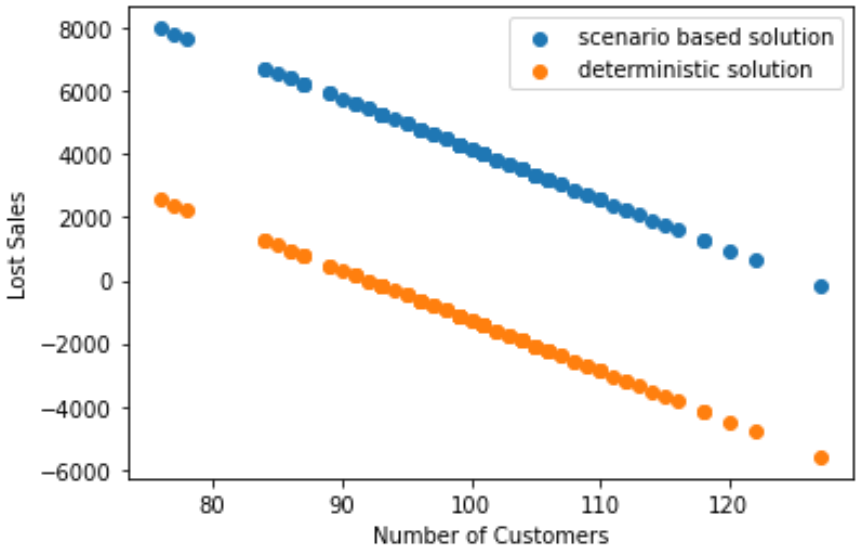


Figure 9: Robustness of the solutions

Based on our analysis, when we base our reasoning on a specific value of K_T (customers' arrival), the model may produce sub-optimal solutions, leading to potential losses as depicted in the figure. As we deviate from the randomly chosen $K_T = 92$, the solution becomes increasingly non-optimal compared to the solution generated by the scenario-based (stochastic)

method. By considering only 100 scenarios, we can typically prevent most losses.

5 Concluding remarks

In this report, we delved into the realm of assortment optimization in the retail industry, an essential aspect of retail analytics. We explored various formulations and algorithms to maximize the profit while adhering to shelf constraints.

Initially, we presented a linear formulation, which laid the foundation for our subsequent development. Through a Python implementation, we demonstrated how exact solutions can be derived, providing valuable insights into the assortment selection process.

We then ventured into a quadratic formulation, which introduced a thorough study of assortment-based substitution. Linearizing and leveraging the model's parameters, we attained exact solutions and devised a novel ML-based heuristic algorithm. Our scoring algorithm, implemented through an ML model, yielded promising results in optimizing the assortment.

In the pursuit of comprehensive assortment optimization, we also addressed the challenging problem of modeling under demand uncertainty. By introducing a stochastic approach, we recognized the importance of accounting for variability in the number of visiting customers, which is prevalent in real-world retail scenarios. The stochastic model allowed us to explore how customer arrival variability affects assortment decisions. We formulated the stochastic variables and explored various resolution methods to handle the uncertainty inherent in demand data. Our implementation and testing of stochastic scenarios revealed the stability, optimality, and robustness of our proposed solution.

As perspectives for future research, there are several important aspects that were not thoroughly studied in this report, presenting potential development avenues for further investigation:

1. Incorporating Demand Uncertainty: The demand data $\boldsymbol{d}$ plays a crucial role in assortment optimization, but in this study, we assumed deterministic demand values. Exploring the uncertainty that arises from demand variability could provide a more realistic representation of the retail environment. Future research could focus on modeling demand uncertainty using probabilistic or stochastic methods, allowing for more robust and risk-aware assortment decisions.

2. Developing a Heuristic for Stochastic Formulation: While we successfully formulated and explored the stochastic model, a dedicated heuristic tailored to this specific formulation was not developed in this report. The design of an efficient heuristic algorithm that handles stochastic scenarios effectively could significantly enhance the scalability and computational efficiency of the stochastic assortment optimization process.

To conclude, this report sheds light on the intricacies of assortment optimization for retailers. We've demonstrated the effectiveness of our algorithms in providing optimal assortments while accounting for shelf constraints and customer arrival's uncertainty. Our findings offer valuable insights and practical implications for retail businesses seeking to enhance their product assortments and better serve their customers. As retailers continue to embrace analytics-driven strategies, the methodologies discussed herein present compelling avenues for further exploration and potential implementation in real-world retail settings.

Future work could extend this study to address the identified perspectives, providing a more comprehensive understanding of assortment optimization in the face of demand uncertainty and dynamic market conditions. Additionally, incorporating an omnichannel structure and comparing various assortment optimization algorithms could further enhance the efficacy and adaptability of the assortment selection process.

6 Acknowledgments

I extend my heartfelt gratitude to ***Prof. Agnès GORGE*** for her invaluable guidance, mentorship, and insightful contributions throughout the course of this research. Her expertise in the field of Business Analytics has been instrumental in shaping the direction of my study and providing me with valuable perspectives. Her unwavering support and thoughtful feedback have greatly enriched the quality of this paper. I am profoundly thankful for her time, encouragement, and dedication to fostering a nurturing academic environment. This work stands as a testament to her impactful influence on my academic journey.

References

[1] Marshall L. Fisher A. Gürhan Kök. Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research, Informs*, 2007.

[2] Ralf W. Seifert Andrey Vasilyev, Sebastian Maier. Assortment optimization using an attraction model in an omnichannel environment. *European Journal of Operational Research*, 2022.

[3] Gijsbrechts Els Nisol Patricia Campo, Katia. Dynamics in consumer response to product unavailability: do stock-out reactions signal response to permanent assortment reductions? *Business Research, Elsevier*, 2004.

[4] Emmett Cox. Retail analytics: the secret weapon.

[5] David P. Williamson James M. Davis, Huseyin Topaloglu. Assortment optimization over time. *Cornell University*, 2013.

[6] Venus Hiu Ling Lo. Capturing product complementarity in assortment optimization. *Cornell University*, 2019.

[7] Roger Roberts et al. Michael Chui, Eric Hazan. The economic potential of generative ai.

[8] David W. Pentico. The assortment problem: A survey. *European Journal of Operational Research*, 2008.

[9] Harald J. van Heerde Robert P. Rooderkerk. Robust optimization of the 0–1 knapsack problem: Balancing risk and return in assortment optimization. *European Journal of Operational Research*, 2015.

[10] R. Spillman. Solving large knapsack problems with a genetic algorithm. *IEEE*, 2002.

[11] Square. Abbreviations 101. Square, 2021. Accessed on 01/07/2023.