Journal of Retailing and Consumer Services 50 (2019) 199–214

Contents lists available at ScienceDirect

Journal of Retailing and Consumer Services

journal homepage: www.elsevier.com/locate/jretconser

Assortment optimization with log-linear demand: Application at a Turkish grocery store

Mustafa Hekimoğlu[a,*], Ismail Sevim[b], Çağlar Aksezer[c], İpek Durmuş[c]

[a] Department of Industrial Engineering, Faculty of Engineering and Fundamental Sciences, Kadir Has University, Istanbul, Turkey
[b] Department of Industrial Engineering, Faculty of Mechanical Engineering, Yildiz Technical University, Istanbul, Turkey
[c] Department of Industrial Engineering, Faculty of Engineering, Işık University, Istanbul, Turkey

A B S T R A C T

In retail sector, product variety increases faster than shelf spaces of retail stores where goods are presented to consumers. Hence, assortment planning is an important task for sustained financial success of a retailer in a competitive business environment. In this study, we consider the assortment planning problem of a retailer in Turkey. Using empirical point-of-sale data, a demand model is developed and utilized in the optimization model. Due to nonlinear nature of the model and integrality constraint, we find that it is difficult to obtain a solution even for moderately large product sets. We propose a greedy heuristic approach that generates better results than the mixed integer nonlinear programming in a reasonably shorter period of time for medium and large problem sizes. We also proved that our method has a worst-case time complexity of $\mathcal{O}(n^2)$ while other two well-known heuristics' complexities are $\mathcal{O}(n^3)$ and $\mathcal{O}(n^4)$. Also numerical experiments reveal that our method has a better performance than the worst-case as it generates better results in a much shorter run-times compared to other methods.

1. Introduction

Retailing business and all its dimensions went through a drastic change with the turn of the century due to increasingly personalized consumer needs, growing number of products, and drastically changing way the sales are conducted (Clay et al., 2002). Empirical evidence indicates that increasing amount of customers seek products that are suited to their individual needs (Ulu et al., 2012). Personalized consumer demand lead manufacturers to design different products to stay competitive as consumers rarely hesitate to switch to another brand (or retail store) when they are dissatisfied with the actual one, a.k.a. low consumer loyalty.

To keep up with this drift, supermarkets tend to increase the range and variety of products that they offer to their customers. Highly diversified customer needs and increasing number of candidate products force supermarkets to increase the variety of goods on their shelves for serving to larger number of consumers and maintain their market shares. However, shelf spaces of supermarkets usually stay the same as it requires significant amount of investment to increase them. Quelch and Kenny (1994) report that the number of goods in supermarkets increased by 16% per year between 1985 and 1992, while shelf space expanded by only 1.5% per year in the same period. In addition to limited shelf space, increasing product variety stands for higher handling costs, more frequent markdowns, and possible loss of economies of scale due to smaller order quantities. Therefore, finding the correct product assortment out of a large candidate set and offering them in a limited shelf space is critical for financial success of supermarkets (Cadeaux, 1999).

Assortment planning is defined by the set of products to be sold at a store chosen out of a larger candidate product pool. Assortment planners aim to choose products that maximize retailers' revenue by considering customers' needs and tends to stay competitive in the market. Assortment decision can be more complicated by extra constraints such as inventory level, shelf space, and minimum number of brands to be stored for each product (Kök and Fisher, 2007). Among these complicating factors, shelf space assignment to each product significantly increases the importance of assortment optimization for a retailer especially in case of limited (or absent) storage facility. When products are directly placed on to shelves, the shelf space assignment determines the maximum inventory level that can be stored in whole supermarket within a replenishment period. Assigning limited shelf space to a product with high demand rate might lead to substantial profit loss due to repeating stockouts.

From a practical perspective, assortment planning is a one-time decision usually taken by managers at the beginning of a selling season. In some cases, it might be necessary to update product assortments due to shifting customer profile or introduction of new products. Hence it might be important to have a solution that generates an output in a reasonable amount of time while satisfying business requirements (Hübner et al., 2013).

* Corresponding author.
E-mail addresses: mustafa.hekimoglu@khas.edu.tr (M. Hekimoğlu), isevim@yildiz.edu.tr (I. Sevim), caglar.aksezer@isikun.edu.tr (Ç. Aksezer), ipek.durmus@isikun.edu.tr (İ. Durmuş).

https://doi.org/10.1016/j.jretconser.2019.04.007
Received 6 July 2018; Received in revised form 23 March 2019; Accepted 5 April 2019

This study develops an assortment optimization scheme that maximizes revenue for a single grocery store that operates as branch of a supermarket chain in Turkey. In the supermarket database, there are more than ten thousand products, and hierarchically all the products are divided into groups, categories and subcategories. This large set of products are located on the shelves. Since placing a brand behind an another is not acceptable, frontage of shelves are assigned to products of certain brands. From another perspective, all available shelf space is divided into facings, which is defined as the number of products that are visible in front of a shelf.

In order to characterize the customer demand, we developed a regression model using point-of-sale data from the supermarket database. Demand is modeled in log-linear fashion, which is the first of its type in the assortment planning literature to the best of our knowledge. After proper validation tests, the regression model is used as a deterministic choice model in the assortment optimization. The model has a mixed integer nonlinear programming (MINLP) nature, aiming to determine the facing levels for all products that maximizes the revenue.

Experiments with the assortment optimization solution reveals that the MINLP model can only be solved for small problem sizes using a standard solver (BARON). For medium and large product sets, the solver generates a very large optimality gap and no feasible solution respectively even after 10 hours of computation time. To remedy this, we develop a greedy heuristic that assigns facings to products iteratively. Numerical experiments and comparison of our results to two meta-heuristics, genetic algorithm and simulated annealing, indicate that our solution method performs significantly better than BARON solver with medium and large problem sizes. Furthermore, the proposed greedy heuristic generates better results than the two meta-heuristics in a much shorter runtimes for all parameter combinations of our test bed. These results indicate the importance of utilization of the mathematical structure of the problem for development of efficient solution algorithms.

This paper is organized as follows: In the next section relevant literature is reviewed. In Section 3 data set and the empirical demand model is presented. The assortment optimization model is presented in Section 4. We conclude our work in Section 5.

2. Literature review

Assortment planning gained importance within the retail sector as a result of fierce competition and emergent requirements in customer satisfaction (Mantrala et al., 2009). Kök et al. (2008) provide an in depth review of the literature from all extents such as inventory control, demand estimation, and assortment optimization.

Inventory control of the retailer is the key to its success from a managerial point of view. Önal et al. (2016) propose an EOQ model for perishable products under the guidance of assortment optimization. They utilize a Tabu Search heuristic algorithm to overcome the computational difficulties of this MINLP problem. Stockouts are a major issue of inventory control. Assortment optimization helps the retailers to quickly identify available shelf space in order to respond stockout situations. Real time shelf space detection systems are readily available to be used in such modeling for daily retailing operations (Frontoni et al., 2017). On the other hand, typical response of the customer to stockout instances arise as substitution behavior. Absentee product of interest is substituted with a different product of the same category, which makes the assortment planning activities even harder as the demand might be random (Honhon et al., 2010; Gilland and Heese, 2013; Goyal et al., 2016). Return policies of the retailers also have impact on assortment planning as an important dimension of inventory control. A study by Alptekinoğlu and Grasas (2014) conclude that the optimal assortment generally implies strict return policies. This leads to assortment that constitutes a balance of risk between consumer and retailer.

Demand estimation and choice modeling have been the most attractive part of the assortment planning literature for researchers. Since the decision on existence or nonexistence of a product within the assortment is a binary decision variable and availability of product varieties are categorical variable, logit functions are popularly employed to model these relationships. Rusmevichientong et al. (2010) consider an assortment optimization problem under the multinomial logit model. The parameters of the multinomial logit model are random because every consumer has different taste for the products of their interest. Similarly, Rusmevichientong et al. (2014) state that assortments are composed of the products which provides the highest revenue and this is called as *revenue-ordered assortments*. In the special case, the nested logit model, products are grouped and organized in nests. A customer coming to the store can decide to make a purchase from one of the nest or leave without purchasing any product. If a nest is chosen, the customer buys one of the product available in that specific nest (Davis et al., 2014). Capacity of a nest plays an important role on the optimal assortment as pointed out by Feldman and Topaloglu (2015). Although we don't incorporate the product pricing to our models, literature consists many studies involving this crucial component of assortment planning (Kök and Xu, 2011; Li et al., 2015).

According to Fisher and Vaidyanathan (2014), the assortment varies based on the local tastes of each store. Their study provides improvement of assortment localization, allowing a constraint on the number of different assortments and quantifying the level of localization effects on revenue. A ranking based consumer choice model is used in order to show consumer tastes by Honhon et al. (2012). According to this choice model, every consumer has their own rankings for the potential products and they intend to buy their highest ranked product offered within the assortment. Golrezaei et al. (2014) handle the problem from a personalization perspective. An intensive use of real time data collection and utilization is required to perform this task on individual customer base. For example; the product suggestions by Amazon.com for each customer dynamically changes depending on factors such as the previous purchases, recent reviews, purchases of other customer who has similar tastes etc. Authors claim that increment in revenue is possible by personalizing the assortment. These studies illustrate that consumer choices and tastes directly affect and determine the assortment structure.

Hwang et al. (2009) assume the demand rate of each product as a function of the location and the displayed inventory level of the product. They developed a mathematical model to maximize the profit of the retailer. Proposed solution includes two segments; one as a slicing structure by guillotine type cuts and other by slicing structure as horizontal cuts only, which both are solved via Genetic Algorithm (GA). Another solution to two-dimensional assortment problem by GA is proposed by Lin (2006). A problem specific encoding scheme which incorporates a novel packing process is used in the GA. Two-dimensional assortment problem is a very difficult problem even when the problem size is not too large. However, numerical results shows that the GA is more effective and efficient than an integer programming optimization scheme.

Such computational inefficiencies on exact solutions, especially in medium to large scale problems, have made heuristic approaches very attractive. Kök and Fisher (2007) determine a methodology in order to forecast the input demand and substitution parameters for the assortment planning problem and develop an assortment optimization algorithm. An iterative heuristic which can solve a series of separable nonlinear knapsack problem was proposed. Their methodology, applied at a leading supermarket chain in the Netherlands, provided more than a 50% increase in profits. Esoteric approaches based on Game Theory principals are also applied successfully, especially in the presence of a competitive market environment. Such studies generally handle the problem in an aggregate manner of the whole market where price and product variety are optimized simultaneously (Hopp and Xu, 2008; Aydin and Heese, 2014; Federgruen and Hu, 2015).

3. Data analysis and demand estimation

Every planning activity in operations management is initiated by demand forecasting. Assortment planning is no exception and requires substantial data analytics activities such as collection, storage, and analysis. This section presents the basics of developing an empirical demand model by regression analysis. Demand function will be utilized as a hard constraint in the assortment optimization model. In depth analysis of the demand model is going to be carried out for further validation of the mathematical function.

3.1. Data set

The retailer under investigation is located in Ankara. Researchers have been given access to company the database including inventory levels, facing quantities, sales quantities and price for every product of all product groups for each day of thirteen weeks.

In a typical supermarket branch, assortment is divided into different groups such as grocery food, nonedibles, fruit and vegetable products, household products, personal care products etc. These product groups branch out categories, subcategories and products hierarchically. For an example to this hierarchy, dairy products, milk, skim milk and A-Brand skim milk can be sorted respectively (see Fig. 1). Sales data contains 40000 products under 677 product groups for this supermarket. The dataset consists of 65050 rows of records. A data preprocessing on stock, facing and sales values reveals some inconsistent records such as negative stock values or extremely high remaining stock quantities.

A detailed investigation into the data source indicates that these inconsistencies stem from transactional errors in the company's databases. After elimination of these false records, we obtained a dataset consisting of 62271 rows. Descriptive statistics of the numeric columns of the dataset are presented in Table 1.

Our demand model considers weekly sales, aggregated from the original data set, for each product. In addition to sales, the dataset includes inventory and facing values for each product. During the aggregation process, we consider weekly average of inventory levels and facing to account for the effect of declining inventory (and lost sales) and assortment updates that might take place in the middle of a week. Our resulting (aggregated) data set consists of thirteen weeks of sale, inventory and facing values for all products in the candidate set.

3.2. Demand estimation by regression

In order to estimate the customer demand a regression analysis is conducted for all candidate products in the data set. The predictor values are as follows: Product ID, product group IDs, stock level and the facing quantity.

For modeling sales data, the log-linear regression is found to be preferable than linear regression. Although more practitioner friendly, linear regression has a certain shortcoming. It may produce negative demand values and this causes numerical difficulties when solving an optimization problem. Unlike the linear demand model, log-linear demand model always yields nonnegative values. Moreover, it can easily be recovered to linear form by taking the logarithm of demand (Talluri and Van Ryzin, 2006).

Table 1
Descriptive statistics for sales data.

	Stock	Facing	Sales Quantity
Average	34.39	17.11	1.47
Standard Dev	73.55	13.67	5.72
Min	0.00	3.00	0.00
Max	996	210	309

Different regression models are conducted by using the predictor variables diversely in order to reach the demand model that fits best. Using the predictor variables diversely assures that different variables are used as predictors in each regression analysis. Resulting regression models are compared through the coefficient of determination strength, R^2 values, as well as Akaike Information Criterion (AIC). Specifically, in the literature AIC is found to be more appropriate than adjusted-R^2 for model selection processes (Hand et al., 2001). In our application we consider both criteria to ensure best accuracy while avoiding overfitting. In each mode, we provide statistics for AIC as well as R^2 value.

3.2.1. Demand model

Demand modeling phase of this study includes analysis and comparison of fifteen different regression models to find the best model. In each model, the logarithm of weekly sales quantity is considered to be the dependent variable whereas amount of facings, remaining stock, prices of different products are taken to be the independent variable. These variable selection is motivated by the fact that retail sales are directly or indirectly effected by the amount of shelf-space allocated to a product Kök et al. (2008). To model the functional relationship between dependent (logarithm of sales quantity) and independent variables, we consider a linear model of which residuals follow Normal distribution.

Among 15 different regression models the one with the highest R^2 value and the lowest Akaike Information Criterion (AIC) is selected as the demand model of this study. In order to examine the regression analyses extensively, readers may refer to Durmuş (2017). A different application of regression models with transformed independent variables can be found in Hekimoğlu and Barlas (2016).

The usage of a dependent variable obtained from a log-linear regression model requires retransformation, which creates retransformation bias (Duan, 1983). In our study, estimated demand of each product are evaluated in an exponential function, $e^{(.)}$. The retransformation bias is treated using the smearing correction factor, $\frac{1}{n}\sum_{i=1}^{n} e^{\varepsilon_i}$, suggested by Duan (1983). In this correction formula n stands for the estimated demand for product i, and n is the total number of products in the dataset. Such a correction of transformed dependent variable is also employed by Caro and Gallien (2012) in a different context.

An exemplary demand model is illustrated below. Regression analyses is conducted to estimate demand function by using *IDProductGroup*, *IDProduct*, *Stock* and *Facing* parameters as predictors. The result of this regression are shown in Table 2, Table 3 and Equation (1).

The regression equation is

$$\begin{aligned} \ln \mathit{Demand} = &-0.698 - (0.175 \times \mathit{IDProductGroup}) \\ &- (0.024 \times \mathit{IDProduct}) \\ &+ (0.009 \times \mathit{Stock}) + (0.013 \times \mathit{Facing}) + \varepsilon, \end{aligned} \tag{1}$$

where ε follows Normal distribution. In the regression model p-values of all predictor variables are smaller than 0.05, so these predictors are significant for demand estimation. The adjusted R^2 value is 0.7082, and it shows that the regression model explains 70.82% of variability present in the customer demand. Moreover, AIC of value of this model is

Table 2
Coefficients of the regression equation.

Term	Coefficient	P-value
Constant	−0.698	0.123
IDProductGroup	−0.175	0.145
IDProduct	0.024	0.005
Stock	0.009	0.000
Facing	0.013	0.001

Table 3
Model summary of the regression equation.

R-sq	R-sq(adj)	R-sq(pred)	AIC
71.06%	70.82%	69.36%	639.58

lower than the all candidate models discussed by Durmuş (2017). Low AIC value indicates a proper model fit in nonlinear regression methodology.

The coefficients of independent parameters demonstrate that IDProduct effects the demand negatively, and the demand increases with the increment of Stock, Facing and IDProductGroup parameters.

$$Demand = e^{-\beta_0+\beta_1 IDProductGroup-\beta_2 IDProduct+\beta_3 RemainingStock+\beta_4 Facing+\varepsilon} \tag{2}$$

According to the Fig. 2, the residuals are scattered in an unbiased manner with minimal outliers. As a result, we conclude that the residual plot justify regression assumptions and the usage of the model in the optimization.

3.2.2. *Validation of the regression model*

Estimation of prediction accuracy is essential for measuring the performance of regression models. Cross-validation is a common model validation technique to estimate the prediction accuracy (Fushiki, 2011).

k-fold cross-validation divides the data set into *k* complementary subsets. $k-1$ subsets are used to estimate model parameters (training) and one subset is used for testing. Running this iteratively (by using different parts for testing) uncovers potential weakness of the model that might be averaged out when the entire data set is used for training. In this study, five-fold cross validation is used to validate the demand model. Steps involved in k-fold cross validation are explained in details below.

Firstly, the data set is divided into k-equal parts. In the literature, k is generally chosen as 5 or 10 depending on the size of the data. In our study, the sample data compose of thirteen weeks of data in the product set, so there are 130 rows of complete data. Setting k to 5, yields subsets consisting of 26 rows of data (for each product).

Secondly, the first subset is used for test set and the rest is kept for training. Regression parameters are calculated with the training set and the obtained model is used to generate demand estimations for the test set. Finally, the MAPE of the first part is calculated by using Equation (3) (Hanke and Wichern, 2009).

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|Y_t-\hat{Y}_t|}{|Y_t|}, \tag{3}$$

where Y_t is actual sales, $\hat{Y}_t$ stands for the demand estimation.

This procedure is repeated five times by using a different test set at each iteration. Results of the cross validation test, MAPE values of each part, and the average of all parts, demonstrate predictive power of the demand model. The MAPE values are shown in Table 4.

The mean absolute percentage error (MAPE) for cross-validation tests is found to be 155.65%. This value is rather high but customary to every demand prediction occurring in such a long time horizon. Uncontrollable factors involving managerial decisions in daily activities might trigger such large deviations. These may be the result of excess inventory levels or promotions initiated by certain suppliers. To improve the cross validation test accuracy, test data sets may be extended both in size and time perspective.

The same procedure is also applied to development of demand models for 500 and 5004 products. Results of these analyses are presented in Durmuş (2017). Once these models are finalized, we proceed to the assortment optimization phase of the study.

4. Assortment optimization problem

The assortment optimization problem aims to maximize a retailer's profit by assigning a limited amount of shelf space to a subset of products in the candidate product set. In this assignment problem, facings are used as the decision variables as products are lined up side by side along the shelf length. Total shelf space capacity, width and length of the products are significant factors affecting the facing decision. The width of a product determines how much shelf length each product's facing seize whereas the length of a product implies the maximum number of products that can be stored on the shelf as products are placed in tandem along the shelf depth. Total shelf capacity is the multiplication of shelf depth and shelf length. Fig. 3 illustrates these terms.

The limit of shelf length (LSL) is defined by taking into consideration of all shelving in the store, and it consists of a lot of different shelving which are located alongside or one on the top of the other. The shelf depth (SD) is designated as 60 cm, and it is a reasonable number

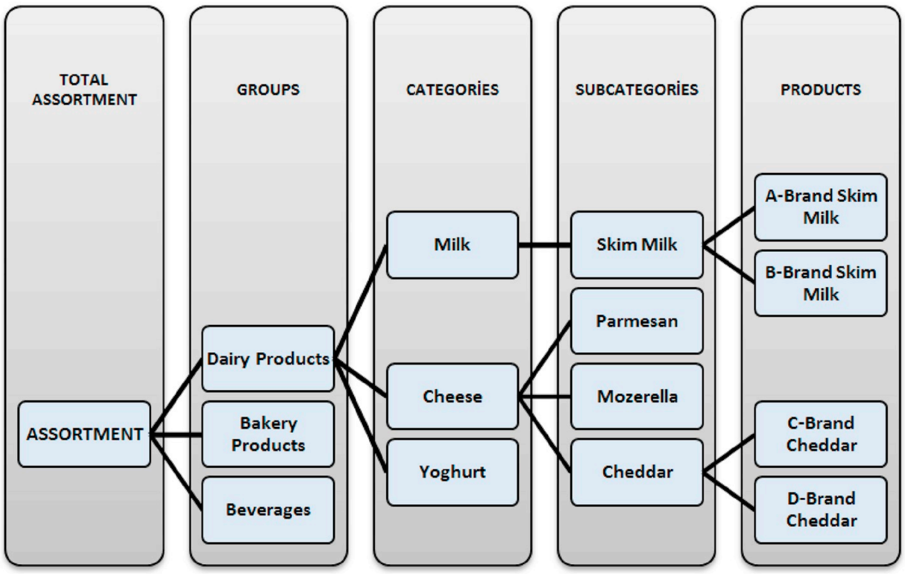

Fig. 1. The hierarchical data structure in the assortment.

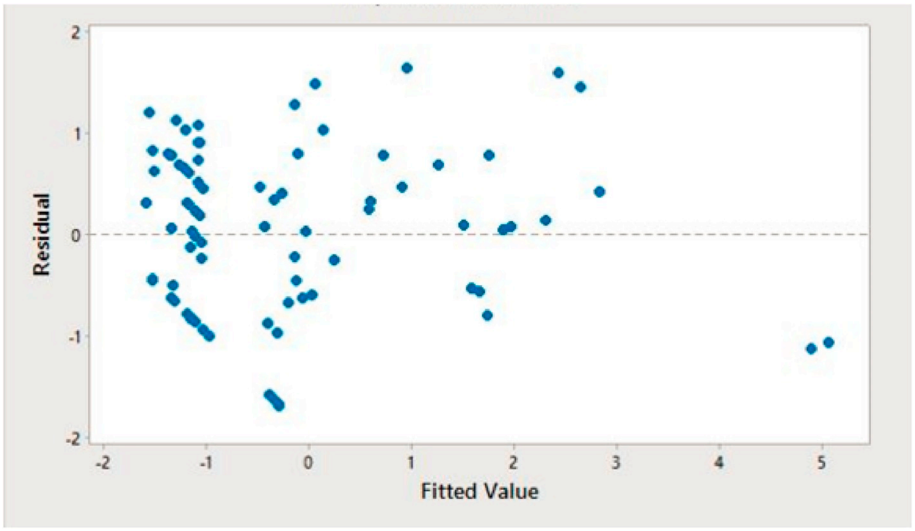

Fig. 2. The plot of residuals versus fits (Response is ln *Demand*).

Table 4
The result of cross-validation test on a sample data.

	MAPE
First part	60.35%
Second part	302.9%
Third part	87.75%
Fourth part	116.4%
Fifth part	210.9%
Average	155.7%

according to shelf standards of supermarkets. For the upper limit of facing (ULF), artificially large amounts, such as 50, are assigned.

4.1. Assortment optimization model

The mathematical model of the assortment problem is defined with the set of candidate products J(see Table 5). The objective of the assortment optimization model is to maximize total expected revenue which is the multiplication of price of each product and its expected sales. Therefore, the objective function is

$$z = \sum_{j\in J} p_j E[sales_j], \tag{4}$$

where $E[sales_j] = E[min(IL_j, d_j)]$ and IL_j is the random variable representing the inventory level of product j. As explained in Section 3.2.1, the demand model considered in this study is estimated from empirical data hence, it consists of a deterministic and a random part ($D_j = d_j e^{\varepsilon_j}$). Then the expected sales becomes as follows:

$$E[sales_j] = E[min((SD/l_j)f_j, D_j)|min((SD/l_j)f_j < D_j]Pr\{(SD/l_j)f_j < D_j\} + (SD/l_j)f_j Pr\{(SD/l_j)f_j \geq D_j\}. \tag{5}$$

Denoting the distribution of the error term with $f_\varepsilon(u)$ and some algebraic manipulation lead us to

$$\begin{aligned} E[sales_j] &= d_j \int_{u=0}^{(SD/l_j)f_j/d_j} uf_\varepsilon(u)du + (SD/l_j)f_j Pr\{(SD/l_j)f_j/d_j < \varepsilon\},\\ &=(SD/l_j)f_j + \int_{u=0}^{(SD/l_j)f_j/d_j} [d_j u - (SD/l_j)f_j]f_\varepsilon(u)du,\\ &=(SD/l_j)f_j - E[s_j], \end{aligned} \tag{6}$$

where

$$E[s_j] = \int_{u=0}^{(SD/l_j)f_j/d_j} ((SD/l_j)f_j - d_j u)f_\varepsilon(u)du. \tag{7}$$

For the rest of the paper, we drop the expectation operator from $sales_j$ and s_j for notational simplicity. We use Equations (6) and (7) in the mathematical model which can be expressed as follows:

$$\max \sum_{j\in J} p_j sales_j \tag{8a}$$

subject to

$$d_j = \beta_0 + \beta_1 gr_j + \beta_2 pr_j + \beta_3 s_j + \beta_4 f_j, \quad \forall j \in J, \tag{8b}$$

$$\sum_{j\in J} f_j w_j \leq \text{LSL}, \tag{8c}$$

$$f_j \leq \text{ULF}, \quad \forall j \in J, \tag{8d}$$

$$(\text{SD}/l_j)f_j - sales_j = s_j, \quad \forall j \in J, \tag{8e}$$

$$s_j = \int_{u=0}^{(SD/l_j)f_j/d_j} ((SD/l_j)f_j - d_j u)f_\varepsilon(u)du, \quad \forall j \in J, \tag{8f}$$

$$f_j \in \mathbb{Z}^+, \quad \forall j \in J, \tag{8g}$$

$$sales_j \geq 0 \quad \forall j \in J. \tag{8h}$$

The expected total revenue of the supermarket is maximized by the objective function (8a). The price of product $j \in J$ is stated by the parameter p_j, and the decision variable d_j indicates the expected demand of product $j \in J$.

The first constraint (8b) is the demand function in Equation (1)

Table 5
Notation of the model.

J	the set of candidate products
j	the index of candidate products in the set J
p_j	the price of the product $j \in J$
d_j	the demand of the product $j \in J$
gr_j	the group ID of the product $j \in J$
pr_j	the product ID of the product $j \in J$
s_j	the remaining stock of the product $j \in J$
f_j	the facing of the product $j \in J$
w_j	the width of the product $j \in J$
l_j	the length of the product $j \in J$
$sales_j$	amount of sales of the product $j \in J$

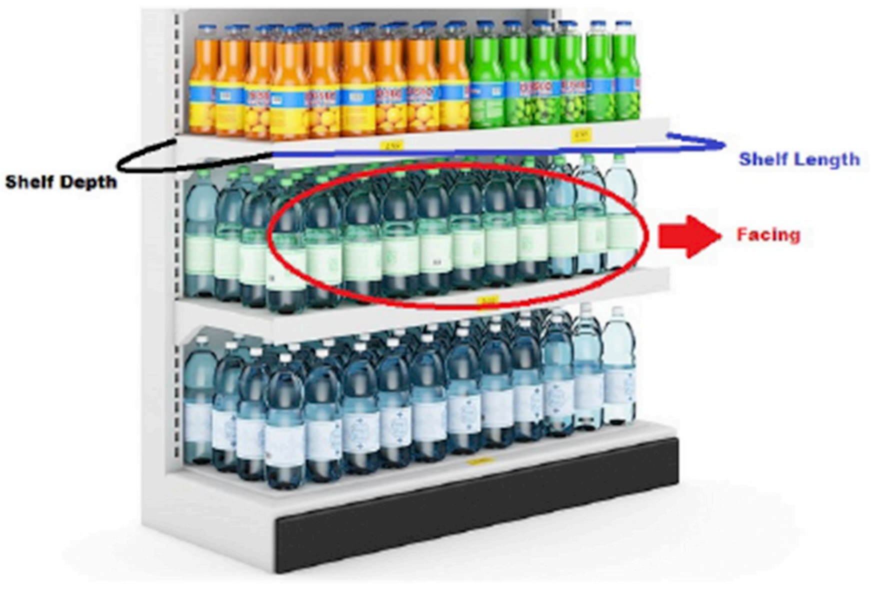

Fig. 3. An examplary shelf arrangement in a grocery store.

where α_i, $i \geq 0$ are greater than zero. Parameters gr_j and pr_j state the group and product codes (identity numbers) of the product j. The decision variable s_j is defined as the expected remaining stock of product j, and the decision variable f_j is defined as the facing quantities of product j. The shelf length, denoted by LSL, limits the facings assigned to each product. In constraint (8c), each product's width is denoted with w_j. A complete list of notation is given in Table 5.

In many supermarkets, products are located depending on the product group which they belong to such as detergents, personal care etc. Usually an additional constraint on facing is necessary in the optimization problem. Such an upper bound (8d) is denoted by ULF in the model.

Constraints 8e and 8f calculate expected sales and remaining stock by considering the maximum available shelf area assigned to product j. We assume that there is no backroom in the store, and all inventory is kept on the shelf. This assumption is consistent with the industry practice as supermarket chains try to avoid having inventory in backrooms to minimize inventory holding cost. In these constraints, SD is the shelf depth, and l_j is the length of product j. Nonnegativity of f_j and s_j are stated in the last two constraints.

4.2. A nonlinear programming model for the problem

4.2.1. Model analysis

Assuming sales is equal to demand and considering the mathematical model presented in (8a)-(8e), we obtain a nonlinear optimization problem for which some analysis on the objective function and the feasible set is required. The objective function (8a) is together with the constraint (8b) implies a convex increasing function of f_j given that β_3, β_4 are positive constants. The rest of the constraints (8c-8e) are linear functions of f_j and s_j which imply a convex feasible set. Therefore, our optimization problem is a convex maximization problem on a convex set. The following theorem presents the nature of the solution for the assortment problem.

Theorem 1. *The following statements hold.*

a) The solution of the assortment optimization problem presented in (4a-4g) has no interior solution, i.e. the optimum solution is in the extremes of the feasible set.
b) Constraint (4e) is a binding constraint for all solutions of the model.
c) $f_j > 0$ for all feasible solutions of the problem.

Proof. Define a set $\mathbf{I}$ including constraints (4c-4g), where the first element of this is the constraint (4c). Also by combining (4a) and (4b) we write the objective function

$$h(\bar{f}, \bar{s}) = \sum_{j \in J} p_j e^{-\alpha_0 + \alpha_1 gr_j - \alpha_2 pr_j + \alpha_3 s_j + \alpha_4 f_j},$$

where $\bar{x}$: $=(x_1, x_2, ..., x_m)$, where m is the number of products in the product set J.

Statement a: Take an interior solution $x \in \mathbb{R}^{2m}$, which is $g_i(x) < b_i$, $\forall\ i \in \mathbf{I}$. Then there exist an ε neighborhood whose elements are also interior solutions of the problem, i.e. $N_\varepsilon(x)$: $=\{y \in \mathbb{R}^{2m}$: $d(y, x) < \varepsilon$, $g_i(y) < b_i$, $\forall\ i \in \mathbf{I}\}$ where $d(y, x)$ is the Euclidean distance between points y and x. As $h(.)$ is a convex increasing function, $h(\tilde{x}) > h(x)$ for any $\tilde{x} = x + \varepsilon \bigtriangledown h(x)$, where $\bigtriangledown$ is gradient vector generator. This means for any interior solution of the problem, there is always another solution which makes the objective function bigger, i.e. there is no optimum interior solution of the problem.

Statement b: We write Karush-Kuhn-Tucker conditions for the problem with two products. Following results can easily be extended for m products.

$$\beta_1 P_1 e^{\beta_0+\beta_1 f_1+\beta_2 s_1} - \mu w_1 + \lambda_1 \eta_1 - \lambda_1 \beta_1 e^{\beta_0+\beta_1 f_1+\beta_2 s_1} - \alpha_1 + \gamma_1 = 0 \tag{9}$$

$$\beta_1 P_2 e^{\beta_0+\beta_1 f_2+\beta_2 s_2} - \mu w_2 + \lambda_2 \eta_2 - \lambda_2 \beta_1 e^{\beta_0+\beta_1 f_2+\beta_2 s_2} - \alpha_2 + \gamma_2 = 0 \tag{10}$$

$$\beta_2 P_1 e^{\beta_0+\beta_1 f_1+\beta_2 s_1} - \lambda_1 - \lambda_1 \beta_2 e^{\beta_0+\beta_1 f_1+\beta_2 s_1} + u_1 = 0, \tag{11}$$

$$\beta_2 P_2 e^{\beta_0+\beta_1 f_2+\beta_2 s_2} - \lambda_2 - \lambda_2 \beta_2 e^{\beta_0+\beta_1 f_2+\beta_2 s_2} + u_2 = 0, \tag{12}$$

$$\mu(f_1 w_1 + f_2 w_2 - LSL) = 0, \tag{13}$$

$$\lambda_1\left(-s_1 + \eta_1 f_1 - e^{\beta_0+\beta_1 f_1+\beta_2 s_1}\right) = 0, \tag{14}$$

$$\lambda_2\left(-s_2 + \eta_2 f_2 - e^{\beta_0+\beta_1 f_2+\beta_2 s_2}\right) = 0, \tag{15}$$

$$\alpha_1(ULF - f_1) = 0, \tag{16}$$

$$\alpha_2(ULF - f_2) = 0, \tag{17}$$

$$\gamma_1 f_1 = 0, \tag{18}$$

$$\gamma_2 f_2 = 0, \tag{19}$$

$$u_1 s_1 = 0, \tag{20}$$

$$u_2 s_2 = 0, \tag{21}$$

$$\mu,\ \lambda_1,\ \lambda_2,\ \alpha_1,\ \alpha_2,\ \gamma_1,\ \gamma_2,\ u_1,\ u_2 \geq 0, \tag{22}$$

where $\eta_i = SD/l_i$ for $i = 1, 2$. It is easy to see that $\lambda_1 = 0$ contradicts with the positivity of u_1 due to Equation (11). This also holds for $\lambda_2 = 0$ with Equation (12).

Statement *c*: Suppose $\gamma_1 > 0$. Given that statement *b* holds, $s_1 < 0$ which is contradiction to primal feasibility due to Equation (11).□

The first statement of Theorem 1 implies that at least one of the constraints should be binding in the optimal solution of the problem. This result together with the convexity of the objective function implies that the problem can also be solved with gradient search modified with constraints.

The second result of the theorem shows the criticality of the constraint (8e) which can be used to calculate remaining inventory (s_j) when a product is assigned to its upper bound (ULF) in the optimal solution as in the following corollary given without Proof.

Corollary 1. *If optimal facing of a product is in* $[0, ULF]$, $f_i^* \in [0, ULF]$, *then* s_i *satisfies the following equation, which has a unique solution.*

$$ln((SD/l_i)f_i^* - s_i) - \beta_0 - \beta_1 f_i^* - \beta_2 s_i = 0. \tag{23}$$

The last statement of the Theorem shows that the problem size can be relatively reduced by dropping the constraint (8f).

Due to convexity of the objective function and the complex structure of the optimization problem, further analysis on optimality conditions only leads intricate conditions. Therefore, we proceed to the application phase in which the empirical model is run with the mathematical model described above.

4.2.2. Discretization scheme based on NLP relaxation

The mathematical model in (8a-8e) is a nonlinear programming model. As described above, its optimal solution consists of facings being equal to ULF for n products, facing equal to a float number in $[0, ULF]$ and facings being equal to values near zero for $m - n - 1$ products due to the statement *c* of Theorem 1. From practical point of view, it seems straightforward to round facings of $m - n - 1$ products down to zero. However, that would leave some unassigned shelf space.

To remedy this, we suggest a discretization scheme which empty some of the previously assigned shelf space (by NLP) and reassign it iteratively in a greedy fashion. Define $J_{UB} \subseteq J$ as the subset of products of which facings are assigned to ULF in the NLP model. Also define remaining shelf space, $lspc{:}=LSL - \sum_{i\in J} w_j f_j$, after facings are assigned.

The algorithm starts with the facings of the NLP solution, denoted with $f_j^{(0)},\ \forall j \in J$. At the beginning of the algorithm facings, which are equal to ULF, are set to $\lfloor ULF/2 \rfloor$ in order to empty some shelf space.

$$f_j^{(0)} = \lfloor ULF/2 \rfloor,\ \forall j \in J_{UB},$$

$$lspc^{(0)} = LSL - \textstyle\sum_{j\in J} w_j f_j^{(0)}.$$

At the beginning of each iteration, the algorithm rounds up all facings $f_j^+ = \lceil f_j^{(i-1)} \rceil,\ \forall j \in J$ and calculates d_j^+ using Equation (23). Then, additional demand per unit shelf space for all products ($z_j{:}=(d_j^+ - d_j^{(i-1)})/w_j,\ \forall j \in J$) is calculated and the facing of the product with maximum z_j is rounded up.

$$f_k^{(i)} = f_k^+,$$

for $k = argmax_{j\in J}\{z_j\}$ and $f_j^{(i)} = f_j^{(i-1)},\ j \in J,\ j \neq k$. This process continues until all shelf space is assigned to all products ($lspc^{(i)} \leq 0$). The heuristic algorithm is formally articulated in Algorithm 1 whereas its performance against the MINLP solution of the problem is presented in the next section.

Algorithm 1
Greedy Heuristic for Discretization of Facings

```
 1: $f_j^{(0)} = \lfloor ULF/2 \rfloor, \forall j \in J_{UB}$.
 2: $lspc^{(0)} = LSL - \sum_{j\in J} w_j f_j^{(0)}$.
 3: Calculate $d_j^{(0)}$ for all $j \in J$ using Equation 23.
 4: Set $i = 1$.
 5: while $lspc^{(i-1)} > 0$ do
 6:     $f_j^+ = \lceil f_j^{(i-1)} \rceil$ for all $j \in J$.
 7:     Calculate $d_j^+$ for all $j \in J$ using Equation 23.
 8:     $k = \arg\max_{j\in J}\{(d_j^+ - d_j^{(i-1)})pr_j/w_j\}$
 9:     $f_k^{(i)} = \lceil f_k^+ \rceil$, and $f_j^{(i)} = f_j^{(i-1)}$ for all $j \in J, j \neq k$.
10:     $d_k^{(i)} = d_k^+$, and $d_j^{(i)} = d_j^{(i-1)}$ for all $j \in J, j \neq k$.
11:     $lspc^{(i)} = LSL - \sum_{j\in J} w_j f_j^{(i)}$
12:     $i = i + 1$
13: end while
```

4.2.3. Computational complexity of the greedy heuristic

The worst-case time complexity of the greedy algorithm is presented in the following theorem.

Theorem 2. *Let n be the number products in the candidate set J. Then the worst case time complexity of Greedy Heuristic (GH) is* $\mathcal{O}(n^2)$.

Proof. *Define ULF to be the upperlimit of facing. The algorithm starts with an initial solution. It has to solve* Equation (23) *to calculate the objective function value of this solution. Bisecton Search (BS) algorithm is used for this purpose. BS needs at worst* $\log(ULF/\varepsilon)$ *steps to terminate, where* ε *is the tolerance. Then, the initilazation of GH takes* $\log(ULF/\varepsilon)$ *steps. In the main loop, GH adds a new item to the facings or increase the facing of a particular item at each iteration to create a new facings. After creating a new facings, GH calls BS to calculate the objective function value. Assume that* $J_{UB} = \emptyset$, *and* $n\max_{j\in J} l_j < LSL$ *i.e. the shelf length is wide enough to store all items at their upper limit at the same time. Then the main loop needs at worst* nU *steps to terminate. At each iteration, step 7 requires* $c_1\log(ULF/\varepsilon)$ *steps whereas others (step 6, 8, 9, 10, 11) take* cn *steps. Step 12 takes* c_2 *amount of time. The total amount of steps to reach the final step is*

$$c_1\log(ULF/\varepsilon) + nULF(cn + c_2).$$

Hence the wost-case time complexity of the GH is $\mathcal{O}(n^2)$.□

In this study, our greedy heuristic is compared with two well-known method from the literature: Genetic Algorithm (GA) and Simulated Annealing (SA). In the following theorems we present worst-case performance results for our implementation of these methods to the assortment planning problem. The proofs of theorems are presented in the appendix of this paper.

Theorem 3. *Let n be the number of items. If the population size, p, and the maximum number of iterations, N, are linearly proportional to the number of items, i.e.* $p = \zeta n$, $N = \zeta_2 n$, *then the worst case time complexity of GA implementation is* $\mathcal{O}(n^4)$.

Theorem 4. *Let n be the number of items. If the maximum number of iterations is propotional to the number of items, i.e.* $NIter = \zeta n$, *then the worst case time complexity of Simulated Annealing (SA) implementation is* $\mathcal{O}(n^3)$.

As can be seen from these results, our heuristic has much better performance than general optimization heuristics from the literature. The main difference stems from the fact that our heuristic utilizes

Table 6
Test bed for the numerical experiment.

Scenarios		
Number of Products	LSL	ULF
50	250	3
	500	4
	750	5
500	2500	6
	5000	7
	7500	
5004	12500	
	25000	
	37500	

Equation (23) for reaching a better solution set whereas other methods random perturbations (SA) and various crossovers between solutions (GA) in previous iterations.

Importantly, the result in Theorem 2 is the worst-case performance of GH and it is derived from a (worst-case) scenario where the NLP solution given above yields a solution in which $f_j = 0$, $\forall\, j_i nJ$. In practice, on the other hand NLP solution gives a warm start to the algorithm which converges much faster than the worst-case performance. Practical performances of our heuristic and other two methods from the literature are presented in the next section.

4.3. Numerical experiments

In this section, we present results of the numerical experiments depicting the performance of the heuristic solution for different number of products and in various parameter settings. To this end we developed three scenarios with 50, 500 and 5004 products. For each product set, a respective empirical model for demand is developed. A detailed description (and summary statistics) of these empirical demand model can be found in Durmuş (2017). In each scenario three different *LSL* values, representing small, medium and large store spaces for a supermarket, are considered. For every product-LSL combination, we consider 5 different *ULF* levels. Therefore, our test bed consists of 45 different parameter combinations given in Table 6.

In each parameter combination we calculate the NLP solution using BARON solver, and then this solution is utilized in the heuristic algorithm. As a benchmark, we run BARON solver for MINLP. For the solution with 50, 500 and 5004 products, we set the solution time to 30000, 100000 and 200000 s respectively. MINLP solution performance of BARON is found to be highly sensitive to the number of products. Average optimality gaps are 21.6% and % for 50 and 500 products respectively. For 5004 products BARON failed to provide a feasible solution within 200000 s. Therefore, NLP solution of the problem is used as a benchmark for 5004 products.

To measure the performance of the heuristic solution, we compared the results of our heuristic with two different types of heuristic methods, genetic algorithm and simulated annealing, from the literature (Borin et al., 1994; Lin, 2006; Rajaram, 2001). Genetic Algorithm (GA) is a common population-based heuristic method for solutions of complex, nonlinear integer programming problems. In this study, we modified the generic GA scheme by Talbi (2009). A two-dimensional chromosome representation with 1-point crossover reproduction scheme is utilized (Anderson et al., 1991). Please refer to Appendix B for further technical details of the GA. Simulated Annealing (SA) is a simulation-based heuristic algorithm commonly used for global optimization problems. In this study we directly adapted the chromosome representation of our GA implementation for the solution representation of SA. Technical details of the SA algorithm are presented in Appendix C. In addition to those metaheuristic methods, we also consider MINLP and the upper bound of the MINLP obtained from the BARON solver.

Table 7
Percent average deviation of all heuristic methods from the MINLP solution with Baron.

	Average	Std.Dev	Max	Min
Greedy vs. Up.Bound MINLP	22.7%	25.7%	60.5%	−11.4%
GA vs. Up.Bound MINLP	51.45%	13.97%	76.01%	16.03%
SA vs. Up.Bound MINLP	31.57%	11.59%	54.40%	13.05%
MINLP Opt. Gap	35.7	20.1	65.9	0.0

Table 8
Detailed optimality gap of all heuristics.

Num. of Product	IP_Opt Gap	GA Gap	SA Gap	Greedy Gap
50	21.6%	41.29%	29.97%	22.7%
500	49.9%	52.34	29.19%	27.9%
5004	–	60.72	35.57%	30.7%

Our results indicate that the greedy heuristic yields 22.7% optimality gap whereas GA and SA solutions creates optimality gaps of 51.45% and 31.57%. Furthermore, the BARON solver's optimality gap is 35.7% on average (Table 7 and Table 8). The standard deviation of the greedy heuristic's performance is close to the other solutions of the problem.

A detailed look into the results indicates that the relative performance of the heuristic algorithm, seems insensitive to the number of products in the problem set whereas MINLP solution's optimality gap is increasing (Fig. 4). The performances of GA and SA are found to be quite sensitive to the number of products and upper limit for facing (*ULF*) (Fig. 4). Specifically, the heuristic solution is overperformed by MINLP for 50 products, whereas its performance is significantly better than MINLP for 500 products. Note that BARON couldn't yield a feasible solution after 200000 s of running. Compared to other methods, our greedy algorithm significantly leads to lower optimality gap for all product number and *ULF* combination. Simulated annealing is the closest rival of our greedy heuristic it is overperformed in all parameter combinations of our test bed.

We also compare our method with other heuristics from the perspective of CPU times in Table 9. Those results indicate that our heuristic method converges in a much shorter run-times, 31 s, whereas the other two heuristics' average run-times are 12.296 and 11.317 min. Similar to the previous statistics, our heuristic method most beneficial for large number of products. Specifically, our methods yields a solution approximately 1 min whereas other methods take more than 30 min to reach their solutions.

5. Conclusion

In retail sector, product variety is increasing significantly, while the shelf space stays almost the same. Hence, assortment planning, which is the selection of goods (out of a large candidate set) and assignment of shelf space to each product to maximize profit, gets important for retailers' financial performance. In the assortment planning problem, utilization of past point-of-sale data is important to take customers' preferences into consideration in the assortment planning problem.

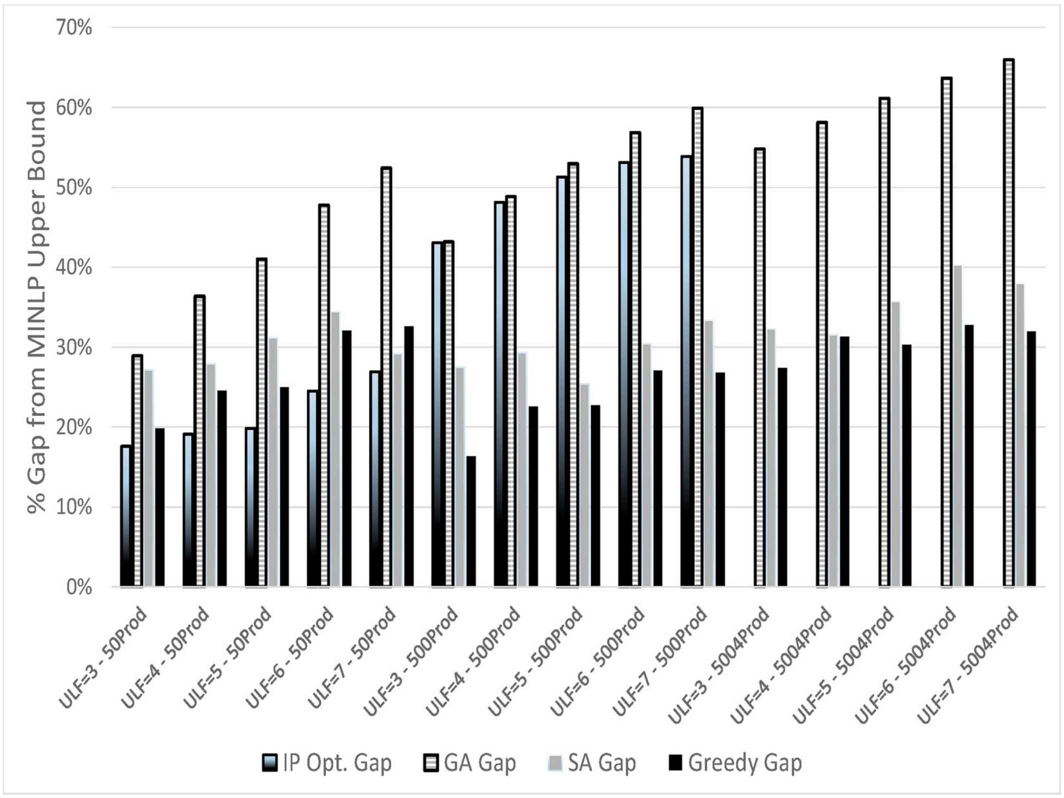

Fig. 4. Deviations for different *ULF* values.

Table 9
CPU time of heuristic algorithms (minutes).

Products	LSL	GA	SA	Greedy
50	250	1.880	0.368	0.001
50	500	1.535	0.395	0.002
50	750	0.811	0.291	0.002
500	2500	1.014	1.833	0.076
500	5000	1.246	1.330	0.129
500	7500	1.232	0.705	0.190
5004	12500	34.432	40.953	1.036
5004	25000	34.695	30.796	1.435
5004	37500	33.822	25.186	1.896
Averages		**12.296**	**11.317**	**0.530**

In this paper, we consider an assortment planning problem of a supermarket in Turkey. To this end, we built a log-linear regression model representing the relation between facing and demand using historical point-of-sale data. This model is utilized in a mixed integer nonlinear programming (MINLP) model that aims to maximize total profit under shelf-space constraints.

MINLP is a difficult-to-solve problem type even for moderately large number of products. Numerical experiments with different problem sizes reveal that the extant (state-of-the-art) solvers can only solve small-size problems. For moderately large problem size, current solvers yield very large optimality gaps even with long run-times. For the real problem size, current solvers failed to generate a feasible solution.

For the solution we develop a greedy heuristic that utilizes mathematical properties derived from relaxed version of the problem. Numerical experiments indicate that our solution is overperformed by BARON solver for only small problem size. For moderately large problem size, our heuristic is better than the MINLP solution. In order to benchmark its performance, we consider two metaheuristics that are common for solution of complex nonlinear problems. Our numerical experiments indicate that our method generates better solutions in a much shorter time periods.

Appendices

A. Business Case of a Turkish Supermarket Chain

In one of stores of a supermarket chain in Istanbul, an international brand of probiotic yoghurt and a local brand of buttermilk are being sold next to each other on a refrigerated shelf. Each product has the same package width, which we call facing. The probiotic yoghurt is sold for 4.5 TL whereas the buttermilk's price is 0.5 TL and this price (profit margin) difference leads assortment planner of the company to assign fifteen facings to the former product and four facings to the latter. On the other hand the yoghurt generates 2.3 TL daily revenue (0.5 item/day demand rate) whereas the buttermilk's revenue is 18.1 TL per day (36 item/day demand rate).

Furthermore, facings determine the amount of space assigned to each product in the refrigerator. Due to the improper shelving, the shelf space for buttermilks is much less than the shelf space of the probiotic yoghurt although demand rate of the former is 70 times higher than the latter's demand. This suboptimal shelving potentially leads lost sales for the buttermilk and high disposal rates of the yoghurt. For more details of the business case, refer Durmuş (2017).

B. A Genetic Algorithm for the Assortment Optimization Problem

We modified the generic GA scheme in (Talbi, 2009). A two-dimensional chromosome representation from (Anderson et al., 1991) is used. Each column of the chromosome refers to an item. There are two rows in the chromosome: (1) Percentage of facings in terms of UFL, hence f_j is obtained when corresponding gene value is multiplied by UFL and rounded to nearest integer less than or equal to this value, (2) Percentage of stock in terms of actual facings, hence when f_j is multiplied by a gene value from this row the corresponding s_j is obtained. Once a chromosome is on hand we can calculate sales values (s_j) as high as possible such that the constraints (8e) and (8h) are not violated. Also, we ensure that constraint (8d) is satisfied by using percentages. Constraints (8f) and (8g) are satisfied by definition. Since, random generation of the chromosomes and random crossover operations may lead to facings such that constraint (8c) is exceeded, we inherited a drop heuristic to repair such violations Li et al. (2003). The heuristic is based on discarding non-promising items from the facings one-by-one until the constraint is satisfied.

B.1. Mutation Operation (Algorithm 2)

Reproduction scheme of the population is based on 1-Point crossover and perturbation of percentages as mutation. The mutation operator changes the value of a single gene, i.e. percentage of f_j or s_j, by adding $\alpha U[-1,1]$ to the gene value where α is a constant and a real number from [0,1], and $U[\circ]$ follows the uniform distribution. If the gene value stays out of the closed interval [0,1], the value is adjusted to 0 if the value is less than 0, and 1 otherwise. The pseudocode of the mutation operation is given in Algorithm 2. The time complexity of the mutation operation is given in Lemma 1.

Lemma 1. *If the population size of the genetic algorithm is linearly propotional to the number of items, denoted by n, then the worst-case time complexity of the mutation operation is $\mathcal{O}(n^2)$.*

Proof. *In the pseudocode of the mutation operation in Algorithm 2 all steps between 4 to 21 take $\mathcal{O}(1)$ amount of time. The for loop in step 3 runs for $n+1$. Assuming the population size is linearly propotional to the the number of items n. Then the worst-case time complexity of the mutation operation is $\zeta n(n+1) = (\zeta n^2 + \zeta n)\mathcal{O}(1) \approx \mathcal{O}(n^2)$.*

B.2. CalculateSales Routine (Algorithm 5)

This algorithm takes $\mathcal{O}(n)$ amount of time as the for loop in Algorithm 5 runs for n amount of iterations with $\mathcal{O}(1)$ operations. This result is given in the following lemma without Proof.

Lemma 2. *The worst-case time complexity of CalculateSales routine is $\mathcal{O}(n)$.*

B.3. Next Population Routine (Algorithm 3)

This routine takes, the number of products (n), population size, which is assumed to be proportional to the number of items ($PopSize = \zeta n$), population matrix for facings, denoted by $\bar{F}$, population matrix for remaining inventory, denoted by $\bar{S}$, width, price and **fv** vectors ($\bar{\mathbf{w}}$, $\bar{\mathbf{p}}$ and $\overline{\mathbf{fv}}$: $=(fv_i)_{i\in I}$).

At the beginning of the procedure, we calculate each product's selection probability $Pr_i := \frac{fv_i}{\sum_{i\in I} fv_i}$ for all $i \in I$. Then we calculate the cumulative probabilities $cPr_i,\ \forall\, i \in I$. Using these cumulative probabilities, we create two candidate vector α and β. Then we crossover them in Steps 13,14 and 19 and correct these new candidate solutions with the feasibility condition 8c in Steps 15–18 and 20–23 in Algorithm 3. In these corrections, we call Drop Heuristic in Algorithm 6 with worst-case time complexity of $\mathcal{O}(n^2)$. Similar operations are executed in for $\bar{\mathbf{S}}$ in steps 26–27.

Lemma 3. *Let n be the number of items and the population size is linearly propotional to the number of items, i.e. $p = \zeta n,\ \zeta > 0$. Then, the worst case time complexity of Next Population routine (Algorithm 3) is $\mathcal{O}(n^3)$.*

Proof. *Next Population routine in Algorithm 3 starts with the summation and division operations in Step 2-3 which take $c_1 n$ amount of time. Step 4-7 take $c_2\zeta n$ amount of time. The for loop between step 8-34 runs $\frac{\zeta n}{2}$ amount of iterations. At each iteration, steps 9, 11 and 13 take c_1 amount of time in total and two maximizations in steps 10 and 12 take $c_2 n$ and $c_4 n$ amount of time units. Steps 14 and 19 copy different parts of $\bar{\mathbf{F}}_\alpha$ and $\bar{\mathbf{F}}_\beta$ to $\bar{x}$ which take $c_5 n$ and $c_6 n$ amount of time. Summations at steps 15 and 20 take $c_7 n$ and $c_8 n$. Steps 16-18 and 21-23 take $c_9 n^2$ and $c_{10} n^2$ amount of time due to Lemma 4. Steps 24-29 form vectos with a dimension n. They take $c_{11} n$ amount of time in total. Steps 30 and 32 take $c_{12} n$ amount of time due to Lemma 2. Summations in steps 31 and 33 take $c_{13} n$ amount of time. Steps 35, 36, 38, 40 and 41 take $c_{14} n$ amount of time units and sorting at steps 37 and 39 take $c_{15} n\log_2 n$. Therefore, the total amount of time units is $\frac{\zeta}{2} n[c_1 + c_{10} + (c_2 + c_4 + c_5 + c_6 + c_7 + c_8 + c_{11} + c_{12} + c_{13} n + c_{14} n)n + (c_9 + c_{10})n^2] + (c_{13} + c_{14})n + c_{15} n\log_2 n$. Hence the worst-case time comlexity is $\mathcal{O}(n^3)$.*

The genetic algorithm terminates when there is no improvement for k successive iterations or the number of iterations exceed a predefined bound N. All parameters of the algorithm is chosen by trial-and-error method. Specifically, the population size and the number of iterations are set to linearly proportional to the number of products in the assortment algorithm. The worst-case time complexity of this implementation is given in Theorem 5.

Theorem 5. *Let n be the number of items. If the population size, p, and the maximum number of iterations, N, are linearly proportional to the number of items, i.e.* $p = \zeta n$, $N = \zeta_2 n$, *then the worst case time complexity of GA implementation is* $\mathcal{O}(n^4)$.

Proof. *The optimization routine given in* Algorithm 4 *consists of two large loops at steps* 2-11 *and steps* 16-33 *and series of assignments at steps* 12-15 *which take in total* $c_1 n$ *time units. The first loop runs for* ζn *iterations. At each iteration steps* 3-5 *take* $c_2 n$. *Drop Heuristic at steps* 6-8 *and* 21-23 *take* $c_3 n^2$ *and* $c_4 n^2$. *Steps* 9 *and* 10 *take* $c_5 n$ *amount of time. The second loop runs for* $\zeta_2 n$ *iterations. At each iteration, Next Population routine* (*step* 17) *runs for* $c_6 n^3$ *amount of time due to* Lemma 3 *and Mutation Operation* (*step* 18) *takes* $c_7 n^2$ *amount of time due to* Lemma 1. *Inner for loop between steps* 19 *and* 26 *runs for* ζn *amount of iteration. At each iteration, step* 20 *takes* $c_9 n$, *steps* 21-23 *take* $c_{10} n^2$, *step* 24 *and* 25 *takes* $c_{11} n$ *amount of time. Step* 27-32 *take* $c_{12} n$ *amount of time. Hence the required amount of time to complete the algorithm is*

$$c_1 n + \zeta n(c_2 n + (c_3 + c_4)n^2 + c_5 n) + \zeta_2 n(c_6 n^3 + c_7 n^2 + \zeta n(c_{10} n^2 + (c_9 + c_{11})n) + c_{12} n).$$

Hence the worst case time complexity is $\mathcal{O}(n^4)$.

Algorithm 2
Mutation Operation of the Genetic Algorithm

```
1: Input: $PopSize, \mathbf{F}, \mathbf{S}, n, \alpha, MProb$
2: for all $1 \le i \le PopSize$ do
3:   for all $1 \le j \le n$ do
4:     $r \leftarrow U[0,1]$
5:     if $MProb \le r$ then
6:       $x \leftarrow \bar{\mathbf{F}}_{\mathbf{ij}} + \alpha * U[-1,1]$
7:       if $x \le 0$ then
8:         $\bar{\mathbf{F}}_{\mathbf{ij}} \leftarrow 0$
9:       else if $x \ge 1$ then
10:        $\bar{\mathbf{F}}_{\mathbf{ij}} \leftarrow 1$
11:      else
12:        $\bar{\mathbf{F}}_{\mathbf{ij}} \leftarrow x$
13:      end if
14:      $x \leftarrow \bar{\mathbf{S}}_{\mathbf{ij}} + \alpha * U[-1,1]$
15:      if $x \le 0$ then
16:        $\bar{\mathbf{S}}_{\mathbf{ij}} \leftarrow 0$
17:      else if $x \ge 1$ then
18:        $\bar{\mathbf{S}}_{\mathbf{ij}} \leftarrow 1$
19:      else
20:        $\bar{\mathbf{S}}_{\mathbf{ij}} \leftarrow x$
21:      end if
22:    end if
23:  end for
24: end for
25: return $\bar{\mathbf{F}}, \bar{\mathbf{S}}$
```

Algorithm 3
Next Population Routine For the Genetic Algorithm

```
1: **Input:** $PopSize, NIter, n, \bar{\mathbf{w}}, \bar{\mathbf{p}}, LSL, UFL$
2: **for all** $1 \le i \le PopSize$ **do**
3:     $\bar{\mathbf{F}}_{\mathbf{ij}}^{(\mathbf{0})} \leftarrow U[0,1], \forall j \in J$
4:     $\bar{\mathbf{S}}_{\mathbf{ij}}^{(\mathbf{0})} \leftarrow U[0,1], \forall j \in J$
5:     $lspc_i \leftarrow LSL - \sum_{j \in J} w_j \lfloor UFL * \bar{\mathbf{F}}_{\mathbf{ij}}^{(\mathbf{0})} \rfloor$
6:     **if** $lspc_i < 0$ **then**
7:         $\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{0})} \leftarrow DropHeuristic(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{0})})$
8:     **end if**
9:     $\bar{\mathbf{sales}} \leftarrow CalculateSales(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{0})}, \bar{\mathbf{S}}_{\mathbf{i}}^{(\mathbf{0})})$
10:    $fv_i \leftarrow \sum_{j \in J} p_j sales_j$
11: **end for**
12: $d \leftarrow \{i : fv_i = \max(\bar{\mathbf{fv}})\}$
13: $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{F}}_{\mathbf{d}}^{(\mathbf{0})}$
14: $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{S}}_{\mathbf{d}}^{(\mathbf{0})}$
15: $\eta \leftarrow fv_d$
16: **for all** $1 \le t \le NIter$ **do**
17:     $\bar{\mathbf{F}}^{(\mathbf{t})}, \bar{\mathbf{S}}^{(\mathbf{t})} \leftarrow NextPopulation(\bar{\mathbf{F}}^{(\mathbf{t-1})}, \bar{\mathbf{S}}^{(\mathbf{t-1})})$
18:     $\bar{\mathbf{F}}^{(\mathbf{t})}, \bar{\mathbf{S}}^{(\mathbf{t})} \leftarrow Mutation(\bar{\mathbf{F}}^{(\mathbf{t})}, \bar{\mathbf{S}}^{(\mathbf{t})})$
19:     **for all** $1 \le i \le PopSize$ **do**
20:         $lspc_i \leftarrow LSL - \sum_{j \in J} w_j \lfloor UFL * \bar{\mathbf{F}}_{\mathbf{ij}}^{(\mathbf{t})} \rfloor$
21:         **if** $lspc_i < 0$ **then**
22:             $\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{t})} \leftarrow DropHeuristic(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{t})})$
23:         **end if**
24:         $\bar{\mathbf{sales}} \leftarrow CalculateSales(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{t})}, \bar{\mathbf{S}}_{\mathbf{i}}^{(\mathbf{t})})$
25:         $fv_i \leftarrow \sum_{j \in J} p_j sales_j$
26:     **end for**
27:     $d \leftarrow \{i : fv_i = \max(\bar{\mathbf{fv}})\}$
28:     **if** $fv_d > \eta$ **then**
29:         $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{F}}_{\mathbf{d}}^{(\mathbf{t})}$
30:         $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{S}}_{\mathbf{d}}^{(\mathbf{t})}$
31:         $\eta \leftarrow fv_d$
32:     **end if**
33: **end for**
34: **return** $\bar{\mathbf{f}}_{\mathbf{heur}}, \eta$
```

Algorithm 4
Genetic Algorithm for the Assortment Problem

```
1: Input: $PopSize, NIter, n, \bar{\mathbf{w}}, \bar{\mathbf{p}}, LSL, UFL$
2: for all $1 \leq i \leq PopSize$ do
3:     $\bar{\mathbf{F}}_{\mathbf{ij}}^{(\mathbf{0})} \leftarrow U[0,1], \forall j \in J$
4:     $\bar{\mathbf{S}}_{\mathbf{ij}}^{(\mathbf{0})} \leftarrow U[0,1], \forall j \in J$
5:     $lspc_i \leftarrow LSL - \sum_{j \in J} w_j \lfloor UFL * \bar{\mathbf{F}}_{\mathbf{ij}}^{(\mathbf{0})} \rfloor$
6:     if $lspc_i < 0$ then
7:         $\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{0})} \leftarrow DropHeuristic(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{0})})$
8:     end if
9:     $\bar{\mathbf{sales}} \leftarrow CalculateSales(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{0})}, \bar{\mathbf{S}}_{\mathbf{i}}^{(\mathbf{0})})$
10:    $fv_i \leftarrow \sum_{j \in J} p_j sales_j$
11: end for
12: $d \leftarrow \{i : fv_i = \max(\bar{\mathbf{fv}})\}$
13: $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{F}}_{\mathbf{d}}^{(\mathbf{0})}$
14: $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{S}}_{\mathbf{d}}^{(\mathbf{0})}$
15: $\eta \leftarrow fv_d$
16: for all $1 \leq t \leq NIter$ do
17:     $\bar{\mathbf{F}}^{(\mathbf{t})}, \bar{\mathbf{S}}^{(\mathbf{t})} \leftarrow NextPopulation(\bar{\mathbf{F}}^{(\mathbf{t-1})}, \bar{\mathbf{S}}^{(\mathbf{t-1})})$
18:     $\bar{\mathbf{F}}^{(\mathbf{t})}, \bar{\mathbf{S}}^{(\mathbf{t})} \leftarrow Mutation(\bar{\mathbf{F}}^{(\mathbf{t})}, \bar{\mathbf{S}}^{(\mathbf{t})})$
19:     for all $1 \leq i \leq PopSize$ do
20:         $lspc_i \leftarrow LSL - \sum_{j \in J} w_j \lfloor UFL * \bar{\mathbf{F}}_{\mathbf{ij}}^{(\mathbf{t})} \rfloor$
21:         if $lspc_i < 0$ then
22:             $\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{t})} \leftarrow DropHeuristic(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{t})})$
23:         end if
24:         $\bar{\mathbf{sales}} \leftarrow CalculateSales(\bar{\mathbf{F}}_{\mathbf{i}}^{(\mathbf{t})}, \bar{\mathbf{S}}_{\mathbf{i}}^{(\mathbf{t})})$
25:         $fv_i \leftarrow \sum_{j \in J} p_j sales_j$
26:     end for
27:     $d \leftarrow \{i : fv_i = \max(\bar{\mathbf{fv}})\}$
28:     if $fv_d > \eta$ then
29:         $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{F}}_{\mathbf{d}}^{(\mathbf{t})}$
30:         $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{S}}_{\mathbf{d}}^{(\mathbf{t})}$
31:         $\eta \leftarrow fv_d$
32:     end if
33: end for
34: return $\bar{\mathbf{f}}_{\mathbf{heur}}, \eta$
```

Algorithm 5
CalculateSales Routine for the Genetic Algorithm

```
1: Input: $\bar{\mathbf{pr}}, \bar{\mathbf{gr}}, \mathbf{l}, \bar{\mathbf{w}}, \bar{\mathbf{p}}, \mathbf{f}, \bar{\mathbf{s}}, \beta, SD, NItem$
2: for all $1 \leq j \leq NItem$ do
3:     $\theta_j \leftarrow \lfloor f_j / l_j * SD \rfloor$
4:     $d_j \leftarrow \exp(\beta_1 + \beta_2 gr_j + \beta_3 pr_j + \beta_4 s_j + \beta_5 f_j)$
5:     $sales_j \leftarrow \min(\theta_j - s_j, d_j)$
6: end for
7: return $\bar{\mathbf{sales}}$
```

C. Simulated Annealing Heuristic for the Assortment Planning Problem

We directly adapted the chromosome representation of our GA implementation for the solution representation of SA. Hence, all constraints are satisfied as discussed in GA by definition and with the aid of the drop heuristic. Neighborhood structure is similar to the mutation operation of GA. It is defined as replacing each value of the first row, i.e., f_j, $\forall j \in J$, with $f_j + \beta * U[-1,1]$ where β is a constant and a real number from [0,1]. Similarly, each s_j is changed with $s_j + \theta U[-1,1]$ where θ is a constant and a real number from [-1,1]. If any of the f_j or s_j stays out of the closed set $[-1,1]$ after the additions, they are adjusted to 1 if their values exceed 1, or 0 otherwise. Then, the drop heuristic takes action and repairs the solution to satisfy constraint (8c) if it is violated. The algorithm terminates when number of iterations exceeds a predetermined limit *N*. Parameters of the SA implementation is decided by trial-and-error.

In order to analyze the computational complexity of the Simulated Annealing algorithm, we first figure out the complexity of the drop heuristic (Algorithm 6) and perturb routine (Algorithm 7). Time complexity of the drop heuristic is given in the following lemma wheras the time complexity of the perturb routine, given in Lemma 5 without Proof as it can be easily seen from the pseudocode.

Lemma 4. *Let n be the number of items. The worst-case time complexity of Drop Heuristic is* $\mathcal{O}(n^2)$.

Proof. *The second step of the algorithm is the sum of the multiplication of* f_j *and* w_j. *Hence the time complexity of this operation is* $c_1 n$. *Suppose the while loop between steps* 3-7 *run for t iterations. At each iteration, the minimization at step* 4 *requires* $c_2 n$ *time unit, step* 5 *takes* c_3 *time unit, the summation of in step* 6 *requires* $c_4 n$ *time units. Hence the total time complexity of the drop heuristic is* $c1n + t((c_2 + c_4)n + c_3)$. *For the worst case analysis, suppose* $LSL = 0$. *Then the while loop has to run* f_j *times for all* $j \in J$. *The number of total iterations is* bn. *Hence the total time complexity is* $b(c_2 + c_4)n^2 + (c_3 b + c_1)n < \mathcal{O}(n^2)$.

Lemma 5. The wost-case time complexity of the perturbation routine is $\mathcal{O}(n)$.

Using Lemma 4 and 5, the computational complexity of the Simulated Annealing algorithm is presented in the following theorem.

Theorem 6. *Let n be the number of items. If the maximum number of iterations is propotional to the number of items, i.e.* $NIter = \zeta n$, *then the worst case time complexity of Simulated Annealing (SA) implementation is* $\mathcal{O}(n^3)$.

Proof. *The implementation starts with initialization phase. A random solution is created and assigned as initial solution at steps* 3-4 *which require* $c_1 n$ *amount of time units. Step* 5 *takes* $c_2 n$. *Assume that this solution is infeasible. Then, the algorithm calls Drop Heuristic* (*steps* 6-8) *which requires* $c_3 n^2$ *amount of time units* (Lemma 4). *The initial feasible solution is used to calculate sales at steps* 9-12 *which take* $c_4 n^2$ *amount of time units. Calculated profit is initialized at step* 13. *This takes* $c_5 n$. *The initialization is followed by the main loop at steps* 14-36. *The loop runs for* ζn *amount of iterations. Steps* 15 *and* 16 *take* $c_6 n$ *amount of time units to generate a solution for iteration i. The modification of this solution with the feasibility condition takes* $c_7 n^2$ (*steps* 17-19). *Sales and profit are calculated at steps* 20 *and* 21 *in* $c_8 n$ *time units. Gradient vector is calculated at steps* 22-34 *in* $c_9 n$ *in the worst case. Cooling is applied at step* 35 *in* c_{10} *amount of time. Total required amount of time of the simulated annealing is*

$$(c_1 + c_2 + c_3)n + c_4 n^2 + \zeta n(c_7 n^2 + (c_6 + c_8 + c_9)n + c_{10}).$$

Hence the worst-case time complexity of the simulated annealing algorithm is $\mathcal{O}(n^3)$.

Algorithm 6
Drop.Heur Routine for the Simulated Annealing Algorithm

```
1: Input: $\bar{\mathbf{f}}, \bar{\mathbf{w}}, \bar{\mathbf{p}}, LSL$
2: $lspc = LSL - \sum_{j \in J} w_j f_j$
3: while $lspc < 0$ do
4:     $k = \arg\min_{j \in J}\{\frac{p_j}{w_j}, f_j \neq 0\}$
5:     $f_k = f_k - 1$
6:     $lspc = LSL - \sum_{j \in J} w_j f_j$
7: end while
8: Output: $\bar{\mathbf{f}}$.
```

Algorithm 7
Perturbation Routine for the Simulated Annealing Algorithm

```
1: Input: $\bar{\mathbf{f}}, \bar{\mathbf{s}}, \beta, \theta, n$
2: for all $1 \leq j \leq n$ do
3:     $x \leftarrow f_j + \beta * U[-1, 1]$
4:     if $x \leq 0$ then
5:         $f_j \leftarrow 0$
6:     else if $x \geq 1$ then
7:         $f_j \leftarrow 1$
8:     else
9:         $f_j \leftarrow x$
10:    end if
11:    $x \leftarrow s_j + \theta * U[-1, 1]$
12:    if $x \leq 0$ then
13:        $s_j \leftarrow 0$
14:    else if $x \geq 1$ then
15:        $s_j \leftarrow 1$
16:    else
17:        $s_j \leftarrow x$
18:    end if
19: end for
20: return $\bar{\mathbf{f}}, \bar{\mathbf{s}}$
```

Algorithm 8
Simulated Annealing Algorithm for Assortment Problem

```
1: Input: $NIter, n, \bar{\mathbf{w}}, \bar{\mathbf{p}}, LSL, UFL, \theta, T_{max}$
2: $T \leftarrow T_{max}$
3: $f_j^{(0)} \leftarrow U[0,1], \forall j \in J$
4: $s_j^{(0)} \leftarrow U[0,1], \forall j \in J$
5: $lspc^{(0)} \leftarrow LSL - \sum_{j\in J} w_j \lfloor UFL * f_j^{(0)} \rfloor$
6: if $lspc^{(0)} < 0$ then
7:     $\bar{\mathbf{f}}^{(\mathbf{0})} \leftarrow DropHeuristic(\bar{\mathbf{f}}^{(\mathbf{0})})$
8: end if
9: $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{f}}^{(\mathbf{0})}$
10: $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{s}}^{(\mathbf{0})}$
11: $\bar{\mathbf{sales}} \leftarrow CalculateSales(\bar{\mathbf{f}}^{(\mathbf{0})}, \bar{\mathbf{s}}^{(\mathbf{0})})$
12: $\eta^{(0)} \leftarrow \sum_{j\in J} p_j sales_j$
13: $\eta_{heur} \leftarrow \eta^{(0)}$
14: for all $1 \le i \le NIter$ do
15:     $\bar{\mathbf{f}}^{(\mathbf{i})}, \bar{\mathbf{s}}^{(\mathbf{i})} \leftarrow Perturbation(\bar{\mathbf{f}}^{(\mathbf{i-1})}, \bar{\mathbf{s}}^{(\mathbf{i-1})})$
16:     $lspc^{(i)} \leftarrow LSL - \sum_{j\in J} w_j \lfloor UFL * f_j^{(i)} \rfloor$
17:     if $lspc^{(i)} < 0$ then
18:         $\bar{\mathbf{f}}^{(\mathbf{i})} \leftarrow DropHeuristic(\bar{\mathbf{f}}^{(\mathbf{i})})$
19:     end if
20:     $\bar{\mathbf{sales}} \leftarrow CalculateSales(\bar{\mathbf{f}}^{(\mathbf{i})}, \bar{\mathbf{s}}^{(\mathbf{i})})$
21:     $\eta^{(i)} \leftarrow \sum_{j\in J} p_j sales_j$
22:     $\Delta \leftarrow \eta^{(i)} - \eta^{(i-1)}$
23:     if $\Delta < 0$ then
24:         $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{f}}^{(\mathbf{i})}$
25:         $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{s}}^{(\mathbf{i})}$
26:         $\eta_{heur} \leftarrow \eta^{(i)}$
27:     else
28:         $r \leftarrow U[0,1]$
29:         if $r \le exp(-\Delta/T)$ then
30:             $\bar{\mathbf{f}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{f}}^{(\mathbf{i})}$
31:             $\bar{\mathbf{s}}_{\mathbf{heur}} \leftarrow \bar{\mathbf{s}}^{(\mathbf{i})}$
32:             $\eta_{heur} \leftarrow \eta^{(i)}$
33:         end if
34:     end if
35:     $T \leftarrow T * \alpha$
36: end for
37: return $\bar{\mathbf{f}}_{\mathbf{heur}}, \bar{\mathbf{s}}_{\mathbf{heur}}, \eta_{heur}$
```

Appendix D. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.jretconser.2019.04.007.

References

Alptekinoğlu, Aydın, Grasas, Alex, 2014. When to carry eccentric products? optimal retail assortment under consumer returns. Prod. Oper. Manag. 23 (5), 877–892.

Anderson, Charles A., Jones, Kathryn F., Ryan, Jennifer, 1991. A two-dimensional genetic algorithm for the ising problem. Complex Syst. 5 (3), 327–334.

Aydin, Goker, Heese, H Sebastian, 2014. Bargaining for an assortment. Manag. Sci. 61 (3), 542–559.

Borin, Norm, Farris, Paul W., Freeland, James R., 1994. A model for determining retail product category assortment and shelf space allocation. Decis. Sci. J. 25 (3), 359–384.

Cadeaux, Jack M., 1999. Category size and assortment in us macro supermarkets. Int. Rev. Retail Distrib. Consum. Res. 9 (4), 367–377.

Caro, Felipe, Gallien, Jérémie, 2012. Clearance pricing optimization for a fast-fashion retailer. Oper. Res. 60 (6), 1404–1422.

Clay, Karen, Krishnan, Ramayya, Wolff, Eric, Fernandes, Danny, 2002. Retail strategies on the web: price and non–price competition in the online book industry. J. Ind. Econ. 50 (3), 351–367.

Davis, James M., Gallego, Guillermo, Topaloglu, Huseyin, 2014. Assortment optimization under variants of the nested logit model. Oper. Res. 62 (2), 250–273.

Duan, Naihua, 1983. Smearing estimate: a nonparametric retransformation method. J. Am. Stat. Assoc. 78 (383), 605–610.

Durmuş, İpek, 2017. An assortment planning problem with a empirical demand model. Master's thesis. Işık Üniversitesi.

Federgruen, Awi, Hu, Ming, 2015. Multi-product price and assortment competition. Oper. Res. 63 (3), 572–584.

Feldman, Jacob B., Topaloglu, Huseyin, 2015. Capacity constraints across nests in assortment optimization under the nested logit model. Oper. Res. 63 (4), 812–822.

Fisher, Marshall, Vaidyanathan, Ramnath, 2014. A demand estimation procedure for retail assortment optimization with results from implementations. Manag. Sci. 60

(10), 2401–2415.
Frontoni, Emanuele, Marinelli, Fabrizio, Rosetti, Roberto, Zingaretti, Primo, 2017. Shelf space re-allocation for out of stock reduction. Comput. Ind. Eng. 106, 32–40.
Fushiki, Tadayoshi, 2011. Estimation of prediction error by using k-fold cross-validation. Stat. Comput. 21 (2), 137–146.
Gilland, Wendell G., Heese, H Sebastian, 2013. Sequence matters: shelf-space allocation under dynamic customer-driven substitution. Prod. Oper. Manag. 22 (4), 875–887.
Golrezaei, Negin, Hamid, Nazerzadeh, Rusmevichientong, Paat, 2014. Real-time optimization of personalized assortments. Manag. Sci. 60 (6), 1532–1551.
Goyal, Vineet, Levi, Retsef, Segev, Danny, 2016. Near-optimal algorithms for the assortment planning problem under dynamic substitution and stochastic demand. Oper. Res. 64 (1), 219–235.
Hand, David J., Mannila, Heikki, Smyth, Padhraic, 2001. Principles of Data Mining (Adaptive Computation and Machine Learning). MIT press, Cambridge, MA.
Hanke, J.E., Wichern, D.W., 2009. Business Forecasting, ninth ed. Pearson Education Limited.
Hekimoğlu, Mustafa, Barlas, Yaman, 2016. Sensitivity analysis for models with multiple behavior modes: a method based on behavior pattern measures. Syst. Dynam. Rev. 32 (3–4), 332–362.
Honhon, Dorothée, Gaur, Vishal, Seshadri, Sridhar, 2010. Assortment planning and inventory decisions under stockout-based substitution. Oper. Res. 58 (5), 1364–1379.
Honhon, Dorothee, Jonnalagedda, Sreelata, Pan, Xiajun Amy, 2012. Optimal algorithms for assortment selection under ranking-based consumer choice models. Manuf. Serv. Oper. Manag. 14 (2), 279–289.
Hopp, Wallace J., Xu, Xiaowei, 2008. A static approximation for dynamic demand substitution with applications in a competitive market. Oper. Res. 56 (3), 630–645.
Hübner, Alexander H., Kuhn, Heinrich, Sternbeck, Michael G., 2013. Demand and supply chain planning in grocery retail: an operations planning framework. Int. J. Retail Distrib. Manag. 41 (7), 512–530.
Hwang, Hark, Choi, Bum, Lee, Grimi, 2009. A genetic algorithm approach to an integrated problem of shelf space design and item allocation. Comput. Ind. Eng. 56 (3), 809–820.
Kök, A Gürhan, Fisher, Marshall L., 2007. Demand estimation and assortment optimization under substitution: methodology and application. Oper. Res. 55 (6), 1001–1021.
Kök, A Gürhan, Fisher, Marshall L., Vaidyanathan, Ramnath, 2008. Assortment Planning: Review of Literature and Industry practice. Retail Supply Chain Management. Springer, pp. 99–153.
Kök, A Gürhan, Xu, Yi, 2011. Optimal and competitive assortments with endogenous pricing under hierarchical consumer choice models. Manag. Sci. 57 (9), 1546–1563.
Li, Guang, Rusmevichientong, Paat, Topaloglu, Huseyin, 2015. The d-level nested logit model: assortment and price optimization problems. Oper. Res. 63 (2), 325–342.
Li, Ken-Li, Dai, Guang-Ming, Li, Qing-hua, 2003. A genetic algorithm for the unbounded knapsack problem. Machine Learning and Cybernetics. In: *2003* International Conference on, vol. 3. IEEE, pp. 1586–1590.
Lin, Chang-Chun, 2006. A genetic algorithm for solving the two-dimensional assortment problem. Comput. Ind. Eng. 50 (1–2), 175–184.
Mantrala, Murali K., Levy, Michael, Kahn, Barbara E., Fox, Edward J., Gaidarev, Peter, Dankworth, Bill, Shah, Denish, 2009. Why is assortment planning so difficult for retailers? a framework and research agenda. J. Retail. 85 (1), 71–83.
Önal, Mehmet, Arda, Yenipazarli, Kundakcioglu, O Erhun, 2016. A mathematical model for perishable products with price-and displayed-stock-dependent demand. Comput. Ind. Eng. 102, 246–258.
Quelch, John A., Kenny, David, 1994. Extend profits, not product lines. Make Sure AllYour Products Are Profitable 14.
Rajaram, Kumar, 2001. Assortment planning in fashion retailing: methodology, application and analysis. Eur. J. Oper. Res. 129 (1), 186–208.
Rusmevichientong, Paat, Shen, Zuo-Jun Max, Shmoys, David B., 2010. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. Oper. Res. 58 (6), 1666–1680.
Rusmevichientong, Paat, Shmoys, David, Tong, Chaoxu, Topaloglu, Huseyin, 2014. Assortment optimization under the multinomial logit model with random choice parameters. Prod. Oper. Manag. 23 (11), 2023–2039.
Talbi, El-Ghazali, 2009. Metaheuristics: from Design to Implementation, vol. 74 John Wiley & Sons.
Talluri, Kalyan T., Van Ryzin, Garrett J., 2006. The Theory and Practice of Revenue Management. Springer Science & Business Media.
Ulu, Canan, Honhon, Dorothée, Alptekinoğlu, Aydın, 2012. Learning consumer tastes through dynamic assortments. Oper. Res. 60 (4), 833–849.