Heuristics for the Assortment Planning Problem under Ranking-based Consumer Choice Models

We model a retailer's assortment planning problem under a ranking-based choice model of consumer preferences. Under this consumer choice model each customer belongs to a type, where a type is a ranking of the potential products by the order of preference, and the customer purchases his highest ranked product (if any) offered in the assortment. In our model we consider products with different price/cost parameters, we assume that the retailer incurs a fixed carrying cost per product offered, a substitution penalty for each customer who does not purchase his first choice and a lost sale penalty cost for each customer who leaves the store empty-handed. In the absence of any restrictions on the consumer types, searching for the optimal assortment using enumeration or integer programming is not practically feasible. The optimal assortment has very little structure so that simple greedy-type heuristics often fail to find the optimal assortment and have very poor worst case bounds. We develop an effective algorithm, called the *In-Out Algorithm*, which always provides an optimal solution and show numerically that it is very fast, e.g., more than 10,000 times faster than enumeration on a problem with 20 products.

Key words: Retailing, Assortment, Choice models, Heuristics

1. Introduction

In this research, we consider a stylized choice model in which consumers preferences are defined based on their ranking of products that could potentially be included in the assortment. We study a retailer's product selection process under this ranking-based consumer choice Model (RBM) in the presence of fixed, substitution, and lost sale penalty costs, investigate the existence of a simple structure for the optimal assortment, examine the performance of greedy-type heuristics, and propose an effective algorithm to solve the problem optimally.

Under the RBM enumerating all possible assortments or using integer programming are practically infeasible methods when the number of potential products to choose from is large. Because the problem is NP-hard, an optimal assortment cannot generally be obtained by ranking products in terms of popularity or profitability. We further show that it is not possible to simplify the search for an optimal assortment by eliminating dominated products. We also show that the optimality gap of greedy-type heuristics can be as large as 100% in our problem which, unlike previous work, includes fixed, substitution and lost sale penalty costs.

We develop a novel algorithm called the '*In-Out Algorithm*' which always obtains an optimal assortment. This algorithm rests on an iterative procedure which at each step includes/excludes

products that improve/worsen the profitability of every assortment. Despite having a complexity equal to that of the enumeration method, we show via an extensive numerical analysis that in practice, the *In-Out algorithm* is very quick. For instance, with 20 products, this algorithm is on average more than 10,000 times faster than the enumeration method and provides a solution in less than 4 minutes for product categories with up to 25 products.

Apart from the RBM, researchers in the past have adopted a variety of consumer choice models to understand structural properties of optimal assortments. For example, under the Multinomial Logit model, van Ryzin and Mahajan (1999) show that, when the prices of the products are the same, the optimal assortment is a popular set, i.e., a set with a certain number of the most popular products. Li (2007) extends their results to products with different prices and costs parameters and finds that the optimal solution is a profitable set, i.e, a set with a certain number of the most profitable products (when store traffic is continuous). Gaur and Honhon (2006) study the assortment problem under the locational choice model with a unimodal distribution of preferences, and find that the coverage intervals of the products must be adjacent. In contrast to the above models, simple structural properties are hard to establish under the RBM.

Despite the lack of structural results the RBM is an appealing way to model consumer choice. It is a very general model in the sense that with certain restrictions on the parameters of the model we can recover the Multinomial Logit, the locational choice model, the downward substitution model (see Honhon et al. (2012b) for examples). In addition, the RBM is an intuitive representation of consumers' choice making process. A number of researchers have used this model to obtain insights from theoretical work and apply it in a practical setting. For example, Mahajan and van Ryzin (2001b) study inventory competition between multiple firms offering substitutes, Mahajan and van Ryzin (2001a) develop a simulation-based method to search for the optimal assortment under stock-out based substitution, Smith et al. (2009) solve a product introduction problem under a Stackelberg game, and Honhon et al. (2012a) consider the assortment planning problem under stock-out based substitution with fixed proportions of customers of each type. Honhon et al. (2012b) consider four practically motivated special cases of the RBM, namely, the one-way substitution, location, outtree and intree preference models, and obtain efficient methods to get the optimal assortment in the same setting as ours. In contrast to the above, we do not put any restriction on the RBM and devleop a general purpose algorithm which can be used to obtain the optimal solution when the number of products to choose from is not too large (less than 30).

To offer empirical justification for the use of a RBM we refer to Farias et al. (2011) where the authors use a non-parametric method to estimate the RBM from limited data, such as sales data or

conjoint survey data. We also refer to Yunes et al. (2007), which illustrates an application of RBM model to streamline the originally vast product lines at Deere & Company. At Deere & Company, Yunes et al. (2007) construct the ranking based model, which is called customer migration model, based on customer segment information and utilities obtained from conjoint studies. In general if conjoint survey data is available, as in Deere & Company, it is relatively straightforward to come up with consumer types and their ranking lists based on the part-worth utilities calculated from the survey (see Green and Krieger (1987) for an illustration).

The performance and computational efficiency of simple heuristics has been explored to a great extent in the product selection and assortment planning literature. For example, Green and Krieger (1985) study the performance of the greedy heuristic; Dobson and Kalish (1988) develop a two-stage heuristic where products are assigned to consumer segments in stage 1 then re-assigned to increase profits in stage 2, and McBride and Zufryden (1988) formulate the product selection problem as a binary integer programming model which solves certain practically relevant instances of the product selection problem to optimality in a computationally efficient manner. The setting in which a manufacturer chooses the set of products to offer for sale has been traditionally known as product line design problem and conjoint survey data has been shown to be useful in solving for a manufacturer's optimal product line design; for example, Kohli and Sukumar (1990) propose a dynamic programming-heuristic using attributes as stages and attribute levels as states. Dobson and Kalish (1993) develop two new greedy-type heuristics, and Nair et al. (1995) propose heuristics based on a beam search approach. See also Balakrishnan and Jacob (1996), Alexouda and Paparrizos (2001), Steiner and Hruschka (2003), Balakrishnan et al. (2004) on the use of genetic algorithms for this problem. Belloni et al. (2008) develop a Lagrangian relaxation method for both the product selection problem and product line design problem and uses it to compare the performance of a number of heuristics.

While there are parallels between the product line design problem and assortment planning problem, our contribution is most relevant in the context of the retail assortment selection problem. Our work differs from the above in the explicit consideration of the impact of fixed, substitution, and lost sale penalty costs on the performance of the heuristics that are used to obtain retail assortments. We show that simple heuristics can perform arbitrarily bad and propose a novel algorithm (the '*In-Out Algorithm*'), which always provides the optimal assortment and is in practice very fast. We also study the impact of a misspecification in the model on the performance of the chosen solution. We find that the optimal assortment is very sensitive to the consumer choice model, profit margins and fixed cost.

In order to obtain our results we make an important and reasonable assumption in our research: that selling prices are fixed. It is not uncommon for retailers to adopt an MSRP (manufacturer's suggested retail price) price strategy; evidence from the data collected by Carlton and Chevalier (2001) based on the fragrance product market shows that a number of stores (namely upscale beauty and department stores) charge MSRP for the products they sell. While contractually adhering to MSRP is rare, supplying exclusive products to retailers who adopt an across-the-board no-discounting policy is not unheard of (see Carlton and Chevalier (2001)). Literature in the operations management area also lends support to the assumption of exogenous prices, see, for example, van Ryzin and Mahajan (1999), Smith and Agrawal (2000), and Gaur and Honhon (2006).

The rest of the paper is organized as follows. We present our model in §2 and then describe the optimization problem in §3. In §4 we discuss four heuristics. We develop the *In-Out algorithm* in §5 followed by our numerical study to demonstrate the effectiveness of the algorithm in §6. §8 concludes our work.

2. Model

We use bold characters to represent vectors and subscripts to denote their components, e.g., τ_j is the j-th component of vector $\boldsymbol{\tau}$. Sets and matrices are denoted by capital letters. $|S|$ denotes the cardinality of set S.

We begin by describing how consumers make choices. Consider a product category consisting of n potential products, indexed 1 to n, where $\mathcal{N} = \{1, \ldots, n\}$ and let 0 denote the no-purchase option. In order to define the purchase behavior of consumers we define a *consumer type*, which is a vector of products that the customer is willing to purchase, arranged in decreasing order of preference. For example, a customer of type $(1, 2, 4)$ has product 1 as his first choice, product 2 as his second choice, product 4 as his third choice, and he never purchases products that do not belong in his type (3 and 5 to n). In general, a type $\boldsymbol{\tau}$ is a vector $(\tau_1, ..., \tau_t)$ of product indices such that $\{\tau_1, ..., \tau_t\} \subseteq \mathcal{N}$ and $0 \leq t \leq n$. The 'null' type is such that $t = 0$ and corresponds to customers who never purchase a product from the product category.[1] Let $\mathcal{T}$ be the set of all possible types, i.e., $\mathcal{T} = \{\boldsymbol{\tau} = (\tau_1, ..., \tau_t) : 0 \leq t \leq n, \tau_t \in \mathcal{N} \text{ and } \tau_1 \neq \tau_2 \neq .... \neq \tau_t\}$. In addition, let $\alpha_{\boldsymbol{\tau}}$ be the proportion of customers of type $\boldsymbol{\tau} \in \mathcal{T}$ in the customer population, such that $\sum_{\boldsymbol{\tau} \in \mathcal{T}} \alpha_{\boldsymbol{\tau}} = 1$, and $\mathcal{T}^+ = \{\tau \in \mathcal{T} : \alpha_{\boldsymbol{\tau}} > 0\}$ be the set of consumer types that exist in the population. In practice, the

[1] Note that the type of a customer can result from a utility maximization procedure as in Mahajan and van Ryzin (2001a). Let $U(x, j)$ be utility assigned by customer x to product j for $j = 1, ..., n$ and let the utility from not purchasing anything be zero, i.e., $U(x, 0) = 0$. Let $U(x, [k])$ be the k-th greatest value in $\{U(x, 0), U(x, 1), ..., U(x, n)\}$, the type of customer x is $(\tau_1, ..., \tau_t)$ if $U(x, \tau_k) = U(x, [k])$ for $k = 1, ..., t$ and $U(x, 0) = U(x, [t + 1])$.

number of types that exist in a population is clearly less than or equal to the theoretically possible number of types, i.e. $|\mathcal{T}^+| \leq |\mathcal{T}| = \sum_{j=0}^{n} C_n^j j! = \sum_{j=0}^{n} \frac{n!}{(n-j)!}$. For example, for $n = 10$, there is a total of 9,864,100 possible types!

We assume consumer-driven substitution, that is, customers decide which products they want to buy from the set of available products; in contrast, under firm-driven substitution the firm allocates products to customers based on the customers' preferences. Consider an assortment of products $S \subseteq \mathcal{N}$ and product $j \in S$. A customer of type $\boldsymbol{\tau}$ for whom product j is the k-th choice, i.e., such that $\tau_k = j$, picks product j for set S if and only if products $\tau_1, ..., \tau_{k-1}$ do not belong to set S. Let $P_j^k(S)$ denote the proportion of customers who pick product j from assortment S and for whom product j is the k-th choice. We have:

$$P_j^k(S) = \begin{cases} \sum_{\substack{\boldsymbol{\tau}=(\tau_1,...,\tau_t): t \geq k, \tau_k = j, \\ \tau_1,...,\tau_{k-1} \notin S}} \alpha_{(\tau_1,...,\tau_t)} & \text{if } j \in S \\ 0 & \text{otw.} \end{cases} \tag{1}$$

When $k > n - |S| + 1$, we have $P_j^k(S) = 0$ as one of the first $k-1$ choices of every customer must be offered in S. Let $P_0(S) = 1 - \sum_{j \in S} \sum_{k=1}^{n-|S|+1} P_j^k(S)$ denote the proportion of customers who do not pick anything from set S.

Let π_j denote the profit margin on a product j. Retailers generally incur a *fixed cost*, denoted by the parameter K, of carrying a product in their assortment. In addition, we assume that the retailer bears a *substitution penalty cost* for every customer who purchases a product that is not his first choice, and that the penalty is higher when he purchases a lower ranked product. More specifically, if a customer buys his k-th favorite product, the penalty is $f(k)$ where f is an non-decreasing function, e.g., $f(k) = b(k-1)$ where $b \geq 0$.[2] We do not make any assumption about $f(k)$, so it is possible that $\pi_j - f(k) < 0$ for some $j, k \in \{1, ..., n\}$, meaning that the retailer may lose money when a customer buys product j as a k-th choice. Also let p denote a *lost sale penalty cost* which measures the disutility experienced by a customer who leaves the store empty-handed. We do not make any assumption regarding the relative values of $f(k)$ and p, however, one might argue that $p \geq f(n)$ makes sense for most product categories, i.e., the penalty for not satisfying a customer is larger than the penalty for making him substitute to his least favorite product.

The retailer operates in a make-to-order setting, that is, she does not carry inventory of the products. Without loss of generality we assume that mean demand is equal to 1 in the product category. Given the consumers' choice structure and the price and costs parameters of the products, the retailer's problem of finding the optimal assortment S^* can be represented as follows:

$$\Pi(S^*) = \max_{S \subseteq \mathcal{N}} \Pi(S).$$

[2] Note that it makes sense to have $f(1) = 0$ but we do not require this condition to hold.

where $\Pi(S)$, the profit associated with the assortment S, is given by

$$\begin{aligned}\Pi(S) &= \sum_{j\in S}\sum_{k=1}^{n-|S|+1} P_j^k(S)[\pi_j - f(k)] - P_0(S)p - K|S| \\ &= \sum_{j\in S}\sum_{k=1}^{n-|S|+1} P_j^k(S)[\pi_j + p - f(k)] - p - K|S|. \end{aligned} \tag{2}$$

Note that our results also apply to a make-to-stock setting with assortment-based substitution when store traffic is continuous (see Honhon et al. (2012b) for a proof of the equivalence of the two settings).

Notation	Definition
n	number of potential products
$\mathcal{N}$	set of potential products
$\boldsymbol{\tau}$	generic consumer type
$\mathcal{T}$	set of possible types
$\mathcal{T}^+$	set of existing types
S	generic assortment
π_j	product j profit margin
$f(\cdot)$	substitution penalty function
p	lost sale penalty cost
K	fixed cost
P_j^k	proportion of customers buying j as their k-th choice
P_0	proportion of customers buying nothing
Π	profit

Table 1 Notation

3. Optimization

In theory, it is always possible to find an optimal assortment by enumeration, that is, by computing the profit associated with each of the 2^n possible sets $S \subseteq \mathcal{N}$ and looking for the set with the highest profit value. However, this solution method is very time-consuming for large n. Honhon et al. (2012b) show that the complexity of the algorithm for computing the purchasing proportions $P_j^k(S)$ for $j,k = 1,...,n$ for a given assortment S is $O(|\mathcal{T}^+|n)$. Hence, the complexity of the enumeration method is $(|\mathcal{T}^+|n2^n)$. Table 11 shows the time it takes to solve a (randomly generated) numerical problem using the enumeration method as a function of n and $|\mathcal{T}^+|$. For $n = |\mathcal{T}^+| = 20$ it takes on average more than 24 hours to solve.

Another approach is to formulate the retailer's optimization problem as an integer programming model. Let x_j for $j = 1,...,n$ be binary variables that are equal to 1 if product j is included in the assortment and 0 otherwise. For all $\tau = (\tau_1,...,\tau_t) \in \mathcal{T}$ and $i = 2,...,t$, let $y_{\tau\tau_i}$ be binary variables that are equal to 1 if customers of type τ buy product τ_i (their i-th choice) from the assortment and 0 otherwise.[3] The retailer's optimization problem can be formulated as follows.

[3] Note that we do not require a binary variable for the case where a customer buys his first choice product, since making a product available, that is setting $x_j = 1$, implies that the customer for whom it is the first choice buys it.

$$\max \sum_{\tau=(\tau_1,...,\tau_t)\in\mathcal{T}} \left[(\pi_{\tau_1} - f(1) + p)x_{\tau_1} + (\pi_{\tau_2} - f(2) + p)y_{\tau\tau_2} + ... + (\pi_{\tau_t} - f(t) + p)y_{\tau\tau_t}\right]\alpha_\tau - p - \sum_{j=1}^{n} Kx_j$$

$$\text{s.t.} \quad x_{\tau_1} + \sum_{j=2}^{k} y_{\tau\tau_j} \geq x_{\tau_k} \text{ for } \tau = (\tau_1, ..., \tau_t) \in \mathcal{T}, k = 2, ..., t \tag{3}$$

$$y_{\tau\tau_j} \leq x_{\tau_j} \text{ for } \tau = (\tau_1, ..., \tau_t) \in \mathcal{T}, j = 2, ..., t \tag{4}$$

$$x_{\tau_1} + \sum_{j=2}^{t} y_{\tau\tau_j} \leq 1 \text{ for } \tau = (\tau_1, ..., \tau_t) \in \mathcal{T} \tag{5}$$

$$x_j, y_{\tau\tau_k} \in \{0, 1\} \text{ for } j = 1, ..., n, \tau = (\tau_1, ..., \tau_t) \in \mathcal{T}, k = 2, ..., t \tag{6}$$

Constraints (3) ensures that a customer gets his j-th choice product only if his 1st to $(j-1)$-th choice products are not available. Constraints (4) guarantees that a customer buys a product that is available in the assortment. Constraints (5) ensures that each customer of type $\tau = (\tau_1, ..., \tau_t)$ buys at most one product. For another mathematical programming formulation of the product selection , see for example McBride and Zufryden (1988) and Anupindi et al. (2008). This integer program has an embedded uncapacitated plant location model in it since constraints (4) to (6) have the same form as those in an uncapacitated plant location problem (see Cho et al. (1983)). Therefore, the retailer's optimization problem is NP-hard. It follows that simple sorting methods will often fail provide an optimal solution. In what follows, we illustrate this fact with some simple examples in order to provide some intuition about the problem.

It is possible to simplify the above integer programming model, when not all possible customer types exist in the population, i.e., when $\mathcal{T}^+$ is a strict subset of $\mathcal{T}$. For example, see Smith and Agrawal (2000) for an application with the one-step substitution model, wherein each consumer type does not contain more than two products. While the integer programming approach is appealing for models with very few products in each consumer type, this approach is not tractable for large values n, as the number of variables, which is equal to $n + \sum_{t=2}^{n} \frac{n!}{(n-t)!}(t-1)$, and the number of constraints, equal to $\sum_{t=2}^{n} \frac{n!}{(n-t)!}(2t-1)$, tend to grow substantially with n. For instance, if $n = 10$, then we have 78,912,820 variables and 167,689,710 constraints! In general, even for values of n as low as 4, we find that solving the integer programming problem takes much longer time than using the enumeration method described above. In practice, solving for the best assortment is generally done off-line by retailers, however prices and costs might change on a weekly basis and the number of potential products is generally greater than 20, making these two methods practically useless. Hence, we need an effective method to find an optimal solution or good performing heuristics.

The tradeoffs involved in the assortment decisions can be explained as follows. On one hand, by stocking all n products the retailer is able to capture the maximum amount of demand and avoid all substitution and lost sale penalty costs. However, fixed costs and the differences in product

profitability may deter the retailer from pursuing such a strategy. On the other hand, the retailer could offer only the most profitable product, in which case, she would save on fixed costs but she may only capture part of the demand and incur substantial substitution and lost sale penalty costs as not all consumers will purchase their most preferred product. In most cases the optimal assortment is somewhere between these two extreme solutions, as illustrated in the following example.

EXAMPLE 1. Let $n=4$. Suppose $\pi_1=8, \pi_2=7, \pi_3=6.5, \pi_4=3, \mathcal{T}^+=\{(4),(3,4),(4,3,2),(2,1,3,4)\}$ and $\alpha_{\boldsymbol{\tau}}=\frac{1}{4}$ for $\boldsymbol{\tau}\in\mathcal{T}^+$. Suppose $p=0$, $f(k)=(k-1)b$, where $b\in\{0,0.75\}$, $K\in\{0,1\}$.

The following table 2 shows the (unique) optimal assortment for all combinations of (b,K).

S^*	$K=0$	$K=1$
$b=0$	$\{1,3\}$	$\{3\}$
$b=0.75$	$\{1,3,4\}$	$\{3\}$

Table 2 Optimal assortments in Example 1.

Not surprisingly, Example 1 shows that a greater substitution penalty leads to a larger optimal assortment and a greater fixed cost leads to a smaller optimal assortment. It also demonstrates that the optimal assortment does not necessarily, i) satisfy all customers (e.g., customers of type (4) do not buy anything from S^* when $b=K=0$), ii) contain the most popular product (e.g., product 4 is not in S^* when $b=K=0$), and iii) contain the product with the highest profit margin (e.g., product 1 is not included in S^* when $b=0$ and $K=1$). Thus, Example 1 illustrates that we cannot count on simple sorting rules based on popularity or profitability (as in van Ryzin and Mahajan (1999) and Li (2007)) to obtain the optimal assortment. However, the following result when $b=K=0$ is worth noting.

PROPOSITION 1. *Let* $M=\{j\in\mathcal{N}:\pi_j=\max_{i\in\mathcal{N}}\pi_i\}$. *If* $K=0$ *and* $f(k)=0$, *there exists an optimal assortment* S^* *including the most profitable products, i.e.,* $M\subseteq S^*$.

`Proof` Suppose that S is optimal but does not contain $j\in M$. Let $P_i(S)\equiv\sum_{k=1}^{n}P_i^k(S)$. When $K=0$ and $f(k)=0$, we have

$$\begin{aligned}\Pi(S\cup\{j\})-\Pi(S) &= \sum_{i\in S\cup\{j\}}P_i(S\cup\{j\})\pi_i-\sum_{i\in S}P_i(S)\pi_i,\\ &= \left(\sum_{i\in S\cup\{j\}}P_i(S\cup\{j\})-\sum_{i\in S}P_i(S)\right)\pi_j+\sum_{i\in S}\left[P_i(S)-P_i(S\cup\{j\})\right](\pi_j-\pi_i)\end{aligned}$$

This expression is non-negative because (1) $\sum_{i\in S\cup\{j\}}P_i(S\cup\{j\})\geq\sum_{i\in S}P_i(S)$ as a customer who buys a product from S always buys a product from $S\cup\{j\}$, (2) $P_i(S)\geq P_i(S\cup\{j\})$ as the

proportion of customers buying product i cannot increase after product j is added to the set and (3) $\pi_j \geq \pi_i$ since $j \in M$. Hence, $\Pi(S \cup \{j\}) \geq \Pi(S)$. If the inequality is strict then S cannot be optimal. If not, $S \cup \{j\}$ is also an optimal solution. Q.E.D.

Proposition 1 shows that, when the fixed cost is negligible, the retailer's optimal assortment always contains the most profitable product(s). However, Example 1 shows that, even in that case, the optimal assortment does not necessarily include a certain number of the most profitable products (e.g. the optimal set $S^* = \{1,3\}$when $K = b = 0$ and product 2 has a higher profit than product 3). Further, the following example illustrates that it is also not possible to eliminate *dominated products*; a product is dominated if there exists another product that is more profitable and more preferred by all customers.

EXAMPLE 2. Let $n = 3$. Suppose $\pi_1 = 20$, $\pi_2 = 10$, $\pi_3 = 8$, $\mathcal{T}^+ = \{(2,1,3),(2,3)\}$ and $\alpha_{\boldsymbol{\tau}} = \frac{1}{2}$ for $\boldsymbol{\tau} \in \mathcal{T}^+$. Suppose $p = K = 0$ and $f(k) = 0$. The optimal assortment is $S^* = \{1,3\}$.

In the above example, every customer prefers product 2 to product 3 and product 2 is more profitable than product 3 since $\pi_2 > \pi_3$, implying that 3 is *dominated* by product 2. And yet, product 3 is included in the optimal assortment. The intuition behind is as follows: product 3 is offered alongside product 1 which is the most profitable product because it brings extra demand without cannibalizing the sales of product 1 at all. In contrast, product 2 would cannibalize the sales of product 1. Pan and Honhon (2012) prove a similar lack of dominance relationship for a choice model with vertically differentiated products when prices are exogenous.

In addition to lack of dominance, popularity or profitability based results, it is well-known that greedy-type heuristics also fail to find the optimal assortment for all problem instances, which illustrates the fact that the products interact in a complex manner in the RBM. We focus on four such heuristics in Section 4.

4. Heuristics

In this section, we present the following four heuristic approaches to the assortment planning problem: the 'Most Profitable', 'Greedy-Add', 'Greedy-Remove' and 'Largest Marginal Benefit' heuristics. The first three have been studied before in the context of a product selection problem without substitution costs (see for example Green and Krieger (1985) and Dobson and Kalish (1993)). The fourth one has, to our knowledge, not been used for this problem.

The Most Profitable (MP) heuristic adds products to the assortment in increasing order of profitability as measured by π_j. Note that Li (2007) and Liu and van Ryzin (2008) show that this algorithm is optimal when customer preferences are modeled by the Multinomial Logit model, which is a special case of the RBM.

Algorithm 1 Most profitable (MP) heuristic

Renumber products in the decreasing order of π_j such that $\pi_1 \geq \pi_2 \geq ... \geq \pi_n$.

Set $S_0 = \emptyset$ **and** $S_k = \{1, ..., k\}$ **for** $k = 1, ..., n$.

Find $S^{MP} = \arg\max_{S_k : k \in \{0,1,...,n\}} \Pi(S_k)$.

Lemma 1 shows that the MP heuristic can perform extremely bad in some cases.

LEMMA 1. *Let S^* be the optimal assortment and S^{MP} be the assortment obtained using the MP heuristic. We have $\frac{\Pi(S^{MP})}{\Pi(S^*)} \geq 0$, i.e., the achievable worst case bound is 0.*

`Proof` Consider the following example. Let $n = 2$, $\pi_1 = 10, \pi_2 = 8$, and $\alpha_{\boldsymbol{\tau}} = \frac{1}{2}$ if $\boldsymbol{\tau} \in \{(1,2),(2)\}$. Suppose $p = f(k) = 0$ and $K = 6$. We have $S^* = \{2\}$ and $\Pi(S^*) = 2$ but $S^{MP} = \emptyset$ and $\Pi(S^{MP}) = 0$. Q.E.D.

In the Greedy-Add (GA) heuristic (also called just 'greedy'), products are added to the assortment in increasing order of their contribution to profit. Note that this heuristic differs from the traditional greedy algorithm because the contribution to profit of a product depends on the set of products already in the assortment, i.e. $\Pi(S \cup \{j\}) - \Pi(S)$ is a function of S. Example 3 shows that the profit of the solutions obtained by the GA heuristic is not necessarily unimodal, which is why the heuristic always scans a total of n (non-empty) assortments.

Algorithm 2 Greedy-Add (GA) heuristic

Set: $S_0 = \emptyset$, $T = \mathcal{N}$, $k = 1$.

while $T \neq \emptyset$ **do**

  Find $j = \arg\max_{i \in T} \{\Pi(S_{k-1} \cup \{i\})\}$.

  Set $T := T \backslash \{j\}$, $S_k := S_{k-1} \cup \{j\}$, $k := k + 1$.

end while

$S^{GA} = \arg\max_{S_k : k \in \{0,1,...,n\}} \Pi(S_k)$.

EXAMPLE 3. Suppose $n = 3$, $\pi_1 = 6, \pi_2 = 20$ and $\pi_3 = 17$. $\mathcal{T}^+ = \{(1),(2,1,3),(3,1,2)\}$ and $\alpha_{\boldsymbol{\tau}} = \frac{1}{3}$ for $\boldsymbol{\tau} \in \mathcal{T}^+$. Suppose $p = K = f(k) = 0$, $k = 1, ..., n$. The GA algorithm constructs the following three sets: $S_0 = \emptyset, S_1 = \{2\}, S_2 = \{2,3\}$ and $S_3 = \{1,2,3\}$. The profit of the sets is equal to 0, 13.33, 12.33 and 14.33 respectively for S_0, S_1, S_2 and S_3 so that $S^{GA} = \{1,2,3\}$.

In Lemma 2 we obtain a worst case bound for the GA heuristic.

LEMMA 2. *Let S^* be the optimal assortment and S^{GA} be the assortment obtained using the GA heuristic. We have $\frac{\Pi(S^{GA})}{\Pi(S^*)} \geq \frac{1}{n-1}$, i.e., the achievable worst case bound is $\frac{1}{n-1}$.*

`Proof` Let S_1^* denote the assortment with the highest profit out of all the sets with cardinality 1. After one iteration, the greedy algorithm picks assortment S_1^*, therefore $\Pi(S^{GA}) \geq \Pi(S_1^*)$.

Claim: $\Pi(S^*) \leq (n-1)\Pi(S_1^*)$.

Let $S^* = \{s_1, ..., s_m\}$ with $s_1 < ... < s_m$ and $1 \leq m \leq n$. Note that if $m = n$, then $S^{GA} = S^*$ since the GA algorithm considers the set $\{1, ..., n\}$ in the last iteration. So we consider $m \leq n-1$. We have

$$\Pi(S^*) = \sum_{j=1}^{m} \sum_{k=1}^{n} P_{s_j}^k(S^*)[\pi_{s_j} + p - f(k)] - p - mK = \sum_{j=1}^{m} \Pi_{s_j}(S^*) - p,$$

where $\Pi_{s_j}(S^*) \equiv \sum_{k=1}^{n} P_{s_j}^k(S^*)[\pi_{s_j} + p - f(k)] - K$ is the contribution of product s_j to the profit of assortment S^*. It follows that $\Pi(S^*) \leq m\Pi_{s_{\hat{j}}}(S^*)$ where $\hat{j}$ is such that $\Pi_{s_{\hat{j}}}(S^*) = \max_{j=1,...,m} \Pi_{s_j}(S^*)$. Because adding products to an assortment can only decrease the proportion of customers choosing a given product as their k-choice, $P_{s_{\hat{j}}}^k(S^*) \leq P_{s_{\hat{j}}}^k(\{s_{\hat{j}}\})$ for $k = 1, ..., n$. Therefore, $\Pi_{s_{\hat{j}}}(S^*) \leq \Pi_{s_{\hat{j}}}(\{s_{\hat{j}}\})$ and $\Pi(S^*) \leq m\Pi_{s_{\hat{j}}}(\{s_{\hat{j}}\}) = m\Pi(\{s_{\hat{j}}\}) \leq m\Pi(S_1^*)$, where the last inequality is by definition of S_1^*. Since $m \leq n-1$, we have $\frac{\Pi(S^{GA})}{\Pi(S^*)} \geq \frac{\Pi(S_1^*)}{\Pi(S^*)} \geq \frac{1}{m} \geq \frac{1}{n-1}$.

Next we show that the bound is achievable with the following example. Suppose $\pi_1 = \frac{\pi}{n-1} + \epsilon$ and $\pi_2 = ... = \pi_n = \pi$, where $\pi, \epsilon > 0$. Let $\alpha_{\boldsymbol{\tau}} = \frac{1}{n-1}$ if $\boldsymbol{\tau} \in \{(1,2), (1,3), ..., (1,n)\}$. Let $p = K = f(k) = 0$. The GA heuristic gives $S^{GA} = \{1\}$ and reaches a profit of $\Pi(S^{GA}) = \frac{\pi}{n-1} + \epsilon$. However the optimal solution is $S^* = \{2, ..., n\}$ and the optimal profit is $\Pi(S^*) = \pi$. Therefore we have $\frac{\Pi(S^{GA})}{\Pi(S^*)} = \frac{\frac{\pi}{n-1}+\epsilon}{\pi}$. If we let $\epsilon \to 0$, we obtain $\frac{1}{n-1}$. Q.E.D.

Note that a similar worst-case bound is obtained by Green and Krieger (1985) for a simplified version of the profit function. This bound $\frac{1}{n-1}$ decreases with n indicating that the GA heuristic can perform very badly when n is large.

The Greedy-Remove (GR) heuristic (also called 'reverse greedy') is similar to the GA heuristic except that one starts with a full assortment then removes products in increasing order of their impact on profit.

Lemma 3 shows that the GR heuristic can also perform extremely bad.

LEMMA 3. *Let S^* be the optimal assortment and S^{GR} be the assortment obtained using the GR heuristic. We have $\frac{\Pi(S^{GR})}{\Pi(S^*)} \geq 0$, i.e., the achievable worst case bound is 0.*

`Proof` Consider the following example. Let $n = 3$, $\pi_1 = \pi_2 = 10, \pi_3 = 8$ and $\alpha_{\boldsymbol{\tau}} = \frac{1}{2}$ if $\boldsymbol{\tau} \in \{(3,1), (3,2)\}$. Suppose $p = f(k) = 0$ and $K = 6$. We have $S^* = \{3\}$ and $\Pi(S^*) = 2$ but $S^{GR} = \emptyset$ and $\Pi(S^{GR}) = 0$. Q.E.D.

The Largest Marginal Benefit (MB) heuristic is inspired by the 'largest marginal revenue' Algorithm of Talluri and Ryzin (2004) which was developed in the context of revenue management.

Algorithm 3 Greedy-Remove (GR) heuristic

Set: $T = \emptyset$, $S_0 = \mathcal{N}$, $k = 1$.

while $T \neq \mathcal{N}$ **do**

Find $j = \arg\max_{i \in S_{k-1}} \{\Pi(S_{k-1} \backslash \{i\})\}$.

Set $T := T \cup \{j\}$, $S_k := S_{k-1} \backslash \{j\}$, $k := k + 1$.

end while

$S^{GR} = \arg\max_{S_k : k \in \{0,1,\ldots,n\}} \Pi(S_k)$.

This is, to our knowledge the first application of this method to a product selection problem. The MB heuristic is similar to the GA heuristic except that products are added in increasing order of their *marginal* benefit when added to set S_k, i.e., the change in profit per extra unit of demand, $B_j(S_k) = \frac{\Pi(S_k \cup \{j\}) - \Pi(S_k)}{P_0(S_k) - P_0(S_k \cup \{j\})}$.

Algorithm 4 Largest Marginal Benefit (MB) heuristic

Set: $S_0 = \emptyset$, $T = \mathcal{N}$, $k = 1$.

while $T \neq \emptyset$ **do**

Find $j = \arg\max_{i \in T} \{B_i(S_{k-1})\}$.

Set $T := T \backslash \{j\}$, $S_k := S_{k-1} \cup \{j\}$, $k := k + 1$.

end while

$S^{MB} = \arg\max_{S_k : k \in \{0,1,\ldots,n\}} \Pi(S_k)$.

The MP heuristic has a complexity of $O(|\mathcal{T}^+| n^2 \log n)$ while the GA, GR and MB heuristics have a complexity of $O(|\mathcal{T}^+| n^3)$ since computing the purchasing proportions for a given assortment S can be done in $O(|\mathcal{T}^+| n)$ time as shown in Honhon et al. (2012b). The following example shows that these four heuristics can simultaneously fail to find the optimal assortment for a given problem.

Example 4. Let $n = 5$. Let $\pi_1 = 8, \pi_2 = 6, \pi_3 = 5, \pi_4 = 9$ and $\pi_5 = 7$. Let $b = 0$ and $K = 1$. Let $\mathcal{T}^+ = \{(2), (3), (2,1), (2,3), (3,1), (3,2), (3,4), (5,1), (5,3)\}$. The $\alpha_{\boldsymbol{\tau}}$ values are given in Table 3:

$\boldsymbol{\tau}$	(2)	(3)	(2,1)	(2,3)	(3,1)	(3,2)	(3,4)	(5,1)	(5,3)
$\alpha_{\boldsymbol{\tau}}$	1/6	2/21	1/12	1/12	1/21	2/21	2/21	1/6	1/6

Table 3 Proportions of customers of each type in Example 4.

Table 4 shows the solutions given by the four heuristics. The optimal assortment (obtained by enumeration) is $S^* = \{2, 3, 5\}$ and $\Pi(S^*) = 3$, so all of the four heuristics give a suboptimal solution.

	Assortment	Profit
MP	$\{1,2,4,5\}$	2.14
GA	$\{1,3\}$	2.92
GR	$\{2,5\}$	2.90
MB	$\{1,3\}$	2.92

Table 4 Solutions from the four heuristics in Example 3.

Given these worst-case bounds, we see that the heuristics can perform quite badly. However, it is of more practical relevance to study how they perform on more realistic examples as a function of the problem size and the fixed cost and substitution costs. We do so in Section 6.

5. The In-Out Algorithm

The *In-Out Algorithm* is an algorithm which always provides an optimal assortment. It is divided into two parts. Part I is a first sorting of the products in order to identify, in an iterative manner, the products which should *definitely* be included in the optimal assortment and the products which should *definitely not* be included in the optimal assortment. In most cases Part I ends with three groups of products: those included definitely, those not included definitely, and those for which the initial sorting in Part I was inconclusive. In Part II the algorithm focuses on this last group of products and constructs a number of candidate assortments, one of which is guaranteed to be optimal. Part I can be completed in $O(|\mathcal{T}^+|n^3)$ and does, in some cases, yield an optimal assortment directly. Part II has the same complexity as the enumeration method, however it is much faster in practice as the number of candidate assortments to compare is generally much smaller than 2^n.

The idea behind the initial sorting conducted in Part I is as follows. A product should *definitely* be in the optimal assortment if it increases profit when added to every possible assortment. A product should *definitely not* be included in the optimal assortment if it decreases profit when added to every possible assortment. Let $C_j(S) = \Pi(S \cup \{j\}) - \Pi(S)$ be the change in profit that results from adding product j to set S. Product j should be included in the optimal assortment if $\min_{S \subseteq \mathcal{N}} C_j(S) \geq 0$. Let I be the set of product which satisfy this condition. On the other hand, product j should not be included in the optimal assortment if $\max_{S \subseteq \mathcal{N}} C_j(S) \leq 0$. Let O be the set of products which satisfy this condition. After some products are added to sets I and O, one can revisit the remaining products in $\mathcal{N} \backslash (I \cup O)$: if $\min_{S: I \subseteq S \subseteq (\mathcal{N} \backslash O)} C_j(S) \geq 0$ then product j should be added to set I and if $\max_{S: I \subseteq S \subseteq (\mathcal{N} \backslash O)} C_j(S) \leq 0$, product j should be added to set O. Hence sets I and O can be calculated iteratively.

The problem with this method is that the calculation of $\min_{S: I \subseteq S \subseteq (\mathcal{N} \backslash O)} C_j(S)$ and $\max_{S: I \subseteq S \subseteq (\mathcal{N} \backslash O)} C_j(S)$ has the same complexity as the enumeration method, i.e., $O(|\mathcal{T}^+|n2^n)$.

Therefore we calculate respectively a lower and an upper bound on these two values which can be calculated in $O(|\mathcal{T}^+|n)$ for given I and O sets. For this, we define $g_1(\boldsymbol{\tau}, S)$ to be a function that returns the product chosen by a customer of type $\boldsymbol{\tau}$ from set S and let $g_2(\boldsymbol{\tau}, S)$ be a function that returns which choice the customer purchases this product as. For example, if $\boldsymbol{\tau} = (3,1)$ and $S = \{1,2\}$, we have $g_1(\boldsymbol{\tau}, S) = 1$ and $g_2(\boldsymbol{\tau}, S) = 2$ because a customer of type $(3,1)$ picks product 1 from set $\{1,2\}$ and it is his second choice. The two functions return zero if consumers of type $\boldsymbol{\tau}$ do not pick any product from set S. Using these functions we rewrite the profit function (2) as:

$$\Pi(S) = \sum_{\boldsymbol{\tau} \in \mathcal{T}^+} \alpha_{\boldsymbol{\tau}} [\pi_{g_1(\boldsymbol{\tau},S)} + p - f(g_2(\boldsymbol{\tau}, S))] - p - K|S|$$

and

$$\begin{aligned}
\min_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} C_j(S) &= \min_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}} \left\{[\pi_{g_1(\boldsymbol{\tau},S\cup\{j\})} - f(g_2(\boldsymbol{\tau}, S\cup\{j\}))] - [\pi_{g_1(\boldsymbol{\tau},S)} - f(g_2(\boldsymbol{\tau}, S))]\right\} - K \\
&\geq \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}} \min_{S^{\boldsymbol{\tau}}:I\subseteq S^{\boldsymbol{\tau}}\subseteq(\mathcal{N}\setminus O)} \left\{[\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}}\cup\{j\})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}\cup\{j\}))] - [\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}))]\right\} - K \\
&= \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}} \delta^-(\boldsymbol{\tau}, j, I, O) - K \qquad (7) \\
\max_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} C_j(S) &= \max_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}} \left\{[\pi_{g_1(\boldsymbol{\tau},S\cup\{j\})} - f(g_2(\boldsymbol{\tau}, S\cup\{j\}))] - [\pi_{g_1(\boldsymbol{\tau},S)} - f(g_2(\boldsymbol{\tau}, S))]\right\} - K \\
&\leq \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}} \max_{S^{\boldsymbol{\tau}}:I\subseteq S^{\boldsymbol{\tau}}\subseteq(\mathcal{N}\setminus O)} \left\{[\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}}\cup\{j\})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}\cup\{j\}))] - [\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}))]\right\} - K \\
&= \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}} \delta^+(\boldsymbol{\tau}, j, I, O) - K \qquad (8)
\end{aligned}$$

where

$$\begin{aligned}
\delta^-(\boldsymbol{\tau}, j, I, O) &= \min_{S^{\boldsymbol{\tau}}:I\subseteq S^{\boldsymbol{\tau}}\subseteq\mathcal{N}\setminus O} [\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}}\cup\{j\})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}\cup\{j\}))] - [\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}))] \\
\delta^+(\boldsymbol{\tau}, j, I, O) &= \max_{S^{\boldsymbol{\tau}}:I\subseteq S^{\boldsymbol{\tau}}\subseteq\mathcal{N}\setminus O} [\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}}\cup\{j\})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}\cup\{j\}))] - [\pi_{g_1(\boldsymbol{\tau},S^{\boldsymbol{\tau}})} - f(g_2(\boldsymbol{\tau}, S^{\boldsymbol{\tau}}))]
\end{aligned}$$

For $\boldsymbol{\tau} \in \mathcal{T}^+$, $\delta^-(\boldsymbol{\tau}, j, I, O)$ ($\delta^+(\boldsymbol{\tau}, j, I, O)$) is the minimum (maximum) difference in profit, net of substitution cost, earned by the firm on consumers of type $\boldsymbol{\tau}$ when product j is added to a set which contains all the products in I and none of the products in O. Algorithm 5 shows how to compute these values. Note that we use a slight abuse of notation when we write $j \in \boldsymbol{\tau}$ since $\boldsymbol{\tau}$ is a vector. Calculating $\delta^+(\boldsymbol{\tau}, j, I, O)$ and $\delta^-(\boldsymbol{\tau}, j, I, O)$ using Algorithm 5 for a given $\boldsymbol{\tau} \in \mathcal{T}^+$ can be done in $O(n)$. Example 5 illustrates the calculation of the $\delta^-(\boldsymbol{\tau}, j, I, O)$ and $\delta^+(\boldsymbol{\tau}, j, I, O)$ values.

Example 5. Let $n = 4$ and $\pi_1 = 10, \pi_2 = 8, \pi_3 = 5, \pi_4 = 7$. Let $f(k) = 0$ for $k = 1, ..., n$. Consider type $\boldsymbol{\tau} = (3,1,2,4)$ and $j = 1$, so that $x = 2$. Table 5 shows the values of $\delta^+(\boldsymbol{\tau}, j, I, O)$ and $\delta^-(\boldsymbol{\tau}, j, I, O)$ as a function of I and O. □

Algorithm 5 Calculating $\delta^+(\boldsymbol{\tau}, j, I, O)$ and $\delta^-(\boldsymbol{\tau}, j, I, O)$ for a given type $\boldsymbol{\tau} = (\tau_1, ..., \tau_t) \in \mathcal{T}^+$.

if $j \in \boldsymbol{\tau}$ **then**

Let x be such that $\tau_x = j$.

Let y be the lowest index such that $\tau_y \in \mathcal{N} \backslash (I \cup O)$.

Let z be the lowest index such that $\tau_z \in I$. If $\tau_1, ..., \tau_t \notin I$, let $z = t+1$.

if Case 0: $z < x$ **then**

$\delta^+(\boldsymbol{\tau}, j, I, O) = \delta^-(\boldsymbol{\tau}, j, I, O) = 0$

end if

if Case 1: $y = x$ and $x < z < t+1$ **then**

$\delta^+(\boldsymbol{\tau}, j, I, O) = \pi_j - f(x) - \min\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq z\}$

$\delta^-(\boldsymbol{\tau}, j, I, O) = \pi_j - f(x) - \max\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq z\}$

end if

if Case 2: $y = x$ and $z = t+1$ **then**

$\delta^+(\boldsymbol{\tau}, j, I, O) = \pi_j - f(x) - \min\left\{0, \min\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq t\}\right\}$

$\delta^-(\boldsymbol{\tau}, j, I, O) = \pi_j - f(x) - \max\left\{0, \max\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq t\}\right\}$

end if

if Case 3: $y < x$ and $x < z < t+1$ **then**

$\delta^+(\boldsymbol{\tau}, j, I, O) = \max\left\{0, \pi_j - f(x) - \min\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq z\}\right\}$

$\delta^-(\boldsymbol{\tau}, j, I, O) = \min\left\{0, \pi_j - f(x) - \max\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq z\}\right\}$

end if

if Case 4: $y < x$ and $z = t+1$ **then**

$\delta^+(\boldsymbol{\tau}, j, I, O) = \max\left\{0, \pi_j - f(x) - \min\left\{0, \min\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq t\}\right\}\right\}$

$\delta^-(\boldsymbol{\tau}, j, I, O) = \min\left\{0, \pi_j - f(x) - \max\left\{0, \max\{\pi_{\tau_k} - f(k) : \tau_k \notin O \text{ and } x+1 \leq k \leq t\}\right\}\right\}$

end if

else $(j \notin \boldsymbol{\tau})$

$\delta^+(\boldsymbol{\tau}, j, I, O) = \delta^-(\boldsymbol{\tau}, j, I, O) = 0$

end if

I	O	y	z	Case	δ^+	δ^-
$\{3\}$	$\emptyset$	2	1	0	0	0
$\{4\}$	$\{3\}$	2	4	1	3	2
$\emptyset$	$\{3\}$	2	5	2	10	2
$\{4\}$	$\emptyset$	1	4	3	3	0
$\emptyset$	$\emptyset$	1	5	4	10	0

Table 5 $\delta^+(\boldsymbol{\tau}, j, I, O)$ **and** $\delta^-(\boldsymbol{\tau}, j, I, O)$ **in Example 5.**

Let $\Delta^-(j,I,O) = \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}}\delta^-(\boldsymbol{\tau},j,I,O)$ and $\Delta^+(j,I,O) = \sum_{\boldsymbol{\tau}\in\mathcal{T}^+} \alpha_{\boldsymbol{\tau}}\delta^+(\boldsymbol{\tau},j,I,O)$. From (7), $\Delta^-(j,I,O) - K$ is a lower bound on $\min_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} C_j(S)$ and from (8), $\Delta^+(j,I,O) - K$ is an upper bound on $\max_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} C_j(S)$. Calculating these two values can be done in $O(|\mathcal{T}^+|n)$. We formalize the *In-Out Algorithm* (Part I) below. Note that, when $K=0$, and $f(k)=0$ for $k=1,...,n$, the Algorithm can be sped up by initializing $I=M$ where M is defined in Proposition 1. The complexity of Part I of the *In-Out Algorithm* is $O(|\mathcal{T}^+|n^3)$.

Algorithm 6 In-Out Algorithm (Part I)

Set $I=O=\emptyset$, Change:=1.
while Change =1 **do**
  Change:=0,
  for all $j\in\mathcal{N}\setminus(I\cup O)$ **do**
    if $\Delta^-(j,I,O)\geq K$ **then**
      $I := I\cup\{j\}$, Change:=1.
    else
      if $\Delta^+(j,I,O)\leq K$ **then**
        $O := O\cup\{j\}$, Change:=1.
      end if
    end if
  end for
end while

Proposition 2. *Let I,O be the sets given by Part I of the* In-Out Algorithm. *There exists an optimal assortment S^*, such that $I\subseteq S^*\subseteq(\mathcal{N}\setminus O)$. Moreover, if, at the end of the algorithm, $I\cup O=\mathcal{N}$, then I is optimal.*

`Proof` We prove the result by induction. In the first iteration of the algorithm (when $I=O=\emptyset$) the products that are added to set I are such that $\Delta^-(j,\emptyset,\emptyset)\geq K$, which implies that $\min_{S:S\subseteq\mathcal{N}} C_j(S)\geq 0$ and thus $\Pi(S\cup\{j\})\geq\Pi(S)$ for all $S\subseteq\mathcal{N}$. Similarly all the products that are included in set O are such that $\Delta^+(j,\emptyset,\emptyset)\leq K$ which implies that $\max_{S:S\subseteq\mathcal{N}} C_j(S)\leq 0$ and thus $\Pi(S\cup\{j\})\leq\Pi(S)$ for all $S\subseteq\mathcal{N}$. Therefore, there must exist an optimal assortment S such that $I\subseteq S\subseteq(\mathcal{N}\setminus O)$ where I and O are obtained at the end of the first iteration.

Now assume that this is true for the k-th iteration and let I and O be the sets which are obtained at the end of the k-th iteration. The products added to I during the $(k+1)$-th iteration

are such that $\Delta^-(j,I,O) \geq K$ which implies that $\min_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} C_j(S) \geq 0$, which means that $\Pi(S\cup\{j\}) \geq \Pi(S)$ for all $I \subseteq S \subseteq (\mathcal{N}\setminus O)$ and we can restrict our attention to these assortments by the induction hypothesis. Similarly, the products added to O during the $(k+1)$-th iteration are such that $\Delta^+(j,I,O) \leq K$ which implies that $\max_{S:I\subseteq S\subseteq(\mathcal{N}\setminus O)} C_j(S) \leq 0$, which means that $\Pi(S\cup\{j\}) \leq \Pi(S)$ for all $I \subseteq S \subseteq (\mathcal{N}\setminus O)$ and we can restrict our attention to these assortments by the induction hypothesis. Hence, there must exist an optimal assortment S such that $I \subseteq S \subseteq (\mathcal{N}\setminus O)$ where I and O are obtained at the end of the $(k+1)$-th iteration. Q.E.D.

The following example illustrates how Part I of the *In-Out Algorithm* works.

EXAMPLE 6. Let $n=4$. Let $\pi_1=8, \pi_2=7$, $\pi_3=5$ and $\pi_4=18$.

Let $\alpha_{\boldsymbol{\tau}} = \frac{1}{6}$ for $\boldsymbol{\tau} \in \{(1),(2),(3),(2,3),(3,4),(1,2,4)\}$ and 0 otherwise. Let $K=0$ and $f(k)=0$ for $k=1,...,n$. The *In-Out Algorithm* finds an optimal assortment after 4 iterations. Table 6 shows the value of the I,O sets as well as the $\Delta^-(j,I,O)$ and $\Delta^+(j,I,O)$. in each iteration.

Iteration	I	O		1	2	3	4
1	$\emptyset$	$\emptyset$	$\Delta^-(j,I,O)$	-0.33	-0.33	-1.33	**0**
			$\Delta^+(j,I,O)$	2.67	3.50	2.50	6.00
2	$\{4\}$	$\emptyset$	$\Delta^-(j,I,O)$	-0.33	-0.33	-1.33	/
			$\Delta^+(j,I,O)$	1.50	3.50	**-0.50**	/
3	$\{4\}$	$\{3\}$	$\Delta^-(j,I,O)$	-0.33	**0.50**	/	/
			$\Delta^+(j,I,O)$	1.50	3.50	/	/
4	$\{2,4\}$	$\{3\}$	$\Delta^-(j,I,O)$	**1.50**	/	/	/
			$\Delta^+(j,I,O)$	1.50	/	/	/
end	$\{1,2,4\}$	$\{3\}$					

Table 6 Iterations of Part I of the In-Out Algorithm in Example 6.

During the first iteration, product 4 is added to set I because $\Delta^-(4,\emptyset,\emptyset)=0$. During the second iteration, product 3 is added to set O because $\Delta^+(3,\{4\},\emptyset)=-0.50<0$. During the third iteration, product 2 is added to set I because $\Delta^-(2,\{4\},\{3\})=0.50>0$. During the fourth iteration, product 1 is added to set I because $\Delta^-(1,\{2,4\},\{3\})=1.50>0$. After the fourth iteration, all 4 products have been allocated to the sets I and O, i.e., $I\cup O=\mathcal{N}$ so the optimal solution is $I=\{1,2,4\}$. □

In cases where $I\cup O\neq\mathcal{N}$, an optimal assortment can be found by enumerating all sets $T\subseteq\mathcal{N}\setminus(I\cup O)$ and looking for the one that gives the highest value of $\Pi(I\cup T)$. We propose part II of the *In-Out Algorithm* in order to speed up this process. Let $m=n-|I|-|O|$. Before starting the algorithm we renumber the products such that products 1 to m belong to set $\mathcal{N}\setminus(I\cup O)$ and $\pi_1\geq\pi_2\geq...\geq\pi_m$. Part II starts with I and O from Part I as inputs and goes as follows. If a product does not meet the $\Delta^-(j,S_i,T_i)\geq K$ nor $\Delta^+(j,S_i,T_i)\leq K$ conditions, it is not clear if this

product should be added to a set that contains all the products in S_i and none of the products in T_i. Therefore, we consider both cases: when j is added and when it is not. Using this idea, Part II of the *In-Out Algorithm* constructs N_m candidate assortments. The optimal assortment is the one with the highest profit amongst these. Example 7 illustrates how Part II of the *In-Out Algorithm* works.

Algorithm 7 In-Out Algorithm (Part II): given I and O as given in Part I

$N_0 = 1$, $S_1 = I$, $T_1 = O$.
for $j = 1, ..., m$ **do**
  $N_j = N_{j-1}$
  for $i = 1, ..., N_{j-1}$ **do**
    if $\Delta^-(j, S_i, T_i) \geq K$ **then**
      $S_i := S_i \cup \{j\}$
    else
      if $\Delta^+(j, S_i, T_i) \leq K$ **then**
        $T_i := T_i \cup \{j\}$.
      else
        $N_j = N_j + 1$, $S_{N_j} = S_i \cup \{j\}$, $T_{N_j} = T_i$, $T_i = T_i \cup \{j\}$
      end if
    end if
  end for
end for
Calculate $\Pi(S_i)$ for $i = 1, ..., N_m$. Let $S^* = \arg\max_{i=1,...,N_m} \Pi(S_i)$.

EXAMPLE 7. Let $n = 5$. Let $\pi_1 = 8, \pi_2 = 5$, $\pi_3 = 3, \pi_4 = 14$ and $\pi_5 = 5$. Let $\alpha_{\boldsymbol{\tau}} = \frac{1}{5}$ for $\boldsymbol{\tau} \in \{(1,3,2),(1,3,4,5),(2,4,3,1,5),(3,2),(5,4,2,1,3)\}$ and 0 otherwise. Let $K = 0$ and $f(k) = 0$ for $k = 1, ..., n$. Part I of the *In-Out Algorithm* yields $I = \{4\}$ and $O = \{5\}$ so that $\mathcal{N} \backslash (I \cup O) = \{1, 2, 3\}$ and $m = 3$. We start the part II of the *In-Out Algorithm* with $S_1 = \{4\}$ and $T_1 = \{5\}$. First we examine product 1: we have $\Delta^-(1, \{4\}, \{5\}) = -0.6 < 0$ and $\Delta^+(1, \{4\}, \{5\}) = 2.6 > 0$, so we set $T_1 = \{1, 5\}$ and create a second candidate assortment: $S_2 = \{1, 4\}$, $T_2 = \{5\}$. Next we examine product 2 and see if it should be added to the two candidate assortments. With respect to the first candidate assortment, we have $\Delta^-(2, \{4\}, \{1, 5\}) = -1.8 < 0$ and $\Delta^+(2, \{4\}, \{1, 5\}) = 0.2$ so we set $T_1 = \{1, 2, 5\}$ and create a third candidate assortment: $S_3 = \{2, 4\}$ and $T_3 = \{1, 5\}$. With respect to the second

candidate assortment, we have $\Delta^-(2,\{1,4\},\{5\}) = -1.8 < 0$ and $\Delta^+(2,\{1,4\},\{5\}) = -0.8 < 0$ so we set $T_2 = \{2,5\}$. Finally we examine product 3 and see if it should be added to the three candidate assortments. With respect to the first candidate assortment, we have $\Delta^-(3,\{4\},\{1,2,5\}) = \Delta^+(3,\{4\},\{1,2,5\}) = -1 < 0$ so we set $T_1 = \{1,2,3,5\}$. With respect to the second candidate assortment, we have $\Delta^-(3,\{1,4\},\{2,5\}) = \Delta^+(3,\{1,4\},\{2,5\}) = 0.6 > 0$ so we set $S_2 = \{1,3,4\}$. With respect to the third candidate assortment, we have $\Delta^-(3,\{2,4\},\{1,5\}) = \Delta^+(3,\{2,4\},\{1,5\}) = -3 < 0$ so we set $T_3 = \{1,3,5\}$. At the end we have three candidate assortments: $S_1 = \{4\}$, $S_2 = \{1,3,4\}$ and $S_3 = \{2,4\}$ which yield a profit equal to 8.4, 9.4 and 8.6 respectively. Hence the optimal assortment is $S_2 = \{1,3,4\}$. Table 7 illustrates the construction of the candidate assortments.

Iteration j	N_j	S_1	T_1	S_2	T_2	S_3	T_3
0	1	$\{4\}$	$\{5\}$				
1	2	$\{4\}$	$\{1,5\}$	$\{1,4\}$	$\{5\}$		
2	3	$\{4\}$	$\{1,2,5\}$	$\{1,4\}$	$\{2,5\}$	$\{2,4\}$	$\{1,5\}$
3	3	$\{4\}$	$\{1,2,3,5\}$	$\{1,3,4\}$	$\{2,5\}$	$\{2,4\}$	$\{1,3,5\}$

Table 7 Iterations of Part II of the In-Out Algorithm in Example 7

The complexity of Part II of the *In-Out Algorithm* is the same as that of the enumeration, i.e., $O(|\mathcal{T}^+|n2^n)$ method because, in theory, the number of candidate assortments can be as large as 2^n. However this is very unlikely to happen in practice. Table 11 in our numerical study shows that the Algorithm is in practice very fast, e.g., on average more than 10,000 times faster than enumeration when $n = 20$.

6. Numerical Study

The objective of this numerical study is twofold. First, we test the performance of the four heuristics presented in Section 4 by computing their optimality gap and study how it varies with the size of the problem, the fixed and substitution costs. Second, we show how the fixed and substitution costs affect the number of products in set I and O generated by Part I of the *In-Out Algorithm*, and compare the computation time of all of the solutions methods, i.e., the four heuristics, the enumeration method, and the *In-Out Algorithm*. Based on this two-part analysis we provide recommendations on the conditions under which the methods are appropriate.

In our numerical experiments we find that the performance of the four heuristics is mostly a function of the number of products n, the fixed cost K, and the substitution penalty function f; the other parameters, namely, the number of types in $\mathcal{T}^+$, the profit margins $\pi_1, ..., \pi_n$, and the lost sale penalty cost p do not appear to have a direct impact. We show a summary of our numerical

results in Tables 8 to 11. Each number in the tables is the average over 1000 problem instances. In each problem instance, the types in $\mathcal{T}^+$ are generated randomly and made equally likely, that is, we set $\alpha_{\boldsymbol{\tau}} = \frac{1}{|\mathcal{T}^+|}$ for $\boldsymbol{\tau} \in \mathcal{T}^+$. The profit margins are obtained using a discrete uniform distribution between 1 and 20. Finally we set $p = 0$.

In Table 8, we compare the performance of the heuristics with respect to the number of products. We use $n \in \{5, 10, 15, 20, 25\}$, and set the number of types in $\mathcal{T}^+$ also equal to n. We use $K = b = 0$. For a given scenario we report the percentage of instances in which each heuristic gives the optimal solution (% opt), the average optimality gap (avg OG), and the maximum optimality gap (max OG), where the optimality gap is computed as $OG = \frac{\Pi(S^*) - \Pi(S)}{\Pi(S^*)}$, where S^* is the optimal assortment and S is the assortment obtained by the heuristic.

heuristic	n	% opt	avg OG (%)	max OG (%)
MP	5	76	1.36	13.64
	10	56	1.48	13.01
	15	32	2.15	11.93
	20	29	1.95	8.43
	25	19	2.02	7.30
GA	5	98	0.35	10.61
	10	93	0.13	4.35
	15	78	0.55	10.22
	20	82	0.27	4.03
	25	75	0.25	4.27
GR	5	98	0.59	31.58
	10	76	1.13	12.50
	15	76	0.64	19.55
	20	66	0.75	9.97
	25	56	0.80	6.18
MB	5	98	0.03	1.59
	10	91	0.11	2.70
	15	88	0.14	3.64
	20	87	0.12	1.67
	25	76	0.17	1.98

Table 8 Performance of the four heuristics with respect to n.

We see in Table 8 that the performance of all four heuristics deteriorates as n increases, this is especially true when it comes to the percentage of problem instances for which they obtain the optimal solution. Overall the MB heuristic performs the best followed by the GA and GR heuristics.

In Table 9 we compare the performance of the four heuristics with respect to the fixed cost and substitution penalty. We set $n = |\mathcal{T}^+| = 15$ and vary the fixed cost $K \in \{0, 2, 4\}$. Regarding the substitution penalty, we assume $f(k) = b(k-1)$ and vary $b \in \{0, 1, 2\}$.

Heuristic		% opt b=0	% opt b=1	% opt b=2	Avg OG (%) b=0	Avg OG (%) b=1	Avg OG (%) b=2	Max OG (%) b=0	Max OG (%) b=1	Max OG (%) b=2
MP	K=0	32	40	44	2.15	2.13	1.54	11.93	7.08	4.95
	K=2	33	42	28	8.47	8.98	7.96	37.93	40.88	100.00
	K=4	43	39	47	12.08	14.49	20.63	100.00	100.00	100.00
GA	K=0	78	87	98	0.55	0.17	0.18	10.22	2.97	2.33
	K=2	88	92	90	0.60	0.55	0.24	11.67	10.13	24.47
	K=4	95	96	98	0.38	0.21	0.27	14.29	12.16	7.69
GR	K=0	76	81	84	0.64	0.60	0.48	19.55	8.47	5.85
	K=2	89	84	85	0.63	1.84	0.69	12.50	11.46	16.67
	K=4	82	80	90	1.85	2.27	3.61	23.08	52.50	100.00
MB	K=0	88	91	87	0.14	0.05	0.09	3.64	2.11	4.83
	K=2	75	75	80	1.12	1.25	1.01	37.93	12.82	31.67
	K=4	76	83	95	2.08	1.79	2.04	25.00	29.41	24.14

Table 9 **Performance of the four heuristics with respect to K and b.**

The results show that even though four heuristics seem to be reaching the optimal solution with a higher frequency as b and K increase, the average optimality gap actually increases, indicating that the performance of all four heuristics actually gets worse as b and K increase. Also we see that the impact is more significant for K than for b. We also see that when $K = 0$, the MB heuristic performs the best but it is the GA heuristic that does better when K becomes larger. The MP heuristic always performs the worst.

The size of the I and O sets generated in Part I has a significant impact on the overall computational time of the *In-Out Algorithm*. Next we study the impact of the model parameters on the number of products included in the I and O sets at the end of part I. Equations (7) and (8) and Algorithm 5 show that the penalty cost p does not have an impact on $\delta^-(\boldsymbol{\tau}, j, I, O)$ and $\delta^+(\boldsymbol{\tau}, j, I, O)$, and thus has no effect on the size of the I and O sets generated in part I. The fixed cost K and substitution cost $f(k)$ have an impact but it is non-monotone. We set $n = |\mathcal{T}^+| = 10$ and vary the fixed cost $K \in \{0, 2, 4\}$. Regarding the substitution cost, we assume $f(k) = b(k-1)$ and vary $b \in \{0, 1, 2\}$. In Table 10, we examine how the average number of products in sets I and O changes with the value of b and K. We see that the average number of products included in the set I and O may decrease or increase with K and b.

Next, we compare the computation time of the different solution methods. Our numerical experiments indicate that only two parameters affect the computation time of the different methods: the number of products n and the number of types in $\mathcal{T}^+$, the other parameters (i.e., $\pi_1, ..., \pi_n$, $\alpha_{\boldsymbol{\tau}}$ for $\boldsymbol{\tau} \in \mathcal{T}^+$, K, b and p) do not have a noticeable impact on it. In Table 11, we report the results of varying the number of products n in $\{5, 10, ..., 25, 30\}$ and the number of types in $\mathcal{T}^+$ in $\{n, n+5..., n+20\}$. In each problem instance, the types in $\mathcal{T}^+$ were generated randomly and

$\|I\|\&\|O\|$ in Part I		b = 0	b = 1	b = 2
K = 0	$\|I\|$	1.56	4.92	5.36
	$\|O\|$	0.87	4.53	6.29
K = 2	$\|I\|$	0.01	0.41	0.09
	$\|O\|$	2.21	7.3	1.44
K = 4	$\|I\|$	0.01	0.03	0.09
	$\|O\|$	7.44	8.81	9.3

Table 10 **Average number of products in sets I and O with respect to K and b.**

made equally likely, that is, we set $\alpha_{\boldsymbol{\tau}} = \frac{1}{|\mathcal{T}^+|}$ for $\boldsymbol{\tau} \in \mathcal{T}^+$. Also the profit margin of each product were obtained using a discrete uniform distribution between 1 and 20. Finally we set $f(k) = 0$ for $k = 1, ..., n$ and $K = p = 0$. All averages were calculated over 1000 problem instances (except when the computation time is over 4 hours, in which case we used 10 problem instances). In all cases we used Matlab 7.6 on a Lenovo Laptop with a Intel Core 2 Duo Processor P8600 with 2.4GHz and 4GB of RAM. We report the average computation times of the four heuristics (MP, GA, GR and MB), the enumeration method (Enum), Parts I and II of the *In-Out Algorithm* separately (IO-I and IO-II) and together (IO-I&II). We also report the average number of products put into sets I and O at the end of part I of the *In-Out Algorithm* ($|I| + |O|$ part I), the average number of candidate assortments generated by Part II of the *In-Out Algorithm*, i.e., N_m (# assortments IO), and the number of assortments considered by the enumeration method, which is equal to 2^n (# assortments Enum.).

We see that the enumeration quickly becomes impractical; it takes over 24 hours for problems with 20 products. The *In-Out Algorithm* (parts I and II) produces an optimal solution in less than 4 minutes for up to 25 products and around 5 hours for $n = 30$. Also its computation time does not vary significantly with the number of types for a fixed value of n. Part I of the *In-Out Algorithm* is very fast but its efficiency (measured by the number of products which get assigned to set I or O) decreases with n and $|\mathcal{T}^+|$. The four heuristics can handle a very high number of products and types, their computation time is still under 10 seconds for n=100.

Based on this numerical analysis, we recommend using the *In-Out Algorithm* for product categories with up to 25 or 30 products, depending on how much time is available. For product categories with more than 30 products, we recommend first using Part I of the *In-Out Algorithm* in order to do a first sorting of the products, followed by the MB heuristic when K is very low, and the GA heuristic otherwise. However one should keep in mind that the heuristics can lead to substantial optimality gaps.

		Computation time								$\lvert I\rvert + \lvert 0\rvert$	# assortments	
n	nt	MP	GA	GR	MB	Enum.	part I	part II	parts I+II	part I	IO	Enum.
5	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.76	1.23	32
5	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.49	1.61	32
5	15	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	3.58	1.6	32
5	20	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	3.5	1.54	32
5	25	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	3.44	1.74	32
10	10	0.00	0.01	0.01	0.01	0.14	0.00	0.01	0.01	5.1	11.05	1024
10	15	0.00	0.01	0.01	0.01	0.16	0.00	0.02	0.02	3.85	19.76	1024
10	20	0.00	0.01	0.01	0.02	0.19	0.00	0.02	0.03	3.75	20.78	1024
10	25	0.00	0.01	0.01	0.02	0.21	0.01	0.03	0.03	3.22	22.29	1024
10	30	0.00	0.01	0.01	0.02	0.23	0.01	0.03	0.04	3.32	22.41	1024
15	15	0.00	0.02	0.02	0.03	67.87	0.01	0.16	0.17	3.72	166.61	32768
15	20	0.00	0.02	0.02	0.04	69.91	0.01	0.22	0.23	3.29	183.41	32768
15	25	0.00	0.02	0.02	0.04	72.00	0.01	0.28	0.28	3.55	207.48	32768
15	30	0.00	0.03	0.03	0.05	72.90	0.01	0.48	0.49	3.42	299.55	32768
15	35	0.00	0.03	0.03	0.05	73.46	0.01	0.37	0.39	3.51	204.47	32768
20	20	0.00	0.04	0.04	0.06	88182.00	0.01	3.54	3.55	3.66	1755.45	1048576
20	25	0.00	0.04	0.04	0.07	87716.00	0.02	2.79	2.80	3.46	1509.03	1048576
20	30	0.00	0.05	0.05	0.08	88019.00	0.02	3.73	3.75	3.54	1684.37	1048576
20	35	0.01	0.05	0.05	0.09	88689.00	0.02	4.83	4.85	2.92	2024.51	1048576
20	40	0.01	0.06	0.06	0.10	89102.00	0.02	8.02	8.05	2.89	2703.03	1048576
25	25	0.01	0.07	0.07	0.12	>100 days†	0.02	202.71	202.72	2.9	17733.33	33,554,432
25	30	0.01	0.08	0.08	0.14	>100 days†	0.02	130.99	131.02	2.89	16134.26	33,554,432
25	35	0.01	0.09	0.09	0.16	>100 days†	0.02	408.38	408.40	2.56	30608.24	33,554,432
25	40	0.01	0.09	0.10	0.17	>100 days†	0.03	442.51	442.53	2.54	31258.02	33,554,432
25	45	0.01	0.10	0.11	0.18	>100 days†	0.03	299.47	299.50	2.95	24123.1	33,554,432
30	30	0.01	0.12	0.13	0.21	>100 years†	0.02	20635.21	20635.23	4.12	209599.6	1,073,741,824
30	35	0.01	0.14	0.14	0.23	>100 years†	0.03	12469.02	12469.04	4.42	163970	1,073,741,824
30	40	0.01	0.15	0.15	0.26	>100 years†	0.03	24439.88	24439.91	2.82	241728	1,073,741,824
30	45	0.01	0.15	0.16	0.27	>100 years†	0.04	38967.73	38967.77	2.74	265722	1,073,741,824
30	50	0.01	0.17	0.17	0.29	>100 years†	0.04	49663.09	49663.13	2.03	302048	1,073,741,824

†obtained by extrapolation.

Table 11 **Average computation time (in seconds) of all solution methods, average number of products put in sets I and O in part I of the In-Out Algorithm and number of assortments to evaluate for the enumeration method and In-Out Algorithm.**

In terms of managerial insights, we find that the MP heuristic, which is the fastest, performs the worst of the four heuristics, which points to the idea that focusing solely on the profitability of the products independently of the nature of customer preferences does not benefit a retailer.

7. Robustness study

In this section, we study how the optimal solution to a given problem varies with the key parameters of our model. In particular, we answer the following question: how much profit is left on the table if the firm uses the wrong value for some key parameters of the model such as the proportions of customers of each type, the profit margins, the fixed cost or the substitution cost? In practice these values might be hard to estimate and/or they may evolve over time so the chosen assortment may

not always be based on the most accurate numbers. Since there are some switching costs involved with changing the assortment, the firm needs to know when it is necessary to re-evaluate the chosen assortment following a change in the environment, such as a promotion or a decrease/increase in costs.

We answer this question numerically. In all cases, our methodology is as follows. We first calculate the assortment S^r which is optimal for a given set of parameter values called the *base set* (the optimal solution is obtained using the In-Out Algorithm). The base set represents the values used by the firm, which may not be the correct ones. Then we vary some of the parameters (one at a time) around the base set. These numbers represent the 'correct' parameter values. Using these numbers, we compute the optimal profit π^* and the profit of assortment S^r under the correct parameters. Then we calculate the percentage optimality gap of assortment S^r, which measures the percentage loss in profit from not using the correct values of the parameters.

First we provide an example which shows that the optimality gap can be higher than 100%. In this example, we assume that the firm has correctly identified the set of possible consumer types in the population but has incorrectly calculated the proportion of customers of each type (for example, because it has neglected the impact of a recent promotion campaign by the manufacturer).

EXAMPLE 8. Let $n=2$. Suppose $\pi_1=10, \pi_2=3, \mathcal{T}^+=\{(1),(2),(1,2)\}, p=0, f(k)=0$ and $K=2$. The firm believes that $\tilde{\alpha}_{(1)}=\tilde{\alpha}_{(1,2)}=0.1$ and $\tilde{\alpha}_{(2)}=0.8$ and chooses assortment $S^r=\{2\}$. However, the true proportions are $\alpha_{(2)}=\alpha_{(1,2)}=0.1$ and $\alpha_{(1)}=0.8$ so the profit of assortment S^r is -1.4, whereas the profit of the optimal assortment ($\{1\}$) is 7. The resulting percentage optimality gap is 120%.

We study the average optimality gap over 1000 problem instances when varying one parameter at at time.[4] In each problem instance, the base values are as follows. The number of products and types is set equal to 5. The profit margins are obtained using a discrete uniform distribution between 1 and 20. We set $p=f(k)=0$ and $K=1$. The types in $\mathcal{T}^+$ are randomly generated and each type in $\mathcal{T}^+$ is equally likely, i.e $\alpha_{\boldsymbol{\tau}}=\frac{1}{|\mathcal{T}^+|}$.

We study the impact of the consumer choice model by considering every possible vector $\alpha_{\boldsymbol{\tau}}$ such that each component is a multiple of 0.2. We study the impact of product profit margins by allowing each π_j value to vary in $\{-50\%,-25\%,0,+25\%,+50\%\}$. We study the impact of the fixed cost by letting $K\in\{0,0.5,1,1.5,...,5\}$. Finally we study the impact of the substitution cost by letting $b\in\{0,0.25,...,2\}$.

[4] In order to avoid cases where the average optimality gaps are dominated by a small number of extreme cases with very high values, we bound the values obtained by 100%. Hence our results are on the conservative side.

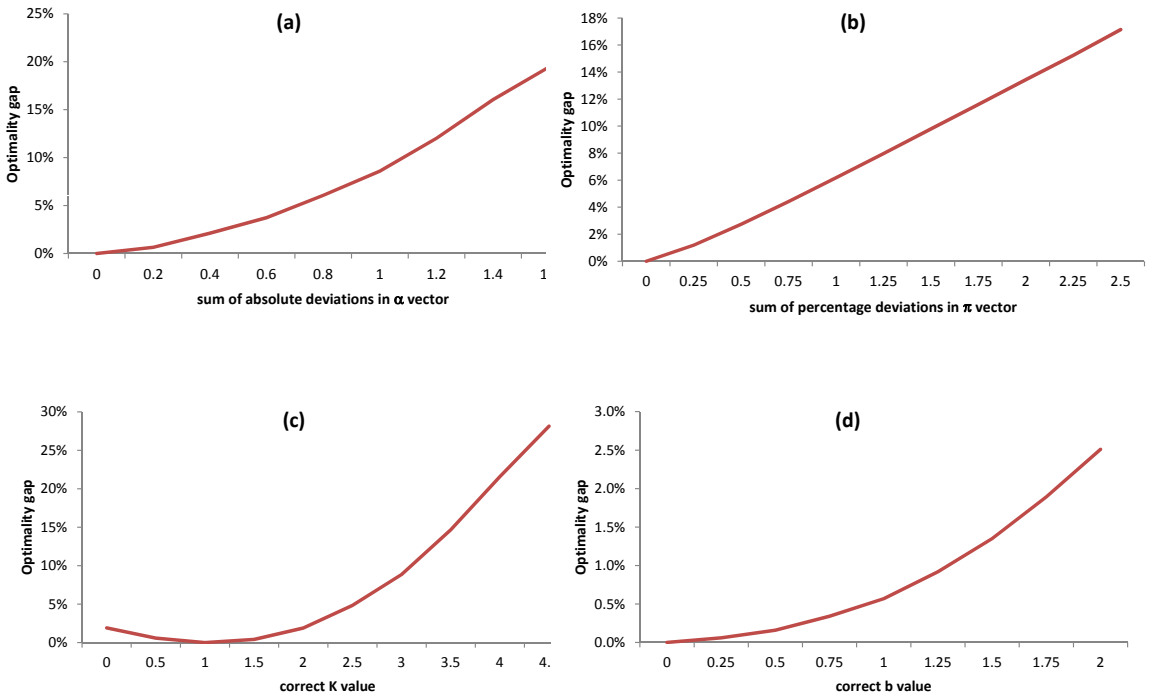

Figure 1 **Optimality gaps with respect to (a) α vector (b) π vector (c) fixed cost K (d) substitution cost b**

Our results are shown in Figure (a) to (d). In (a) we show that average optimality gap as a function of the sum of absolute deviations between the correct α vector and the value from the base set. In (b) we show the average optimality gap as a function of the sum of percentage deviations correct the π vector and the base set. In (c) to (d) we show the average optimality gap as a function of the correct K and b values.

It is not surprising to see that in all four cases, the optimality gaps increase as the deviations between the correct parameters and base values increase (for the fixed cost K remember that the base value is $K = 1$ so that the curve in (c) first decreases then increases). The most important insight comes from comparing these values across all four graphs. We see that miscalculating the value of the substitution cost b has little impact on the profit of the chosen assortment but this is not true for the α, π vectors and K. For example, a deviation of 0.4 in the α vector, e.g., assuming that $\alpha = (0.2, ..., 0.2)$ when the correct value is $(0.3, 0.1, 0.3, 0.1, 0.2)$ leads to an average 2.32% loss in profit.

In conclusion we see that it is important for the firm to correctly evaluate the α and π vectors as well as the fixed cost K. Using incorrect parameter values can lead to a substantial loss in profit.

8. Conclusion

In this research we study the assortment planning model under a ranking-based consumer (RBM) choice model. We demonstrate that while the RBM model is appealing, it is well known that the enumeration and integer programming methods quickly become impractical as the number of potential products to offer increases and that simple sorting rules often fail to find the optimal solution and have very poor worst case bounds. We further show that dominated products, i.e, products which are less profitable and less preferred than another product by all customers, cannot be eliminated as they might be included in the optimal assortment. We designed an algorithm, called the *In-Out Algorithm*, which always yields an optimal solution. In theory the *In-Out Algorithm* has the same complexity as the enumeration method but in practice it is much faster: more than 10,000 faster than enumeration for a problem with 20 products. Based on an extensive numerical analysis we recommend the use of our *In-Out Algorithm* for problems with up to 25 to 30 products. For an assortment problem with more products, we recommend the use of Part I of the *In-Out Algorithm* followed the GA heuristic (when the fixed cost is small) or the MB heuristic (when it is large), but the firm should keep in mind that the solution may be sub-optimal. We also show that it is important to correctly evaluate the consumer choice model, profitability of the products and the fixed cost because using incorrect parameters can lead to a substantial loss in profit.

We conclude our paper by discussing a few limitations of our research that we think are avenues for future research. We make the assumption of fixed prices that allows us to use the RBM model effectively. We recognize that this assumption is valid only when retailer charges the MSRP or any other set price. While allowing for endogenous pricing in our model is desirable it also makes the analysis fairly complex. Endogenous pricing will demand a consumer choice model where the purchase probabilities are a function of prices, we hope to include these complex interactions in our future work.

Another limiting assumption is the that of stock-out based substitution which obviates the need for taking into account stock levels of the product. Under stock-out based substitution, the demand for a product not only depends on the initial assortment, but also on the inventory level of the other products. Also, the profit value depends on the sequence in which customers come to the store (Mahajan and van Ryzin (2001a)). Although some research exists to understand the performance of simulation based optimization (Mahajan and van Ryzin (2001a)) and optimal solutions under special models (Honhon et al. (2012a)), we believe there is scope for improving the search for optimal assortments in this space.

References

Alexouda, G. and K. Paparrizos (2001), 'A genetic algorithm approach to the product line design problem using the sellers return criterion: An extensive comparative computational study.', *Eur.J. Oper. Res.* **134**(1), 165–178.

Anupindi, R., S. Gupta and M. Venkataramanan (2008), *Managing Variety on the Retail Shelf: Using Scanner Data to Rationalize Assortments.*, In Retail Supply Chain Management: Quantitative Models and Empirical Studies, Springer, New York.

Balakrishnan, P. V., R. Gupta and V. Jacob (2004), 'Development of hybrid genetic algorithms for product line designs', *IEEE Trans. Systems, Man, Cybernetics* **34**(1), 468–483.

Balakrishnan, P. V. and V. S. Jacob (1996), 'Genetic algorithms for product design', *Management Sci.* **42**(8), 1105–1117.

Belloni, A., R. Freund, M. Selove and D. Simester (2008), 'Optimizing product line designs: Efficient methods and comparisons', *Management Science* **54**(9), 1544–1552.

Carlton, D. W. and J. A. Chevalier (2001), 'Free riding and sales strategies for the internet', *Journal of Industrial Economics* **49**(4), 441–461.

Cho, D. C., E. L. Johnson, M. Padberg and M. R. Rao (1983), 'On the uncapacitated plant location problem. i: Valid inequalities and facets', *Mathematics of Operations Research* **8**(4), 579–589.

Dobson, G. and S. Kalish (1988), 'Positioning and pricing a product line', *Marketing Science* **7**(2), 107–125.

Dobson, G. and S. Kalish (1993), 'Heuristics for pricing and positioning a product line using conjoint and cost data', *Management Sci.* **39**(2), 160–175.

Farias, V.F., S. Jagabathula and D. Shah (2011), A non-parametric approach to modeling choice with limited data. Working Paper, MIT.

Gaur, V. and D. Honhon (2006), 'Assortment planning and inventory decisions under a locational choice model', *Management Science* **52**(10), 1528–1543.

Green, P. E. and A. M. Krieger (1985), 'Models and heuristics for product line selection', *Marketing Science* **4**(1), 1–19.

Green, P. E. and A. M. Krieger (1987), 'A simple heuristic for selecting "good" products in conjoint analysis', *Applications of Management Science* **5**, 131–153.

Honhon, D., S. Jonnalagedda and X. A. Pan (2012b), 'Optimal algorithms for assortment selection under ranking-based consumer choice models', *Manufacturing and Service Operations Management* . Forthcoming.

Honhon, D., V. Gaur and S. Seshadri (2012a), 'Profit maximization and risk reduction in sourcing from multiple suppliers', *Production and Operations Management* . Forthcoming.

Kohli, R. and R. Sukumar (1990), 'Heuristics for product-line design using conjoint analysis', *Management Sci.* **36**(12), 1464–1478.

Li, Z. (2007), 'A single-period assortment optimization model', *Production and Operations Management* **16**(3), 369–380.

Liu, Q. and G. van Ryzin (2008), 'On the choice-based linear programming model for network revenue management.', *Manufacturing and Service Operations Management* **10**(2), 288–310.

Mahajan, S. and G. van Ryzin (2001a), 'Stocking retail assortments under dynamic consumer substitution', *Operations Research* **49**(3), 334–351.

Mahajan, S. and G. van Ryzin (2001b), 'Inventory competition under dynamic consumer choice', *Operations Research* **49**(5), 646–657.

McBride, R. D. and F. S. Zufryden (1988), 'An integer programming approach to the optimal product line selection problem', *Marketing Science* **7**(2), 126–140.

Nair, S. K., L. S. Thakur and K. Wen (1995), 'Near optimal solutions for product line design and selection: Beam search heuristics', *Management Sci.* **41**(5), 767–785.

Pan, X. A. and D. Honhon (2012), 'Assortment planning for vertically differetiated products', *Production and Operations Management* . Forthcoming.

Smith, J. C., C. Lim and A. Alptekinoğlu (2009), 'New product introduction against a predator: A bilevel mixed-integer programming approach', *Naval Research Logistics* **56**(8), 714–729.

Smith, S.A and N. Agrawal (2000), 'Management of multi-item retail inventory systems with demand substitution', *Operations Research* **48**(1), 50–64.

Steiner, W. and H. Hruschka (2003), 'Genetic algorithms for product design: How well do they really work?', *Internat. J. Market Res.* **45**(2), 229–240.

Talluri, K.T and G. J. van Ryzin (2004), 'Revenue management under a general disrete choice model of consumer behavior', *Management Science* **50**(1), 15–33.

van Ryzin, G. and S. Mahajan (1999), 'On the relationship between inventory costs and variety benefits in retail assortments', *Management Science* **45**(11), 1496–1509.

Yunes, T. H., D. Napolitano, A. Scheller-Wolf and S. Tayur (2007), 'Building efficient product portfolios at john deere and company', *Operations Research* **55**(4), 615–629.