An algorithm for assortment optimization under parametric discrete choice models

Tien Mai
Singapore-MIT Alliance for Research and Technology (SMART), mai.tien@smart.mit.edu

Andrea Lodi
CERC - Data science, Polytechnique Montréal, andrea.lodi@polymtl.ca

This work concerns the assortment optimization problem that refers to selecting a subset of items that maximizes the expected revenue in the presence of the substitution behavior of consumers specified by a parametric choice model. The key challenge lies in the computational difficulty of finding the best subset solution, which often requires exhaustive search. The literature on constrained assortment optimization lacks a practically efficient method that is general to deal with different types of parametric choice models (e.g., the multinomial logit, mixed logit or general multivariate extreme value models).

In this paper, we propose a new approach that allows to address this issue. The idea is that, under a general parametric choice model, we formulate the problem into a binary nonlinear programming model, and use an iterative algorithm to find a binary solution. At each iteration, we propose a way to approximate the objective (expected revenue) by a linear function, and a polynomial-time algorithm to find a candidate solution using this approximate function. We also develop a greedy local search algorithm to further improve the solutions. We test our algorithm on instances of different sizes under various parametric choice model structures and show that our algorithm dominates existing exact and heuristic approaches in the literature, in terms of solution quality and computing cost.

Key words: parametric choice model, assortment optimization, binary trust region, greedy local search.
History:

1. Introduction

Assortment optimization is an important problem that arises in many practical applications such as retailing, online advertising, and social security. The problem refers to select a subset of items that maximizes the expected revenue in the presence of the substitution behavior of consumers specified by a choice model. Typically, an assortment decision consists of two main steps, namely, (i) training a choice model to predict the behavior of customers for a given set of products, and (ii) solving an assortment optimization problem built based on that trained choice model to get the best assortment decision. For the first step, there exists a number of parametric models based on the discrete choice framework (McFadden 1978), i.e., the framework that allows to describe the choices of decision makers among alternatives under certain general assumptions. These models are widely used in many demand modeling applications (e.g. Ben-Akiva and Lerman 1985), and believed to be accurate in many contexts. The second step often requires an exhaustive search over a large sets of feasible assortments, in which the large number of possible solutions could make the optimization problem intractable. Existing approaches often focus on designing algorithms, exact or heuristic, for some specific choice models and mostly considering uncapacitated problems, or problems under a simple upper bound constraint on the size of the assortment. In other words, to the best of our knowledge, it lacks a solution framework that is general enough to deal with generalized choice models, e.g., the multivariate extreme value (MEV) family model (McFadden 1978). In this paper, we address this issue by proposing a new algorithm that allows to solve problems under various parametric choice model structures.

The first step for an assortment decision is building a demand model that can predict the behavior of customers when they are offered a set of products. More precisely, we aim at specifying a probabilistic model that can assign a probability distribution over the products. The random utility maximization framework (McFadden 1978, Train 2003) is widely used in the context. The principle of this framework is that each product is associated with a random utility, and a customer selects a product in an assortment by maximizing his/her utilities. The utility associated with a product j is assumed to be a sum of two parts, one can be observed and the other one cannot be observed by the analyst. More precisely, an utility u_j associated with product j can be written as $u_j = \beta^{\mathrm{T}} a_j + \epsilon_j$, where a_j is the vector of attributes/features of product j and β is the vector of the model parameters, which can be obtained by estimating/training the choice model, and ϵ_j is the random part that cannot be observed by the analyst. Under the maximum utility framework, this way of modeling allows to calculate the probability that a customer selects a product i if he/she is offered an assortment S, that is $P(u_i \geq u_j, \forall j \in S)$. This probabilities allow to compute a log-likelihood function based on a set of observations, and then the model parameters can be

estimated using maximum likelihood estimation. Given a vector of parameters estimated, these probabilities also allow to compute the expected revenue given by an offered assortment.

Different assumptions made on the random terms ϵ_j lead to different choice models, and there are a number of existing choice models that can be used for modeling demand. Among the existing discrete choice models, the multinomial logit (MNL) is the most widely used due to its simple structure. However, there is an issue related to the independence of irrelevant alternatives (IIA) property of this model (Ben-Akiva and Lerman 1985), which may not hold in several contexts and lead to inaccurate prediction. There are a number of models that have been proposed in order to relax this property, e.g., the nested logit (Ben-Akiva 1973, Ben-Akiva and Lerman 1985), the cross-nested logit Vovsha and Bekhor (1998), the paired comparison logit (Koppelman and Wen 2000), the ordered generalized extreme value (Small 1987), and the network MEV model (Daly and Bierlaire 2006). These models all belong to the MEV family of models (McFadden 1978). The MEV model, in particular the cross-nested and network MEV models are generally flexible, as one can show that such models can approximate any random utility maximization models (Fosgerau et al. 2013).

Beside the MEV family, mixed logit (MMNL) is also convenient to relax the IIA from the MNL model. This model is refereed to as the *Random Parameter Logit* model, as it is extended from the MNL by assuming that parameters are random. Similar to some MEV models, the MMNL one is also able to approximate any random utilities choice model (McFadden and Train 2000). However, the choice probabilities given by the MMNL model have no closed form and often require simulation to approximate, leading to the fact the the estimation and the application of this model is expensive in some contexts.

It is worth mentioning that, apart from the aforementioned models (for which we refer to parametric choice models), non-parametric models have recently received a growing attention. In particular, Feige et al. (2011) have proposed a generic choice model for the case of limited data, where the choice behavior is represented by a distribution over all possible rankings of the alternatives. This non-parametric model is general, but the estimation as well as the application in revenue management problems is costly, as one needs to deal with a very large number of possible rankings, which is factorial on the number of products.

The difficulty when solving the assortment optimization problem under a discrete choice model lies in the fact that the resulting expected revenue function is highly nonlinear and non-convex, so in general, in order to obtain the best assortment, one must solve a mixed-integer nonlinear and non-convex optimization problem, which is computationally hard. For instance, if the number of products is 100, then the number of subsets that we have to consider (if there is no restriction on the size of the assortments) is 2^{100}. There is also a trade-off between having a flexible and accurate

(in prediction) demand model and the complexity of the corresponding optimization problem. For instance, under the multinomial logit (MNL) model, the objective function is simply a fraction of two linear functions, and there exist efficient algorithms that allow to find optimal assortments in polynomial time (Rusmevichientong et al. 2010). But for more flexible choice models, e.g., the mixed multinomial logit or nested logit models (Train 2003), the resulting objective functions are much more complicated, and the optimization problems becomes difficult to solve. Yet, to the best of our knowledge, the only approach that is general enough to handle a general class of parametric choice models is the simple greedy local search proposed by Jagabathula (2014). This approach is however time consuming, in particular when the number of products is large. In this paper, we exploit the structure of the objective function to design a new "local search type" algorithm, which is not only efficient to find a good solution, but also general enough to handle constrained problems.

Our contributions:

(i) We propose a new approach that allows to make an assortment decision where the customers' behavior can be captured by various choice models (most of the parametric choice models in the literature, including the MNL, MMNL and network MEV models). Our algorithm, called Binary Trust Region (BiTR), is based on the idea that we can iteratively approximate the objective function by a linear or quadratic one using Taylor's expansion. Then, we can perform a search over a local region using this approximate function to find a better assortment solution. In this context, at each iteration, the algorithm requires solving a sub-problem, which is a mixed-integer linear programming problem, to find a candidate solution. For some relevant special cases, we propose a polynomial-time algorithm that allows to solve these sub-problems exactly.

(ii) We suggest a way to improve solutions given by the BiTR by performing a greedy local search algorithm, i.e., searching over a local region by enumerating all feasible solutions in this region. We present a mathematical representation and investigate some theoretical properties of the approach under the MNL model, which would help to further improve the greedy algorithm.

(iii) We test our algorithm on instances under the MNL, MMNL and network MEV models using a real data set from a retail business. The results based on several instances of different sizes are promising, as our approach dominates existing heuristic and exact approaches in the literature, in terms of solution quality and computing cost.

(iv) Our approach provides an efficient way to make assortment decisions under flexible and general choice models, e.g., the MMNL and network MEV models. Thus, it could also be useful for the dynamic version of the static problem considered in this paper, i.e., the network revenue management problem (Liu and Van Ryzin 2008).

The remainder of the paper is structured as follows. In Section 2, we review the relevant literature in assortment optimization. In Section 3, we present in detail different parametric choice models that can be used to model demand, and the assortment problem under such models. We present our BiTR and the greedy local search in Section 4. The numerical results are provided in Section 5, and finally Section 6 concludes.

2. Literature review

There is a rich literature for the assortment problem under different parametric choice models. For the MNL problem, Talluri and Van Ryzin (2004) show that the unconstrained problem can be solved by enumerating a small number of revenue-ordered subsets. For the capacitated MNL (i.e., problem with an upper bound constraint on the size of the assortment), Rusmevichientong et al. (2010) show that the optimal assortment may no-longer be revenue-ordered. In addition, they suggest an efficient algorithm to find the best assortment with the complexity of $O(mC)$, where m is the number of products and C is the maximal number of products that an assortment can have. They also develop a method to learn the model parameters and optimize the profit at the same time. Rusmevichientong and Topaloglu (2012) consider a robust optimization version of the MNL problem, i.e. the model parameters are not known certainly, but belong to a compact uncertainty set. Interestingly, they show that for the uncapacitated problem, the revenue-ordered subsets remain optimal even when the goal is to maximize the worst-case expected revenue.

The problem under the MMNL model is typically NP-hard (Désir and Goyal 2014, Bront et al. 2009). Bront et al. (2009) present a mixed-integer linear programming (MILP) formulation for the MMNL problem, so the problem can be solved using a MILP solver, e.g., CPLEX, GUROBI. They also suggest a greedy heuristic that achieves good performance in their experiments. Méndez-Díaz et al. (2010) strengthen the mixed-integer programming formulation through valid inequalities. Rusmevichientong et al. (2014) consider two special cases of the uncapacitated MMNL model for which they show that the revenue-ordered subsets are optimal. There are also near-optimality algorithms for such problem, e.g., Désir and Goyal (2014) propose a fully polynomial time approximation scheme (FPTAS) for the capacity constrained MMNL problem.

There are also some studies focusing on the nested logit model. Davis et al. (2014) study the problem under the two-level nested logit model and show that if the nest dissimilarity parameters are all less than one and the no-purchase alternative belongs to a nest of its own, the uncapacitated problem can be solved to optimality in a computationally efficient manner. Gallego and Topaloglu (2014) extend this result for the uncapacitated problem and Li et al. (2015) consider the assortment problem under the d-level nested logit model. Yet, to the best of our knowledge, there is no study for the problem under the cross-nested model (an alternative/product can belong to more than one nest) or the general MEV model (McFadden 1978, Daly and Bierlaire 2006, Mai et al. 2017).

Jagabathula (2014) has recently proposed a local search algorithm, called ADXOpt, that can be used with any choice model. This algorithm is based on three simple operations, i.e., adding or removing one product, or exchanging an available product with a new one. Jagabathula (2014) also shows that if the problem is the capacitated MNL, ADXOpt converges to an optimal assortment solution. This algorithm has been shown to have very good performance in some numerical experiments (Bertsimas and Mišić 2017).

Finally, regarding the assortment optimization under a non-parametric choice model, Feige et al. (2011) also propose a constraint sampling based method to estimate the non-parametric model from choice observations from consumers. The estimation can be done by maximum likelihood, or norm minimization thanks to the work of van Ryzin and Vulcano (2014, 2017), or a column generation algorithm (Bertsimas and Mišić 2017, Jena et al. 2017). Moreover, once the estimation is performed, the assortment problem under this non-parametric model can be solved conveniently using mixed-integer linear programming (Bertsimas and Mišić 2017).

3. Assortment optimization under parametric choice models

In this section we briefly present some basic concepts of discrete choice modeling and the assortment problem under parametric choice models

3.1. Parametric discrete choice models

The discrete choice framework assumes that each individual (decision maker) n associates an utility u_{ni} with each alternative/option i in a choice set S_n. This utility consists of two parts: a deterministic part v_{ni} that contains observed attributes/features, and a random term ϵ_{ni} that is unknown to the analyst. Different assumptions for the random terms lead to different types of discrete choice models. In general, a linear-in-parameters formula is used, i.e, $v_{ni} = \beta^{\mathrm{T}} a_{ni}$, where "$_{\mathrm{T}}$" is the transpose operator, β is a vector of parameters to be estimated and a_{ni} is the vector of attributes of alternative i as observed by individual n.

The random utility maximization (RUM) framework (McFadden 1978) is the most widely used approach to model discrete choice behaviors. This framework assumes that the decision maker aims at maximizing the utility, so the choice probability that an alternative i in choice set S_n is chosen by individual n is given as

$$P(i|S_n) = P(v_{ni} + \epsilon_{ni} \geq v_{nj} + \epsilon_{nj}, \forall j \in S_n). \tag{1}$$

The MNL model is widely used in this context. This model results from the assumption that the random terms $\epsilon_{ni}, i \in S_n$, are independent and identically distributed (i.i.d.) and follow the standard Gumbel distribution. The corresponding choice probability has the simple form

$$P(i|S_n) = \frac{e^{v_{ni}}}{\sum_{j \in S_n} e^{v_{nj}}}.$$

If the model is linear-in-parameters, the choice probabilities can be written as a function of parameters β as

$$P(i|S_n) = \frac{e^{\beta^{\mathrm{T}} a_{ni}}}{\sum_{j\in S_n} e^{\beta^{\mathrm{T}} a_{nj}}}.$$

It is well known that the MNL model exhibits the IIA property, which implies proportional substitution across alternatives. This property means that for two alternatives, the ratio of the choice probabilities is the same no matter what other alternatives are available or what the attributes of the other alternatives are. In this context we note that if alternatives share unobserved attributes (i.e., random terms are correlated), then the IIA property does not hold.

Coverserly, MMNL is one of the models that allow to relax the IIA property of the MNL model. This model is often used in practice as it is fully flexible in the sense that it can approximate any random utility model (McFadden and Train 2000). In the MMNL model, parameters β are assumed to be random, and the choice probability is obtained by taking the expectation over the random coefficients

$$P(i|S_n) = \mathbb{E}_\beta \left[\frac{e^{\beta^{\mathrm{T}} a_{ni}}}{\sum_{j\in S_n} e^{\beta^{\mathrm{T}} a_{nj}}} \right] = \int \frac{e^{\beta^{\mathrm{T}} a_{ni}}}{\sum_{j\in S_n} e^{\beta^{\mathrm{T}} a_{nj}}} f(\beta) d\beta,$$

where $f(\beta)$ is the density function of β. Then, a Monte Carlo method can be used to approximate the expectation, i.e., if we assume $\beta_1, \ldots, \beta_K$ being K realizations sampled over the distribution of β, then the choice probabilities can be computed as

$$P(i|S_n) \approx \hat{P}_K(i|S_n) = \frac{1}{K} \sum_{k=1}^{K} \frac{e^{\beta_k^{\mathrm{T}} a_{ni}}}{\sum_{j\in S_n} e^{\beta_k^{\mathrm{T}} a_{nj}}}.$$

The MMNL model is preferred in many applications due to its flexibility (McFadden and Train 2000). As mentioned, these models are typically costly to estimate because the estimation requires simulation. In addition, the use of the MMNL also leads to additional complexity to the assortment optimization problem.

The IIA property from the MNL model can also be relaxed by making different assumptions on the random terms $\{\epsilon_1, \ldots, \epsilon_{|S_n|}\}$, resulting in several choice models, e.g., the nested logit (Ben-Akiva 1973), network MEV models (Daly and Bierlaire 2006). In general, such models allow for different ways of modeling the correlation between alternatives. For examples, a nested logit model is appropriate when the set of alternatives can be partitioned into different subsets (called nests), and these subsets are assumed to be disjoint. The cross-nested logit model generalizes the nested one by allowing alternatives to belong in more than one nest. As mentioned, the cross-nested logit model can approximate any RUM models (Fosgerau et al. 2013).

In the following, we present the formulation of the choice probabilities given by a cross-nested logit model of nests (Ben-Akiva and Bierlaire 1999)

$$P(i|S_n) = \sum_{l\in\mathcal{L}} \frac{\left(\sum_{j\in S_n} \alpha_{jl} e^{\mu_l v_{nj}}\right)^{1/\mu_l}}{\sum_{l'\in\mathcal{L}} \left(\sum_{j\in S_n} \alpha_{jl'} e^{\mu_{l'} v_{nj}}\right)^{1/\mu_{l'}}} \frac{\alpha_{il} e^{\mu_l v_{ni}}}{\sum_{j\in S_n} \alpha_{jl} e^{\mu_l v_{nj}}}, \tag{2}$$

where $\mathcal{L}$ is the set of nests, α_{jl} and μ_l, $\forall j \in S_n$, $l \in \mathcal{L}$, are the parameters of the cross-nested model. These parameters have the properties that (i) $\mu_l > 0$, $\forall l \in \mathcal{L}$, and (ii) $\alpha_{jl} > 0$ if alternative j belongs to nest l, and $\alpha_{jl} = 0$ otherwise. If each alternative belongs to only one nest, the model becomes a nested logit model and the choice probability of alternative i in nest l can be written in a simpler form as

$$P(i|S_n) = \frac{e^{\mu_{l_i} v_{ni}} \left(\sum_{j\in \text{nest } l_i} e^{\mu_{l_i} v_{nj}}\right)^{1/\mu_{l_i}-1}}{\sum_{l'\in\mathcal{L}} \left(\sum_{j\in \text{nest } l'} e^{\mu_{l'} v_{nj}}\right)^{1/\mu_{l'}}},$$

where l_i is the nest/subset that contains alternative $i \in S_n$. Note that in this case $\alpha_{il} = 1$ if i in nest l, otherwise $\alpha_{il} = 0$.

The network MEV model generalizes most of existing MEV models including the nested and cross-nested models. This model is generally flexible as it allows to represent the correlation structure between alternatives by a rooted, directed graph where each node without successors is an alternative (see Figure 1). Choice probabilities given by a network MEV model are typically complicated, as their computation requires performing recursive equations or solving dynamic programming problems (Mai et al. 2017).

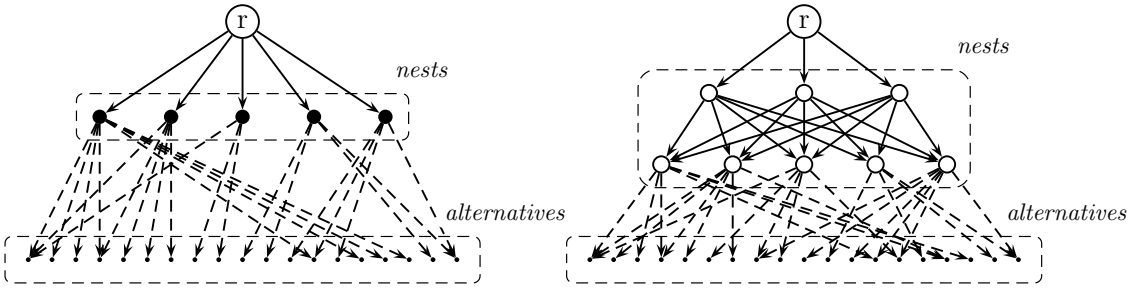

Figure 1 **Two-level and three-levels network MEV structures (Mai et al. 2017)**

3.2. Assortment optimization

Based on the discrete choice framework, we aim at defining a parametric model that can predict the choice behavior of customers in the market. We assume that there are m products available, indexed from 1 to m. There is also the no-purchase alternative (the possibility that the customer does not purchase any of the products in the choice set that is offered), and we denote that alternative by index 0. This no-purchase alternative is always available in any assortment. The entire set of all

possible alternatives now becomes $\mathcal{U} = \{0, 1, \ldots, m\}$. The expected revenue from offering the set of products $S \subset \mathcal{U}$ is denoted by $R(S)$ and is given by

$$R(S) = \sum_{i \in S} r_i P(i|S), \tag{3}$$

where r_i is the revenue of option i. If a linear-in-parameters MNL model is used, then $R(S)$ can be written simply as

$$R(S) = \sum_{i \in S} r_i \frac{e^{\beta^{\mathrm{T}} a_i}}{\sum_{j \in S} e^{\beta^{\mathrm{T}} a_j}},$$

where a_i, $i \in \mathcal{U}$ is the vector of attributes/features of alternative i, and β is the vector of parameter estimates given by the choice model. Note that for notational simplicity, we omit an index for individual n but note that the utilities can be individual specific.

The problem that we wish to solve is to find the set of products $S^* \subseteq \mathcal{U}$, $S^* \supset \{0\}$, that maximizes the expected revenue

$$\max_{\substack{S \subseteq \mathcal{U} \\ S \supset \{0\}}} R(S). \tag{4}$$

Solving the problem typically requires enumerating all the subsets of the full choice set $\mathcal{U}$, which is intractable as the number of subsets is 2^m. The problem can be formulated in an integer optimization form as follows. Let $x_i, i \in \mathcal{U}$ be a binary variable that is 1 if $i \in S$ and 0 otherwise. The problem (4) can be written as follows

$$\max_{\substack{x_i \in \{0,1\} \\ i \in \mathcal{U}}} \sum_{i \in S} r_i P(i|S; S = \{i|x_i = 1\} \cup \{0\}). \tag{5}$$

Under the MNL model, the problem can be formulated into the integer nonlinear model

$$\max_{\substack{x_i \in \{0,1\} \\ i \in \mathcal{U}}} \quad \sum_{i=1}^{m} \frac{r_i x_i V_i}{V_0 + \sum_{j=1}^{m} x_j V_j}, \tag{AO-MNL}$$

where $V_i = e^{\beta^{\mathrm{T}} a_i}$, $i = 0, \ldots, m$. In the case that the MMNL model is used, the assortment optimization is formulated accordingly as

$$\max_{\substack{x_i \in \{0,1\} \\ i \in \mathcal{U}}} \quad \sum_{i=1}^{m} \frac{1}{K} \sum_{k=1}^{K} \frac{r_i x_i V_{ik}}{V_{0k} + \sum_{j=1}^{m} x_j V_{jk}}, \tag{AO-MMNL}$$

where $V_{ik} = e^{\beta_k^{\mathrm{T}} a_i}$, $i = 0, \ldots, m$, and $\beta_1, \ldots, \beta_K$ are K realizations sampled over the randomness of β.

If we use a network MEV model, the problem formulation becomes more complicated, e.g., the integer nonlinear programming model under the nested logit (each product belongs to only one nest) reads as

$$\max_{\substack{x_i \in \{0,1\} \\ i \in \mathcal{U}}} \quad \sum_{i=1}^{m} \frac{x_i V_i^{\mu_{l_i}} \left(\sum_{j \in M_i} x_j V_j^{\mu_{l_i}} \right)^{1/\mu_{l_i} - 1}}{\sum_{l' \in \mathcal{L}} \left(\sum_{j \in (\text{nest } l')} x_j V_j^{\mu_{l'}} \right)^{1/\mu_{l'}}}, \tag{AO-Nested}$$

where $\mathcal{L}$ is the set of nests, l_i is the nest that contains product i, $i = 0, \ldots, m$, and μ_l, $l \in \mathcal{L}$, are the parameters of the nested model. Problem (AO-Nested) is typically difficult to solve exactly, even its continuous relaxation is, as the objective function is nonlinear and highly non-convex. Note that (AO-Nested) becomes (**??**) if $\mu_l = 1$, $\forall l \in \mathcal{L}$.

We do not write out the formulations for the problems under more complicated choice models, e.g., the cross-nested (Vovsha and Bekhor 1998) or network MEV models (Daly and Bierlaire 2006, Mai et al. 2017), but note that they are more complicated than (AO-Nested). In the case of the network MEV model, the choice probability function as well as the expected revenue even have no closed form, and need to be evaluated by recursive equations.

The challenge when solving the assortment optimization problems mentioned above is the nonlinearity and non-convexity of the objective functions. For the MNL and MMNL models, it is possible to formulate the nonlinear problems into MILPs, so we can overcome the non-convexity issue and the problem can be solved by a Brand-and-Bound algorithm (Bront et al. 2009, Méndez-Díaz et al. 2010). However, this approach leads to large MILP models with many additional variables and constraints, making the MILP difficult to solve for large instances. Moreover, for the MEV problem, it is very hard to formulate the nonlinear problems into MILP models.[1] The approach presented in the next section provides a convenient way to deal with such problems.

4. The Algorithmic Framework

In this section, we introduce the binary trust region (BiTR) algorithm, for which the search is driven by the gradient of the objective function. The algorithm is enhanced by a greedy local search that is useful to improve the solutions given by the BiTR. Then, we consider in detail the special case in which the assortment optimization problem is only subject to bound constraints (on the size of the assortment) and we show that the sub-problems that BiTR iteratively solves are solvable in polynomial time. In general, the BiTR algorithm is heuristic but for the case of the MNL with only bound constraints, we show that the overall algorithm becomes an exact method. In addition, thanks to the linear-fractional structure of the MNL-based problem, we show that several steps of the greedy local search algorithm can be skipped.

4.1. Binary Trust Region Algorithm

We first write the assortment problem incorporating linear business constraints as

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & Ax \leq b \\ & x_i \in \{0,1\}, \end{aligned} \tag{AO}$$

[1] The possibility of transforming the problem into a MILP in case of the MNL model is granted by considering assortment constraints that are linear. This is generally the case and few types of simple constraints on the assortment structure will be discussed in Section 4.

where $f(x)$ is the objective function (i.e., expected revenue), and $Ax \leq b$ are some business constraints. We note that these constraints include $x_0 = 1$ to ensure that the non-purchase item is always available in any assortment, and the most popular constraint that can be included is the capacity constraint, i.e., $\sum_i x_i \leq U$ for a constant $1 \leq U \leq m+1$. It is important to note that in the context of parametric choice models, the continuous relaxation of $f(x)$ is continuously differentiable.

Our algorithm is inspired by the trust region method in continuous optimization (Nocedal and Wright 2006). This is an iterative algorithm where, at each iterate, we build a model function and define a region around the current iterate within which we trust the model function to be an adequate representation of the objective function. Then, we find a next iterate by maximizing the model function inside the region that we trust in the hope of finding a new candidate solution with better objective value. The size of the region is reduced or enlarged according to the quality of the new solution found.

We first introduce how to define a model function to approximate the objective function around an iterate x^k, i.e, at iteration k. Since $f(x)$ is continuously differentiable, at a point x the value of $f(x)$ can be expressed as

$$f(x) = f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k + t(x - x^k))(x - x^k),$$

for some scalar $t \in (0, 1)$, where ∇f and $\nabla^2 f$ are the first and second derivatives of function f. So, if we define a model function $m_k(x)$ as

$$m_k(x) = f(x^k) + \nabla f(x^k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k),$$

where B_k is some symmetric matrix, then the difference between $m_k(x)$ and $f(x)$ is $O(||p||^2)$, where $p = x - x^k$, meaning that $m_k(x)$ can be a good approximation of $f(x)$ if $||p||$ is small. Note that if B_k is equal to the true Hessian $\nabla^2 f(x^k)$, then the model function actually agrees with the Taylor series to three terms. In this case the difference is $O(||p||^3)$ and the model function becomes even more accurate when p is small.

We now turn our attention to our assortment problem noting that the problem contains binary variables, leading to the fact that the steps p cannot be too small. Actually, it requires that $p \in \{-1, 0, 1\}^{m+1}$, so $||p|| \geq 1$. So, in this context, the model function m_k cannot give an approximation with any approximation error as in continuous cases, but we expect that the approximation errors are small enough to help us find a better binary candidate solution.

The BiTR algorithm works as follows. At each iteration k with iterate x^k, we define a model function $m_k(x)$ and maximize that function in a region to obtain a new solution $\bar{x}^k$. If $f(\bar{x}^k) > f(x_k)$,

we update $x^{k+1} = \bar{x}^k$, remain or enlarge the trust region, and go to the next iteration, otherwise we keep the current solution, i.e., we set $x^{k+1} = x^k$, and reduced the region. We stop the algorithm when none of the operations result in a strict increase in the revenues. Moreover, because we cannot guarantee that the algorithm converges to an optimal solution, even a local optimum, we can perform a local search in order to check whether we can find a better solution. We describe each step of the algorithm in the following.

To obtain a new solution at each iteration, we seek a solution of the following subproblem:

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & m_k(x) = f(x^k) + \nabla f(x^k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T B_k (x - x_k) && \text{(P1)} \\ \text{subject to} \quad & Ax \leq b && (6) \\ & ||x - x^k|| \leq \Delta_k \\ & x \in \{0,1\}^{m+1}, \end{aligned}$$

where $\Delta_k > 0$ is the trust-region radius at iterate k, and $||x - x^k||$ is a norm of vector $x - x^k$. In our context, we choose the Manhattan norm so the constraints in (P1) can be linearized. Moreover, (P1) is a mixed-integer quadratic problem, which could be expensive to solve. In the case of continuous optimization, the more B_k is close to the Hessian, the more accurate the model function is. It is, however, not the case with integer variables, as the length of step $(x - x^k)$ cannot be arbitrarily small, but needs to be greater than 1. So, in our case, in order to simplify the sub-problem, we just choose $B_k = 0$. So, in summary, we write (P1) as

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \nabla f(x^k)^T x && \text{(P2)} \\ \text{subject to} \quad & Ax \leq b \\ & \sum_{i=0}^{m} |x_i - x_i^k| \leq \Delta_k && (7) \\ & x \in \{0,1\}^{m+1}, \end{aligned}$$

where, in turn, constraint (7) can be linearized as

$$\sum_{\substack{i=0 \\ x_i^k = 1}}^{m} (1 - x_i) + \sum_{\substack{i=0 \\ x_i^k = 0}}^{m} x_i \leq \Delta_k, \tag{8}$$

so as (P2) becomes a mixed-integer linear programming problem.[2]

Once linearized, problem (P2) can be solved by using a commercial solver (e.g., CPLEX, GUROBI) and, typically, after solving it and obtaining a candidate solution $\bar{x}_k$, we can evaluate

[2] Constraint (8) is referred to as local branching constraint by (Fischetti and Lodi 2003).

the solution as well as adjust the trust region radius Δ_k by computing the agreement between the model function m_k and the objective function f at x_k as

$$\rho_k = \frac{f(\bar{x}_k) - f(x_k)}{m_k(\bar{x}_k) - m_k(x_k)} = \frac{f(\bar{x}_k) - f(x_k)}{\nabla f(x^k)^T(\bar{x}_k - x_k)}.$$

We note that since $\bar{x}_k$ is obtained by minimizing the model m_k over a region that includes x_k, the denominator is non-negative. Thus, if $\rho_k < 0$, then $f(\bar{x}_k)$ is less than $f(x_k)$, so the candidate $\bar{x}_k$ must be rejected, and we also need to reduce the Δ_k in the hope of having more accurate model function m_k. On the other hand, if $\rho_k > 0$ then $\bar{x}_k$ is accepted and the radius Δ_k can be expanded to extend the search, or kept equal if ρ_k is not sufficiently large.

We stop the algorithm if after some successive iterations the objective values are not improved. It is easy to see that the Algorithm 1 always terminates after a finite number of iterations. This is due to the fact that there is a finite number of possible solutions and the algorithm only stays at each solution candidate for a limited steps. Moreover, in order to further improve the solution given by the algorithm, we can perform a greedy local search. The idea is to search in a local region around the solution given by the trust region algorithm. In summary, Algorithm 1 describes our binary trust region algorithm.

The following remarks are in order. First, $\Delta_{\max}$ and $\Delta_{\min}$ stand for the maximum and minimum values that the radius of the trust region can take, and they are integer values. Second, in a basic trust region algorithm in continuous optimization, the radius Δ_k is reduced or enlarged by multiplying with scalars (smaller and larger than 1, respectively). In our binary problem we just simply add/remove one unit so that the radius remains integer. Third, we expect that from Step 1 to Step 4, the algorithm performs more quickly as compared to a greedy local search algorithm (e.g., Algorithm 2 or the ADXOpt), as these steps require smaller numbers of function evaluations. And finally, as the last step of Algorithm 1 is a greedy local search algorithm, the final solution given by the BiTR, therefore, inherits some nice properties of the local search, one of them being the global convergence for the case of the MNL model, as will be pointed out in the next section.

Greedy Local Search. We now consider Step 3 of Algorithm 1 in which a greedy local search algorithm, i.e., a search algorithm based on enumerating all the feasible solutions in a local area is run to improve the solution provided by the first part if BiTR. The idea of a local search approach is that we iteratively perform a search over a local region of a solution candidate. The local search around a point x^* can be formulated into the mathematical programming model

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & f(x) \\ \text{subject to} \quad & Ax \leq b \\ & \sum_{i=0}^{m} |x_i - x_i^*| \leq \Delta, \\ & x \in \{0,1\}^{m+1}, \end{aligned} \tag{LS}$$

Algorithm 1: Binary trust region algorithm

 # 1. Initializing

Choose an initial solution x_0, $\Delta_{\max} > \Delta_{\min} \geq 1$, $\Delta_{\min} \leq \Delta_0 \leq \Delta_{\max}$, $\eta > 0$, $k = 0$

 # 2. Iteratively perform the search by solving subproblems

do

  Compute $\bar{x}_k$ by solving subproblem in (P2)

  Compute the agreement $\rho_k = \frac{f(\bar{x}_k) - f(x_k)}{\nabla f(x^k)^T(\bar{x}_k - x_k)}$

  if $\rho_k > 0$ **then**

    $x_{k+1} = \bar{x}_k$ *# We accept the candidate*

    if $\rho_k > \eta$ **then**

      $\Delta_{k+1} = \min\{\Delta_k + 1, \Delta_{\max}\}$ *# We enlarge the trust region radius*

    else

      $\Delta_{k+1} = \Delta_k$ *# We maintain the trust region radius*

  else

    # We keep the current candidate and reduce the trust region radius

    $x_{k+1} = x_k$

    $\Delta_{k+1} = \max\{\Delta_k - 1, \Delta_{\min}\}$

  $k = k + 1$

until *After some successive iterations, the objective $f(x^k)$ is not increased*;

 # 3. Run a greedy local search algorithm from the current candidate x_k to improve it

Execute Algorithm 2.

where Δ is an integer number standing for the radius of the local region that we wish to search. In general, we can choose Δ small, so all the feasible solutions can be enumerated. If we let $A(x)$ denote the assortment given by binary solution x, i.e., $A(x) = \{i | x_i = 1\}$, then if $\Delta = 1$, we can solve (LS) by searching over the set of assortments obtained by adding or removing one item from $A(x^*)$, and for $\Delta = 2$ we can search over the set of assortments obtained by adding/removing one or two items or exchanging an existing item with a new item from the entire choice set. In Algorithm 2 we present a general representation of the local search method for the case $\Delta = 2$.

We have the following remarks in order. First, in Step 1.2, M stands for the set of all assortments satisfying the business constraints, including the constraint that the non-purchase option has to be available in any assortment. Second, Steps 2.1, 2.2 and 2.3 correspond to the case that $\Delta = 1$ and those steps require $O(m)$ evaluations of the expected revenue function. In addition, Steps 2.4 to 2.7 correspond to $\Delta = 2$ and require $O(m^2)$ objective function evaluations. These steps could become time consuming when the number of products is large. Third, the steps from 2.4 to 2.7 can be performed in turn in order to reduce the complexity at each iteration, i.e., we can perform the search on M_k^X first and if we cannot find a better assortment we continue the search on M_k^{2D} and

Algorithm 2: Greedy local search

 # 1. Initializing

1.1. Choose an initial point x^0, let $S_0 = S(x_0)$, $R_0 = f(x_0)$, $k = 0$

1.2. Define $X = \{x | Ax \leq b,\ x \in \{0,1\}^{m+1}\}$, and $M = \{S |\ S = A(x),\ x \in X\}$

 # 2. Greedily perform the search by adding and removing products

repeat

2.1. $M_k^D = \{S |\ S = S_k \backslash \{i\}, i \in S_k \backslash \{0\}\}$ *# Deletion*

2.2. $M_k^A = \{S |\ S = S_k \cup \{i\}, i \notin S_k\}$ *# Addition*

2.3. Select

$$\bar{S} = \text{argmax}_{S \in (M_k^D \cup M_k^A) \cap M} R(S)$$

if $R(\bar{S}) \leq R(S_k)$ **then**

2.4. $M_k^X = \{S |\ S = S_k \backslash \{i\} \cup \{j\}, i \in S_k \backslash \{0\}, j \notin S_k\}$ *# Exchange*

2.5. $M_k^{2D} = \{S |\ S = S_k \backslash \{i,j\}, i,j \in S_k \backslash \{0\},\}$ *# Deletion of two products*

2.6. $M_k^{2A} = \{S |\ S = S_k \cup \{i,j\}, i,j \notin S_k,\}$ *# Addition of two products*

2.7. Select

$$\bar{S} = \text{argmax}_{S \in (M_k^X \cup M_k^{2A} \cup M_k^{2D}) \cap M} R(S)$$

2.8. **if** $R(\bar{S}) > R(S_k)$ **then**

$S_{k+1} = \bar{S}$, $k = k+1$

Until $R(\bar{S}) \leq R(S_k)$;

Return S_k.

then on M_k^{2A}, otherwise we accept the better assortment and go to the next iteration. Fourth, if we remove Steps 2.5 and 2.6, the algorithm becomes similar to the ADXOpt algorithm proposed in Jagabathula (2014), except for the fact that we allow the algorithm to perform the local search under general business constraints (instead of only a cardinality constraint on the number of offered products). Finally, we note that, in general, the local search algorithm requires a search over a large set of assortments, and could be slow if the number of products is large, so it is critical to start from a good starting point x_0, so that less iterations would be required until the algorithm stops. Moreover, in some applications where finding a local optimum is too costly, we can also stop the algorithm when it exceeds a time budget.

4.2. Solving (P2) under bound constraints

In this section, we consider the special case in which the linear constraints (6) are simple bound constraints, i.e., $LB \leq |S| \leq UB$.

We first let S_k and S denote the assortments given by binary vectors x^k, x, respectively, and define function

$$C(S) = \begin{cases} \nabla f(x^k)^T x = \sum_{i \in S} \nabla f(x^k)_i & \text{if } Ax \leq b \\ \mathcal{N} & \text{otherwise,} \end{cases}$$

where $\mathcal{N}$ is a "small enough" number. In fact, $\sum_{i=0}^{m} |x_i - x_i^k| \leq \Delta_k$ for a given $\Delta_k > 0$ is equivalent to the situation that there are at most Δ_k products that either appear in S or in S_k, but not in both. So, the constraint $\sum_{i=0}^{m} |x_i - x_i^k| \leq \Delta_k$ can be reformulated in the equivalent form $|S \triangle S_k| \leq \Delta_k$, where $\triangle$ is the symmetric difference operator, i.e., $S \triangle S_k = (S \backslash S_k) \cup (S_k \backslash S)$. So under a bound constraint on the size of the assortment, (P2) can be written equivalently in another form as

$$\begin{aligned} \underset{S \subset \mathcal{U}}{\text{maximize}} \quad & C(S) \\ \text{subject to} \quad & LB \leq |S| \leq UB \\ & |S \triangle S_k| \leq \Delta_k \\ & S \supset \{0\}, \end{aligned} \tag{P4}$$

We also remark that, given set S_k, a set S such that $|S \triangle S_k| \leq \Delta_k$ can be obtained by performing at most Δ_k operations of adding new products or removing available products to/from S_k. For the sake of illustration, in Figure 2 we also show an example of S and S_k with $\Delta_k = 3$. In this example, we can obtain S by removing two products (on the left of of the figure) from S_k and add one new product (on the right of the figure). The algorithm described in the following is based on this remark. More precisely, we can perform the search by adding and/or removing products from S_k under the condition that the number of operations does not exceed Δ_k, and benefiting from the fact that the objective is an affine function.

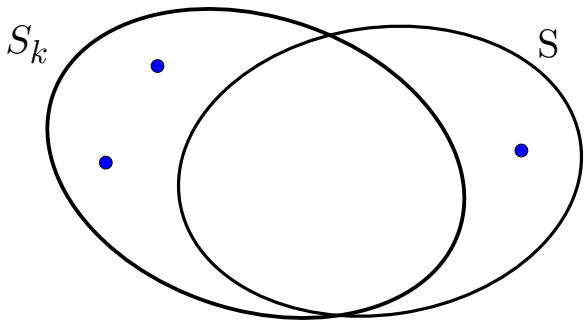

Figure 2 **An illustration of S and S_k when $\Delta_k = 3$.**

Algorithm 3: Solving sub-problem (P4)

Begin

1. Take a number of largest elements of arrays $\{-\nabla f(x^k)_i | i \in S_k \backslash \{0\}$, and $\{\nabla f(x^k)_i | i \in \mathcal{U} \backslash S_k$

$$-\nabla f(x^k)_{\sigma_1} \geq -\nabla f(x^k)_{\sigma_2} \geq ... \geq -\nabla f(x^k)_{\sigma_{\min\{\Delta_k, |S_k|-1\}}}$$
$$\nabla f(x^k)_{\pi_1} \geq -\nabla f(x^k)_{\pi_2} \geq ... \geq \nabla f(x^k)_{\pi_{\min\{\Delta_k, m+1-|S_k|\}}}$$

where $\sigma_1, \dots \sigma_{\min\{\Delta_k, |S_k|-1\}} \in S_k \backslash \{0\}$ and $\pi_1, \dots, \pi_{\min\{\Delta_k, m+1-|S_k|\}} \in \mathcal{U} \backslash S_k$

2. Define a function $\mathcal{T} : [0, 1, ..., \Delta_k] \times [0, 1, ..., \Delta_k] \to \mathbb{R}$

$$\mathcal{T}(v,d) = \begin{cases} \mathcal{N} & \text{if } v + d > \Delta_k, \text{or } v > |S_k| - 1, \text{ or } d > m + 1 - |S_k| \\ \mathcal{N} & \text{if } |S_k| + v - d \notin [LB, UB] \\ \sum_{j=1}^{d} \nabla f(x^k)_{\pi_j} - \sum_{i=1}^{v} \nabla f(x^k)_{\sigma_i} & \text{otherwise,} \end{cases}$$

where $\mathcal{N}$ is a "very small" number.

3. Select $(v^*, d^*) = \text{argmax}_{0 \leq v, d \leq \Delta_k} \mathcal{T}(v,d)$, and return

$$S^* = S_k + \pi_1 + \ldots + \pi_{v^*} - \sigma_1 - \ldots - \sigma_{d^*}.$$

It is possible to prove that the running time of Algorithm 3 is polynomial with respect to m and Δ_k. Moreover, a solution given by Algorithm 3 is also an optimal solution to (P4) (Theorem 1). We start the proof by introducing the following lemma.

LEMMA 1. *At iterate k, the function $\mathcal{T}$ defined in the algorithm satisfies*

$$C(S_k) + \mathcal{T}(v,d) \geq \max_{S \supset \{0\}} \{C(S) |\ |S \backslash S_k| = v,\ |S_k \backslash S| = d\},$$

For all v, d such that $0 \leq v \leq \min(\Delta_k, |S_k| - 1)$, and $0 \leq d \leq \min(\Delta_k, m + 1 - |S_k|)$, and $|S_k| + v - d \in [LB, UB]$.

Proof of Lemma 1. Indeed, if a set S such that $|S \backslash S_k| = v$ and $|S_k \backslash S| = d$, meaning that there exit products $i_1, \dots, i_d \in S_k \backslash \{0\}$ and $j_1, \dots, j_v \in \mathcal{U} \backslash S_k$ such that

$$S = S_k - i_1 - \ldots i_d + j_1 + \ldots j_d.$$

The capacity of S is $|S_k| + v - d$. So under the assumption of the lemma and the definition of $C(S)$, the expected revenue given by S is

$$C(S) = C(S_k) - \nabla f(x^k)_{i_1} - \ldots - \nabla f(x^k)_{i_d} + \nabla f(x^k)_{j_1} + \ldots + \nabla f(x^k)_{j_v}.$$

And because $-\nabla f(x^k)_{i_1},\ldots,-\nabla f(x^k)_{i_d}$ and $\nabla f(x^k)_{j_1},\ldots-\nabla f(x^k)_{j_v}$ are the d and v largest elements in $\{-\nabla f(x^k)_i | i\in S_k\backslash\{0\}\}$, and $\{\nabla f(x^k)_i | i\in\mathcal{U}\backslash S_k\}$, respectively, so we have

$$C(S)\leq C(S_k)-\nabla f(x^k)_{\sigma_1}-\ldots-\nabla f(x^k)_{\sigma_d}+\nabla f(x^k)_{\pi_1}+\ldots-\nabla f(x^k)_{\pi_v}=C(S_k)+\mathcal{T}(v,d).$$

Q.E.D.

The following theorem shows the convergence as well as the complexity of Algorithm 3.

THEOREM 1. *Algorithm 3 returns an optimal solution to* (P4) *and the complexity is* $O(\Delta_k m+\Delta_k^2)$.

Proof of Theorem 1. This is obvious to verify that Step 1 runs in $O(\Delta_k m)$ time and Steps 2 and 3 runs in $O(\Delta_k^2)$ time. So, in total the running time of Algorithm 3 is $O(\Delta_k m+\Delta_k^2)$. Note that most of the cases Δ_k is much smaller than m and the complexity can be approximated by $O(m)$.

Now we prove that the assortment S^* returned by Algorithm 3 is an optimal solution to (P4). In order to do so, we just need to verify that

$$C(S^*)\geq\max_{S\supset\{0\}}\{C(S)|\,|S\triangle S_k|\leq\Delta_k,\,LB\leq|S|\leq UB\}.$$

From Step 3 we have

$$\begin{aligned}C(S^*)&=C(S_k)+\mathcal{T}(v^*,d^*)\\&=C(S_k)+\max\{\mathcal{T}(v,d)|\,0\leq v,d\leq\Delta_k\}\\&=\max_{\substack{0\leq v\leq\min(\Delta_k,|S_k|-1)\\0\leq d\leq\min(\Delta_k,m+1-|S_k|)\\|S_k|+v-d\in[LB,UB]}}\{C(S_k)+\mathcal{T}(v,d)\}.\end{aligned}$$

So, according to Lemma 1

$$\begin{aligned}C(S^*)&\geq\max_{\substack{0\leq v\leq\min(\Delta_k,|S_k|-1)\\0\leq d\leq\min(\Delta_k,m+1-|S_k|)\\|S_k|+v-d\in[LB,UB]\\S\supset\{0\}}}\Big\{C(S)\Big|\,|S\backslash S_k|=v,\,|S_k\backslash S|=d\Big\}\\&\geq\max_{\substack{|S|\in[LB,UB]\\S\supset\{0\}}}\Big\{C(S)\Big|\,|S\triangle S_k|\leq\Delta_k\Big\}.\end{aligned}\tag{9}$$

The last equation is due to that fact that

$$\left\{S\subset\mathcal{U}\Big|\exists v,d\text{ such that }\begin{cases}0\leq v,d\leq\Delta_k\\v\leq|S_k|-1\\d\leq m+1-|S_k|\\|S\backslash S_k=v\\S_k\backslash S|=d\end{cases}\right\}\equiv\Big\{S\Big|S\subset\mathcal{U},\,|S\triangle S_k|\leq\Delta_k\Big\}.$$

Finally, (9) indicates that S^* is an optimal solution to (P4). This completes the proof. Q.E.D.

The special case of the MNL model. For the problem under the MNL model, due the fact that the objective function is simply a function of linear functions, it is possible to prove some results that guarantee the convergence of the local search algorithm, as well as allow to improve Algorithm 2. More precisely, Jagabathula (2014) shows that under the MNL and an upper bound constraint ($|S| \leq UB$), the ADXOpt can return an optimal solution. In addition to his results, we consider the MNL problem with a bound constraint ($LB \leq |S| \leq UB$), and we can show that Algorithm 2 has some interesting properties that not only guarantee the convergence to an optimal solution, but also gives some suggestions to remove unnecessary steps to further speed up the local search algorithm. We present these properties through the following series of propositions and we refer the reader to Appendix for the proofs.

PROPOSITION 1. *Under the MNL model and a bound constraint $LB \leq |S| \leq UB$, the solution given by Algorithm 2 is optimal.*

PROPOSITION 2. *Under the MNL model and a bound constraint $LB \leq |S| \leq UB$, at iteration k, if Step 2.3 of Algorithm 2 returns an assortment $\bar{S}$ such that $R(S_k) \geq R(\bar{S})$ and $LB < |S_k| < UB$, then S_k is an optimal solution and the algorithm can be terminated.*

REMARK 1. Proposition 2 indicates that after performing the search by adding and removing one products, if we obtain a solution that is strictly in the bound, then that is an optimal solution and we do not need to perform Steps 2.4 - 2.7 of Algorithm 2. Moreover, we can show, by a similar way, that for an unconstrained problem, after Step 2.3 if $R(S_k) \geq R(\bar{S})$, then S_k an optimal solution. To prove this, we only need to consider the case that $|S_k| = m+1$ (the case that S_k only contains non-purchase product is not reasonable). In this case S_k contains all the products, and we also can show that $A(-V_i) \geq B(-V_i r_i), \forall i \in \mathcal{U}$. Now for any other assortment $S \subset \mathcal{U}$, we always can obtain S by removing some products from S_k. According to the above inequality we also can prove easily that $R(S) \leq R(S^*)$. This remark is also consistent with those found by Jagabathula (2014).

PROPOSITION 3. *Under the MNL model and a bound constraint $LB \leq |S| \leq UB$, at an iteration k, if $R(S_k) \geq R(S), \forall S \in M_k^D \cup M_k^A$, then $R(S_k) \geq R(S), \forall S \in M_k^{2D} \cup M_k^{2A}$.*

REMARK 2. The results from Proposition 3 simply indicate that under the MNL model and a bound constraint, Steps 2.4 and 2.5 can be safety removed from the algorithm. In other words, after Step 2.3 if we cannot find a better assortment, we can continue the search by performing only the "exchange" operation (i.e., Step 2.4). In this context Algorithm 2 is similar to ADXOpt.

PROPOSITION 4. *Under the MNL model, at iteration k, a product $j \notin S_k$ can be added to S_k if $R(S_k) < r_j$, and a product $i \in S_k \setminus \{0\}$ can be removed from S_k if $R(S_k) > r_j$.*

REMARK 3. These results can be useful for reducing the computing cost. More precisely, at iteration k we can check if $R(S_k) \geq \max_{j \notin S_k} r_j$ then the step "addition" can be skipped, and if $R(S_k) \leq \min_{i \in S_k \setminus \{0\}} r_i$ then the step "deletion" can be skipped. In addition, if $R(S_k) \geq \max_{j \notin S_k} r_j$ and $R(S_k) \leq \min_{i \in S_k \setminus \{0\}} r_i$, then if $|S_k|$ is strictly in the bound, $|S_k|$ is an optimal solution (Proposition 2), otherwise we continue the search with the "exchange" operation. As a result, Steps 2.1-2.2 of Algorithm 2 can be modified to reduce the sizes of $|M_k^D|$ and $|M_k^D|$ as follows: $M_k^D = \{S |\, S = S_k \setminus \{i\}, i \in S_k \setminus \{0\}, R(S_k) > r_i\}$ and $M_k^A = \{S |\, S = S_k \cup \{i\}, i \notin S_k\}, R(S_k) < r_j\}$.

These results also directly lead to the optimality of revenue-ordered subsets for uncapacitated MNL (Talluri and Van Ryzin 2004). More precisely, the above results indicate that the optimal assortment S^* should satisfy the inequality $\min_{i \in S^* \setminus \{0\}} r_i \geq R(S^*) \geq \max_{j \in \mathcal{U} \setminus S^*} r_j$, which simply leads to the optimality of the revenue-ordered subsets.

PROPOSITION 5. *Under the MNL model and a bound constraint $LB \leq |S| \leq UB$, if at an iteration k we have $S_k = LB$, and after Step 2.3 of Algorithm 2 $R(\bar{S}) \leq R(S_k)$, then from the next iteration only steps of type "exchange" can provide better assortments.*

REMARK 4. Proposition 5 indicates that if the algorithm reaches to an *exchange* step and if the size of the assortment is LB, then we should keep exchanging products to find an optimal solution (steps "deletion" and "addition" can be skipped), and the size of the optimal assortment is $|S^*| = LB$. We also note that it is not the same for the case of $|S_k| = UB$. Indeed, if $|S_k| = UB$, then the condition that no product should be removed from S_k is that $R(S_k) \leq \min_{i \in S_k \setminus \{0\}} r_i$. If a product $i \in S_k \setminus \{0\}$ is exchanged with $j \notin S_k$, then the new expected revenue is $R(S_{k+1}) = \frac{A - V_i r_i + V_j r_j}{B - V_i + V_j}$. And we also have the fact that $\lim_{r_j \to \infty} R(S_{k+1}) = \infty$, so if there is a product j with "large enough" revenue, the condition $R(S_{k+1}) \leq \min_{i \in S_{k+1} \setminus \{0\}} r_i$ may be violated, meaning that at iteration k we can probably remove a product to get a better assortment.

5. Numerical studies

In this section, we report the results of our computational experiments performed to assess the effectiveness of the BiTR algorithm on different problem instances.

5.1. Data and models

We illustrate the performance of the approach using a real data set of the sales of a major US shoes retailer. The data set was provided by the JDA Software (https://jda.com/), a company developing software for the retail industry. There are 1053 different products across the whole period. Each item is characterized by a set of different features, i.e., class, sub-class, brand, material and color. We use a data set collected from the week 35th to 52nd of the year 2013 in 229 stores across the U.S. There are 3,565 assortments given to the customers and there are 134,320

transactions/observations recorded. The number of products in the assortments vary between 43 and 162.

There are several products' features that can be taken into account, e.g, price, item class, item material and item color. Some features take real positive values, (e.g., price), and some take discrete values (e.g., item color, item class). We build choice models based on these features and note that discrete-value features are included into the models using binary attributes. For example, there is an attribute a referring to the red color, so for any item, its a attribute takes value 1 if the item is red, and 0 otherwise. In total, there are 111 binary attributes.

We specify MNL, MMNL and network MEV models for the experiment using the above attributes. These models are estimated/trained using maximum likelihood estimation. We do not present the estimation results because it is out of scope of this paper, but we note that the network MEV model estimation is more computationally expensive as compared to the MNL and MMNL models, and we use the techniques from Mai et al. (2017) to speed up the network MEV model estimation. Moreover, we observe that the network MEV model performs better than the others in terms of in- and out-of-sample fits.

In these experiments, we compare our BiTR with the ADXOpt proposed in Jagabathula (2014), as it is the only general method that can deal with instances under the three choice models above. The both algorithms are implemented in MATLAB to have a fair comparison. Moreover, Jagabathula (2014) shows that his algorithm performs relatively better as compared to other existing heuristic approaches. For the MNL and MMNL problems, due to the fact that it is possible to formulate the optimization problems into MILP models and solve them using a commercial solver, we also compare our algorithm with the MILP approach proposed in Bront et al. (2009).

Finally, before presenting our experimental results for instances under the three choice models above, we note that the codes for estimating the discrete choice models are implemented in MATLAB, and we have used an Intel(R) 3.20 GHz machine with a x64-based processor. It is running in the 64-bit Window 10 Operating System. The machine has a multi-core processor but we only use one processor to estimate the models as the code is not parallelized. For maximizing the log-likelihood we use the limited memory BFGS algorithm (L-BFGS) (see for instance Nocedal and Wright 2006, Chapter 9).

5.2. Case study 1: Multinomial logit - MNL

We test different methods when the choice model is the MNL. In this context, the assortment optimization problem (AO-MNL) (see, Section 3.2) is a 0-1 linear fractional programming model, and it is well known that it is possible to formulate a 0-1 linear fractional programming model into an equivalent MILP model (Wu 1997). More precisely, this can be done by defining variables

$$y = \frac{1}{V_0 + \sum_{i=1}^{m} x_i V_i},$$

and (AO-MNL) can be rewritten as

$$\begin{aligned} \underset{x,y}{\text{maximize}} \quad & \sum_{i=1}^{m} r_i V_i x_i y \\ \text{subject to} \quad & V_0 y + \sum_{i=1}^{m} V_i x_i y = 1 \\ & Ax \leq b \\ & x_i \in \{0,1\}\, i = 1,\dots,m, \\ & y \geq 0. \end{aligned} \tag{10}$$

The nonlinear term $x_i y$ can be linearized by defining new continuous variables $z_i = x_i y$, $i = 1,\dots,m$. Since y is continuous, and $x_i, \forall i$, are 0-1 variables, Wu (1997) suggests that z_i can be included in the model using the following inequalities: (i) $y - z_i \leq H - Hx_i$, (ii) $z_i \leq y$ and (iii) $z_i \leq Hx_i$, for $i = 1,\dots,m$, where H is a large positive value that defines a valid upper bound for y. In this context, it is enough to choose $H = 1/e^{v_0^*}$, and the constraints $z_i \leq Hx_i$ can be tightened by $(e^{v_0^*} + e^{v_i^*})z_i \leq x_i$, $i = 1,\dots,m$ (Méndez-Díaz et al. 2010). So, we obtain the following MILP formulation for (AO-MNL):

$$\begin{aligned} \underset{x,y}{\text{maximize}} \quad & \sum_{i=1}^{m} r_i V_i z_i \\ \text{subject to} \quad & Ax \leq b \\ & V_0 y + \sum_{i=1}^{m} V_i z_i = 1 \\ & V_0 y - V_0 z_i \leq 1 - x_i,\ i = 1,\dots,m \\ & z_i \leq y,\ i = 1,\dots,m \\ & (V_0 + V_i) z_i \leq x_i,\ i = 1,\dots m \\ & x_i \in \{0,1\},\ z_i \geq 0,\ i = 1,\dots,m \\ & y \geq 0. \end{aligned} \tag{11}$$

We compare the performance of the BiTR, ADXOpt and MILP approaches. We consider in this experiment three types of feasible sets, i.e., no-constraints, capacity constraints ($0 \leq |S| \leq UB$) and bound constraints ($LB \leq |S| \leq UB$). Note that the ADXOpt presented in Jagabathula (2014) cannot directly handle bound constraints, so we use the extended version in Algorithm 2, where the initial points are chosen arbitrarily in the feasible set X. Moreover, it is well known that the unconstrained MNL problem can be efficiently solved by only considering *revenue-ordered* subsets of products (Talluri and Van Ryzin 2004). So, for the unconstrained case, we also report the computing time for the unconstrained MNL instances with the revenue-order (RO) approach. Note that, as proven in Section 4.2, for the MNL case BiTR is an exact method in case of the classes of

constraints considered in these experiments. This is true for ADXOpt too for unconstrained and capacity constraints only.

In this experiment, we choose a time budget of 600 seconds, meaning that when an approach exceeds the time budget, we stop and report the best objective value found. For each instance and each method, we report the computing time as well as the percentage gap between the corresponding objective value and the best one found by the three approaches, e.g., the percentage gap associated with the ADXOpt is computed as follows

$$\%Gap = \frac{Best\ value - Value\ found\ by\ ADXOpt}{Best\ value} \times 100.$$

Table 1 reports the computing time when solving the MNL instances using the MILP (via the MILP solver CPLEX), ADXOpt, BiTR and RO approaches, where the symbol "-" indicates that the approach exceeds the time budget of 600 seconds, and in the cases that the objectives are not the optimal, we report the percentage gaps in parentheses. Bold numbers indicate the best performance. The RO is, expectedly, the fastest approach to solve unconstrained instances. The BiTR is slower than the RO, but the differences, in terms of computing time, are small. It is also clear that the BiTR dominates the MILP and ADXOpt approaches in terms of both computing time and solution quality. For the MILP approach, even though there are 15/24 instances that CPLEX cannot prove optimality within the time budget, all the solutions returned are optimal. The ADXOpt algorithm is generally faster than the MILP, but there are 6/24 instances where the ADXOpt cannot find optimal solutions. It is important to note that all the solutions given by the BiTR without the "local search step" (i.e., Algorithm 1 without Step # 3) are also optimal. In other words, the solutions obtained after Step #2 of Algorithm 1 are optimal, and in Step #3 the algorithm only needs to check the optimality of the solutions using the properties presented in Section 4 above.

In order to provide a view of the performance of the BiTR and ADXOpt approaches, we take the instances of 1,000 products and plot in Figure 3 the computing times and objective values over iterations. It is clear that the BiTR converges remarkably faster to the optimal solution compared to the ADXOpt. For unconstrained and capacity instances, the ADXOpt manages to find optimal solutions within the time budget, but it is not the case with bound constraints (i.e, $300 \leq |S| \leq 500$, and $650 \leq |S| \leq 750$).

5.3. Case study 2: Mixed logit - MMNL (random parameters logit)

In this section, we report the computing results with MMNL instances. We assume that the price sensitivity parameter β_p is no-longer deterministic, but follows a normal distribution, i.e., $\beta_p \sim N(\beta_p^0, \sigma_p)$, and β_p^0, σ_p are model parameters to be estimated. The model parameters can be obtained

m	Constraints	MILP	ADXOpt	BiTR	RO
100	-	0.2	0.52	0.12	**0.03**
	$\|S\| \leq 50$	0.3	0.60	**0.12**	
	$30 \leq \|S\| \leq 50$	0.4	0.56	**0.13**	
	$50 \leq \|S\| \leq 70$	0.4	1.72	**0.14**	
200	-	0.4	1.69	0.14	**0.06**
	$\|S\| \leq 100$	-	2.16	**0.14**	
	$70 \leq \|S\| \leq 100$	0.8	7.41	**0.21**	
	$120 \leq \|S\| \leq 160$	1.0	30.81	**0.22**	
400	-	-	6.76	0.20	**0.13**
	$\|S\| \leq 20$	-	6.76	**0.21**	
	$100 \leq \|S\| \leq 200$	1.4	52.92	**0.35**	
	$250 \leq \|S\| \leq 350$	18.4	246.72	**0.45**	
600	-	-	13.39	0.31	**0.21**
	$\|S\| \leq 300$	-	13.56	**0.30**	
	$200 \leq \|S\| \leq 300$	-	-(0.05)	**0.69**	
	$450 \leq \|S\| \leq 550$	-	-(0.02)	**0.60**	
800	-	-	24.08	0.54	**0.31**
	$\|S\| \leq 400$	-	25.13	**0.55**	
	$300 \leq \|S\| \leq 400$	-	-(1.38)	**1.46**	
	$550 \leq \|S\| \leq 650$	-	-(1.64)	**1.18**	
1,000	-	-	35.58	0.65	**0.42**
	$\|S\| \leq 500$	-	35.68	**0.65**	
	$300 \leq \|S\| \leq 500$	-	-(1.37)	**1.67**	
	$650 \leq \|S\| \leq 750$	-	-(2.41)	**1.68**	

Table 1 **Computing time (in seconds) and percentage gaps (%) for the MNL instances.**

via maximum likelihood estimation. However, the estimation is out-of-scope of this experiment, so we just fixed those parameters and use them for testing the performance of our algorithm.

In the case of the problem with the MMNL model, an equivalent MILP formulation can also be obtained (Méndez-Díaz et al. 2010). More precisely, we can define

$$y_k = \frac{1}{V_{0k} + \sum_{i=1}^{m} x_i V_{ik}}, \forall k, \text{ and } z_{ik} = x_i y_k, \forall i, k,$$

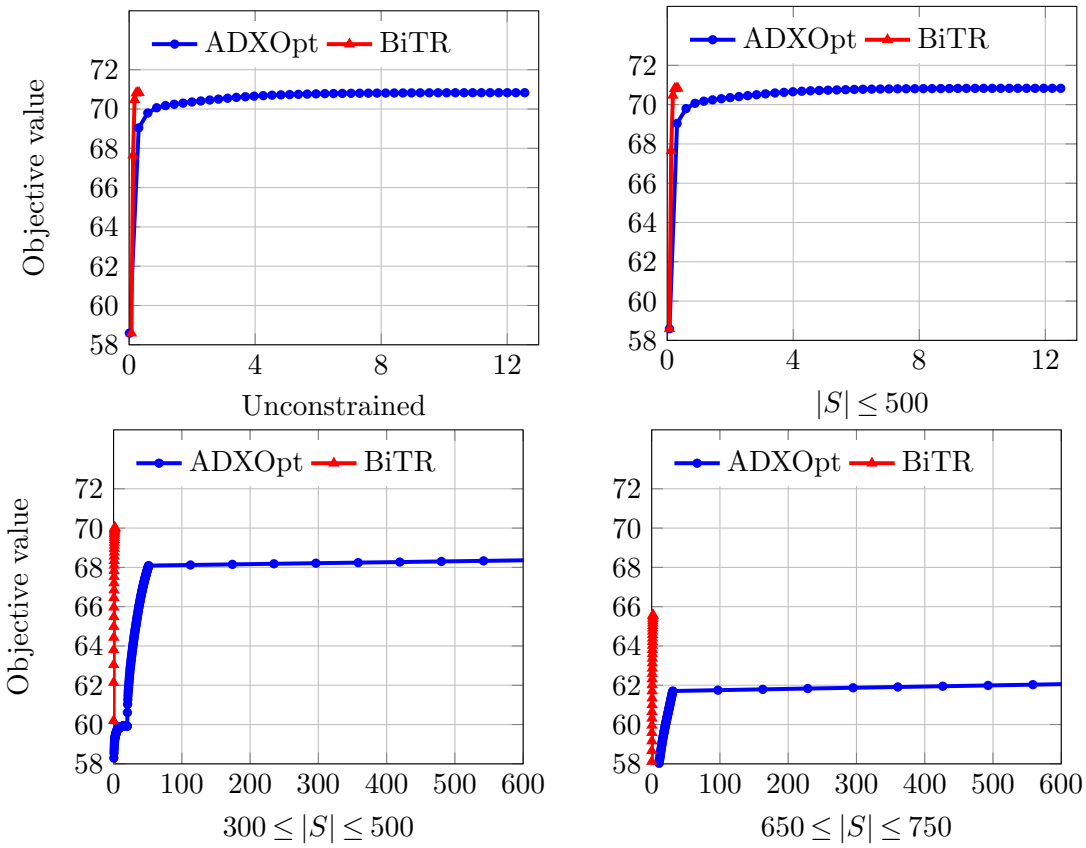

Figure 3 **Computing time and objective values found, given by the BiTR and ADXOpt for the MNL problem with 1,000 products, and given a time budget of 600 seconds.**

then (AO-MMNL) can be reformulated in a linear 0-1 form as

$$
\begin{aligned}
\underset{x,y}{\text{maximize}} \quad & \frac{1}{K}\sum_{i=1}^{m}\sum_{k=1}^{K} r_i V_{ik} z_{ik} \\
\text{subject to} \quad & Ax \leq b \\
& V_{ok} y_k + \sum_{i=1}^{m} V_{ik} z_{ik} = 1, \ \forall k \\
& V_{0k} y_k - V_{0k} z_{ik} \leq 1 - x_i, \ \forall i,k \\
& z_{ik} \leq y_k, \ \forall i,k \\
& (V_{0k} + V_{ik}) z_{ik} \leq x_i, \ \forall i,k \\
& x_i \in \{0,1\}, \ z_{ik} \geq 0, \ \forall i,k, \\
& y_k \geq 0, \ \forall k.
\end{aligned} \tag{12}
$$

The model (12) consists of M binary and $K(M+1)$ continuous variables, and $3MK+K$ constraints. So, the size of this model increases proportionally with the number of products M and number of draws K, leading to the fact that the model may be difficult to solve for large-scale instances (e.g., instances of thousands of products).

We generate samples of $K = 100, 200$ and 500 for the experiment. In this case study, due to the large number of products and the complexity of the objective function, the cost to perform Steps 2.4 - 2.7 of Algorithm 1 and the "exchange" step of ADXOpt are expensive, we therefore only use the steps of adding or removing one item for both the ADXOpt and the local search of Algorithm 1. Table 2 reports the numerical results for the MILP, ADXOpt, BiTR and the BiTR without the "local search" step (BiTR-noLS) (i.e., Algorithm 1 without Step #3), where bold numbers indicate best methods, i.e., return optimal solutions in shortest computing time. Similar to the MNL case, the "-" indicates that CPLEX fails to return an optimal solution within a time budget of 600 seconds. For each instance and method, if the objective value found is not the best one, we report in parentheses the percentage gaps with respect to the best solution found by all methods.

The results in Table 2 clearly show that the BiTR approach is very competitive. On the one hand, it clearly outperforms ADXOpt as a heuristic algorithm. On the other hand, BiTR provides solutions whose quality is way better than the one of CPLEX solving the MILP, and faster. Of course, one needs to recall that CPLEX while solving the MILP formulation is designed to prove optimality, so the comparison is only in terms of practical use of the approaches. Finally, the BiTR-noLS is the fastest algorithm, and it is interesting to see that the approach manages to return best objective values for 54/72 instances, and for the others the percentage gaps are also small.

We also report the computing time and objective values over iterations for the BiTR and ADX-Opt approaches, in order to see how the two approaches converge to solutions. Similar to the MNL case, we take the instances of 1,000 products with $K = 500$ (the largest instances). Figure 4 reports the computing time and objective value for the four types of feasible sets. Clearly, the BiTR converges to the best solution quickly as compared to the ADXOpt. For unconstrained and capacitated instances, ADXOpt manages to find good solutions within 600 seconds, but it is not the case for the instances with bound constraints.

Given that fact that the ADXOpt and BiTR algorithms are heuristic, we also test the three approaches on small-size instances in order to validate the quality of the solutions found. Note that for these small instances, we do not remove step "exchange" from ADXOpt and Steps 2.4 - 2.7 from Algorithm 2, like we did instead for the large instances considered above. For these instances, the MILP approach is able to return optimal solutions, so we use the optimal values given by the MILP approach to evaluate the solutions given by the ADXOpt and BiTR. Table 3 reports the results based on instances of 10, 20 and 30 products, where bold numbers indicate the fastest methods. Interestingly, all the approaches are able to find optimal solutions for all the instances. The ADXOpt approach is the fastest one, and the computing times of the MILP approach start to be remarkably larger than those required by the two other approaches for $m > 20$. It is interesting to note that the ADXOpt is faster than the BiTR for small-size instances. This can be explained

	K	100				200				500			
m	Constraints	MILP	ADXOpt	BiTR	BiTR-noLS	MILP	ADXOpt	BiTR	BiTR-noLS	MILP	ADXOpt	BiTR	BiTR-noLS
100	-	-(10.1)	0.8	0.6	**0.3**	-(12.0)	0.8	0.6	**0.3**	-(12.8)	1.5	**0.7**	0.2 (0.02)
	$\|S\| \leq 50$	-(7.9)	0.5	0.5	**0.2**	-(15.1)	0.8	0.6	**0.3**	-(15.5)	1.5	**0.7**	0.2 (0.02)
	$30 \leq \|S\| \leq 50$	-(4.2)	0.7 (0.4)	0.5	**0.2**	-(6.8)	1.2 (0.5)	0.5	**0.2**	-(14.8)	2.5 (0.6)	0.6	**0.3**
	$50 \leq \|S\| \leq 70$	-(2.0)	0.5 (1.7)	0.5	**0.2**	-(7.8)	0.8 (1.7)	0.5	**0.2**	-(10.3)	1.7 (1.8)	0.7	**0.3**
200	-	-(18.2)	1.6	**0.6**	0.1 (0.1)	-(17.9)	3.0	0.6	**0.2**	-(18.2)	6.0	1.0	**0.4**
	$\|S\| \leq 100$	-(21.7)	1.6	**0.6**	0.1 (0.1)	-(21.7)	3.0	0.6	**0.2**	-(21.5)	6.0	1.1	**0.5**
	$70 \leq \|S\| \leq 100$	-(16.1)	2.6 (4.9)	0.6	**0.3**	-(19.1)	4.6 (5.5)	0.7	**0.3**	-(18.8)	9.7 (5.8)	0.9	**0.4**
	$120 \leq \|S\| \leq 160$	-(7.7)	1.6 (4.7)	0.6	**0.3**	-(10.3)	2.7 (5.1)	0.6	**0.3**	-(10.4)	4.9 (5.6)	0.8	**0.4**
400	-	-(20.8)	6.5	0.9	**0.4**	-(20.7)	10.9	1.2	**0.5**	-(20.7)	24.7	**4.3**	0.3 (0.03)
	$\|S\| \leq 20$	-(24.6)	6.4	0.9	**0.4**	-(23.7)	11.2	1.3	**0.5**	-(24.1)	24.6	**4.2**	0.3 (0.03)
	$100 \leq \|S\| \leq 200$	-(23.1)	22.4 (5.3)	0.9	**0.4**	-(22.3)	38.7 (5.7)	1.1	**0.5**	-(22.7)	83.3 (5.8)	1.7	**0.7**
	$250 \leq \|S\| \leq 350$	-(9.6)	14.0 (4.5)	0.9	**0.5**	-(10.3)	23.5 (4.4)	1.1	**0.6**	-(10.6)	53.5 (4.4)	1.4	**0.7**
600	-	-(24.1)	13.7	**1.8**	0.6 (0.1)	-(25.6)	20.6	**4.0**	0.3 (0.1)	-(25.7)	49.2	4.6	**0.4**
	$\|S\| \leq 300$	-(25.0)	13.7	**1.8**	0.6 (0.1)	-(26.1)	20.2	**3.9**	0.3 (0.1)	-(26.6)	48.7	4.6	**0.5**
	$200 \leq \|S\| \leq 300$	-(19.5)	47.9 (7.1)	1.6	**0.9**	-(20.9)	83.6 (7.2)	1.9	**1.0**	-(21.2)	183.6 (7.1)	2.8	**1.2**
	$450 \leq \|S\| \leq 550$	-(5.0)	14.0 (4.4)	1.1	**0.7**	-(8.1)	24.6 (4.3)	1.3	**0.8**	-(8.1)	53.1 (4.4)	1.8	**1.0**
800	-	-(25.7)	24.7	2.2	**1.0**	-(26.8)	45.4	**6.6**	0.5 (0.1)	-(26.9)	96.7	**7.5**	0.7 (0.2)
	$\|S\| \leq 400$	-(27.0)	24.9	2.2	**0.9**	-(27.6)	45.4	**6.6**	0.5 (0.1)	-(28.1)	96.1	**7.6**	0.7 (0.2)
	$300 \leq \|S\| \leq 400$	-(19.5)	112.2 (6.3)	2.4	**1.6**	-(20.5)	204.7 (6.4)	2.9	**1.7**	-(20.9)	427.8 (6.5)	4.2	**2.0**
	$550 \leq \|S\| \leq 650$	-(7.4)	55.2 (4.5)	1.9	**1.4**	-(10.0)	101.0 (4.4)	2.4	**1.6**	-(10.1)	230.2 (4.4)	3.1	**1.8**
1,000	-	-(27.4)	45.7	**6.8**	0.6 (0.3)	-(28.4)	78.5	7.5	**0.6**	-(29.0)	166.2	**10.8**	0.7 (0.0)
	$\|S\| \leq 500$	-(29.3)	45.9	**6.9**	0.6 (0.3)	-(29.5)	79.2	7.4	**0.6**	-(30.0)	164.8	**10.6**	0.8 (0.0)
	$300 \leq \|S\| \leq 500$	-(23.1)	185.3 (7.4)	3.3	**2.2**	-(23.5)	346.0 (7.6)	4.1	**2.3**	-(23.9)	601.1 (9.5)	6.5	**2.7**
	$650 \leq \|S\| \leq 750$	-(10.0)	105.4 (4.4)	2.7	**1.9**	-(11.7)	195.8 (4.5)	3.2	**2.1**	-(12.4)	422.7 (4.6)	4.6	**2.5**

Table 2 Computing time (seconds) and percentage gaps (%) for the MMNL instances

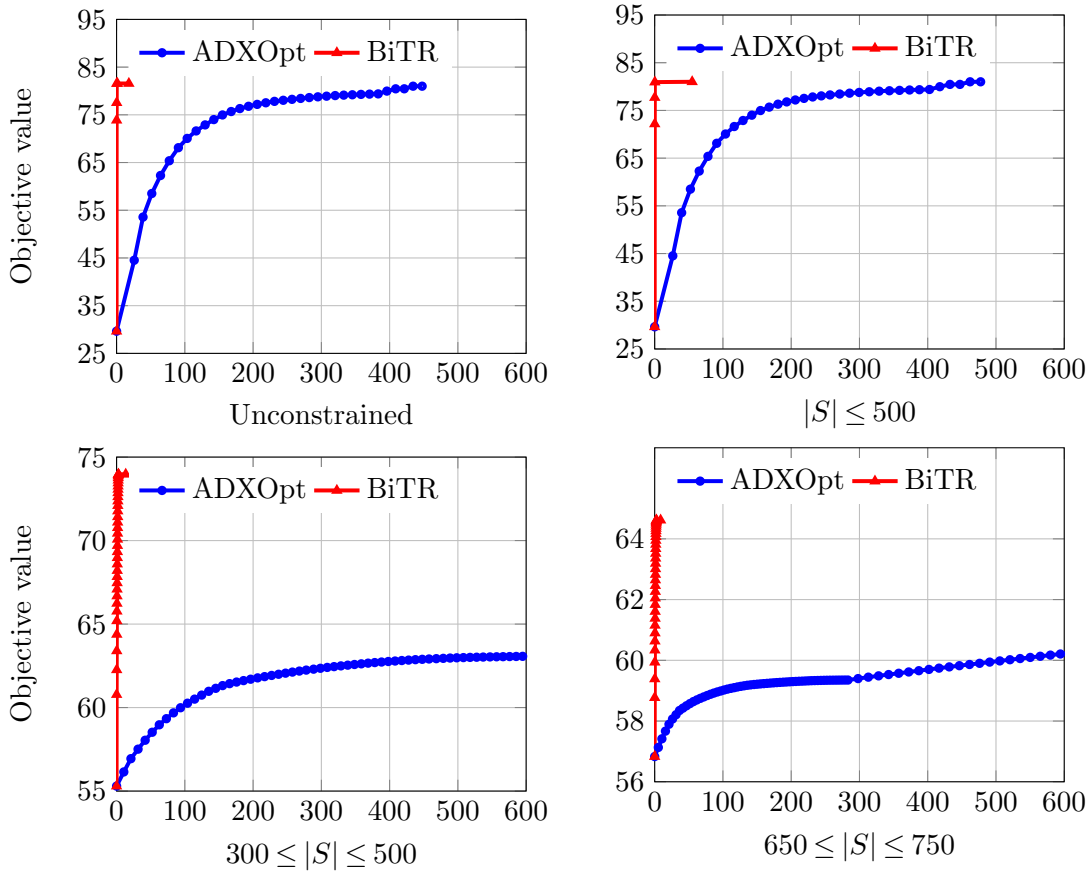

Figure 4 **Computing time and objective values found, given by the BiTR and ADXOpt algorithms for MMNL instances with 1,000 products, $K = 500$, and given a time budget of 600 seconds.**

by the fact that, for these instances, the cost of computing the objective function is much lower as compared to the cost for solving the sub-problem of the BiTR algorithm.

	K	5			10			20		
m	Constraints	MILP	ADXOpt	BiTR	MILP	ADXOpt	BiTR	MILP	ADXOpt	BiTR
10	-	**0.21**	0.61	0.52	0.14	**0.01**	0.46	0.15	**0.01**	0.45
	$\|S\| \leq 3$	0.17	**0.02**	0.55	0.14	**0.01**	0.54	0.17	**0.01**	0.47
	$3 \leq \|S\| \leq 5$	0.15	**0.01**	0.47	0.14	**0.01**	0.47	0.16	**0.01**	0.47
	$5 \leq \|S\| \leq 7$	0.22	**0.01**	0.47	0.14	**0.01**	0.46	0.15	**0.01**	0.48
20	-	1.62	**0.03**	0.45	4.47	**0.03**	0.48	6.82	**0.03**	0.46
	$\|S\| \leq 10$	1.53	**0.03**	0.48	2.65	**0.03**	0.47	5.69	**0.03**	0.46
	$3 \leq \|S\| \leq 10$	1.53	**0.03**	0.47	3.14	**0.02**	0.48	5.30	**0.02**	0.49
	$10 \leq \|S\| \leq 15$	1.38	**0.03**	0.46	2.99	**0.03**	0.45	9.01	**0.02**	0.46
30	-	28.75	**0.06**	0.47	240.71	**0.07**	0.48	601.64	**0.07**	0.49
	$\|S\| \leq 15$	26.29	**0.06**	0.50	142.74	**0.06**	0.48	500.60	**0.07**	0.50
	$10 \leq \|S\| \leq 15$	31.18	**0.05**	0.50	145.94	**0.05**	0.48	489.38	**0.06**	0.48
	$15 \leq \|S\| \leq 20$	34.12	**0.05**	0.50	233.92	**0.05**	0.49	601.20	**0.06**	0.50

Table 3 **Computing time (seconds) for the MMNL problem with small-size instances, all the approaches return optimal solutions.**

5.4. Case study 3: Network MEV model

For this case study, we build a cross-nested structure by grouping the products according to certain common features. For example, we create a nest grouping of products whose color is red, or a nest grouping of products that belong to a specific item class (see Figure 5 for an illustration). This way of modeling results in a cross-nested logit model (i.e., a two-level network MEV model) of 111 nests and the network of correlation structure contains 1,981 directed links. We estimate the parameters μ and α and the parameters associated with all the products' attributes. In total, there are 223 parameters to be estimated. We use the dynamic programming techniques proposed by Mai et al. (2017) to accelerate the estimation and the computation of the objective function (i.e., expected revenue).

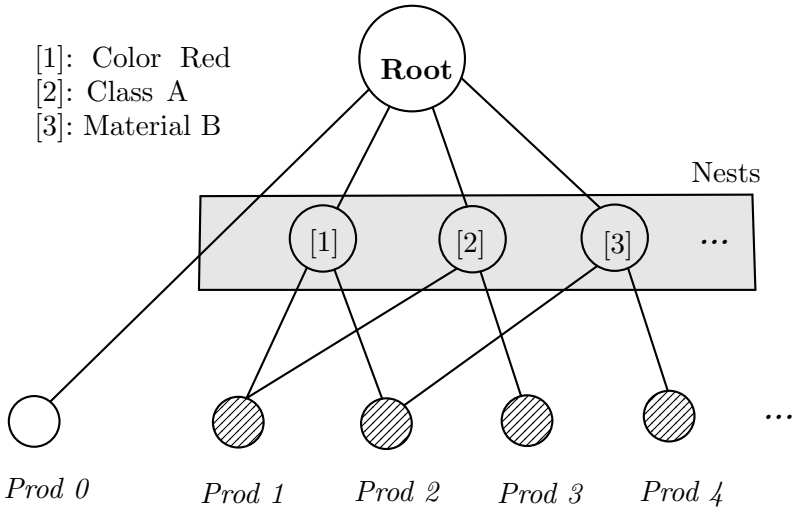

Figure 5 **Example of a cross-nested correlation structure between products.**

In this study, we also remove the "exchange" step from ADXOpt and Steps 2.4 - 2.7 from Algorithm 2, due to that fact that these steps are too costly to perform. Table 4 reports the computing time and percentage gap (%) for the ADXOpt, BiTR and BiTR-noLS (i.e., Algorithm 1 without Step #3). Bold numbers again indicate the fastest methods among those returning best solutions. Symbol "-" is again used to indicate that the approach exceeds the time budget of 600 seconds without that any feasible solution is obtained. In general, the BiTR-noLS is the fastest one, and the BiTR is faster than the ADXOpt. The BiTR returns best solutions for 17/24 instances, and the ADXOpt manage to find best solutions for 15/24 instances. The average percentage gap given by the BiTR is 0.12, which is significantly smaller than the average percentage gap of 0.24 given by the ADXOpt. We also note that, even if it is very fast, the solutions given by the BiTR-noLS are not as good as those given by the other approaches, meaning that the "local search" step of Algorithm 1 really helps improve the solutions given by the BiTR-noLS.

m	Constraints	ADXOpt	BiTR	BiTR-noLS
100	-	2.6(0.02)	**2.6**	1.1(2.29)
	$\|S\| \leq 50$	2.4(0.02)	**2.2**	0.8(2.29)
	$30 \leq \|S\| \leq 50$	3.4	2.5	**2.3**
	$50 \leq \|S\| \leq 70$	2.9(0.02)	**1.7**	1.6(0.20)
200	-	**8.2**	8.5	1.2(3.57)
	$\|S\| \leq 100$	**8.2**	8.4	1.1(3.57)
	$70 \leq \|S\| \leq 100$	**13.0**	4.2(0.77)	4.0(0.77)
	$120 \leq \|S\| \leq 160$	8.0(1.80)	**7.6**	2.1(6.73)
400	-	**39.8**	24.5(0.07)	3.9(0.84)
	$\|S\| \leq 200$	**41.0**	23.7(0.07)	3.9(0.84)
	$100 \leq \|S\| \leq 200$	**102.0**	13.0(0.32)	10.4(0.47)
	$250 \leq \|S\| \leq 350$	54.3(0.24)	**12.3**	12.1(0.76)
600	-	82.9	**33.3**	12.7(0.22)
	$\|S\| \leq 300$	82.4	**32.8**	12.7(0.22)
	$200 \leq \|S\| \leq 300$	**211.6**	22.6(0.43)	21.8(0.43)
	$450 \leq \|S\| \leq 550$	75.7(0.02)	**16.0**	15.7(0.77)
800	-	183.0(0.05)	**92.7**	21.0(0.55)
	$\|S\| \leq 400$	183.3(0.05)	**93.1**	20.6(0.55)
	$300 \leq \|S\| \leq 400$	**375.1**	35.7(0.25)	34.5(0.25)
	$550 \leq \|S\| \leq 650$	**228.8**	24.3(0.90)	23.6(0.90)
1,000	-	476.7	**166.3**	24.3(0.77)
	$\|S\| \leq 500$	478.2	**166.4**	24.5(0.77)
	$300 \leq \|S\| \leq 500$	-(3.57)	**57.9**	45.2(0.10)
	$650 \leq \|S\| \leq 750$	-	**44.2**	39.2(0.06)
Average percentage gap		0.24	0.12	1.16

Table 4 Computing time (seconds) and percentage gaps (%) for the network MEV instances.

In Table 4, the performance of the ADXOpt seems to be comparable on several instances. Thus, we provide an additional experiment to have a closer look at the performance of our methods. More precisely, we test the MEV instances with small time budgets, i.e., 120 and 20 seconds. Table 5 reports the percentage gap for the ADXOpt, BiTR and BiTR-noLS, where the values of 0.0 indicate that the corresponding algorithms return the best solutions. It is clear from the table that the ADXOpt performs well on small instances of less than 400 items. However, for larger instances, the gaps given by the ADXOpt are remarkably high. Especially, with instances of 1000 items, ADXOpt's gaps can go up to 19.7%. In general, the experiment shows the effectiveness of our approaches in handling large-scale instances with a very limited time budget.

Now, we turn our attention to the convergence of the BiTR and ADXOpt under the largest instances, i.e., instances of 1,000 products. In Figure 6, we plot the computing time and objective value for the four types of feasible sets. Similar to what we observed in the previous case studies, the BiTR manages to go quickly to good solutions, while the ADXOpt can only improve the objective value slowly, and it exceeds the time budget for 2/4 of the instances considered.

Similarly to the case study with MMNL instances, we also test on small instances to validate the quality of the solutions given by the two heuristic approaches BiTR and ADXOpt. More precisely,

m	Constraints	Time budget = 120s			Time budget = 20s		
		ADXOpt	BiTR	BiTR noLS	ADXOpt	BiTR	BiTR noLS
100	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$\|S\| \leq 50$	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$30 \leq \|S\| \leq 50$	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
	$50 \leq \|S\| \leq 70$	**0.0**	0.2	0.2	**0.0**	0.2	0.2
200	-	**0.0**	**0.0**	0.8	0.9	**0.0**	0.8
	$\|S\| \leq 100$	**0.0**	**0.0**	0.8	0.5	**0.0**	0.8
	$70 \leq \|S\| \leq 100$	**0.0**	0.8	0.8	5.8	**0.0**	**0.0**
	$120 \leq \|S\| \leq 160$	**0.0**	0.6	0.6	1.4	**0.0**	**0.0**
400	-	**0.0**	0.1	0.8	3.7	**0.0**	0.5
	$\|S\| \leq 200$	**0.0**	0.1	0.8	3.7	**0.0**	0.5
	$100 \leq \|S\| \leq 200$	8.7	**0.0**	0.1	12.0	**0.0**	0.1
	$250 \leq \|S\| \leq 350$	1.2	**0.0**	**0.0**	8.1	**0.0**	**0.0**
600	-	0.6	**0.0**	0.2	8.4	**0.0**	0.1
	$\|S\| \leq 300$	0.5	**0.0**	0.2	7.0	**0.0**	0.1
	$200 \leq \|S\| \leq 300$	9.9	**0.0**	**0.0**	12.8	**0.0**	**0.0**
	$450 \leq \|S\| \leq 550$	0.8	**0.0**	**0.0**	4.8	**0.0**	**0.0**
800	-	2.2	**0.0**	0.5	12.9	**0.0**	0.1
	$\|S\| \leq 400$	2.5	**0.0**	0.5	15.0	**0.0**	**0.0**
	$300 \leq \|S\| \leq 400$	11.4	**0.0**	**0.0**	14.4	**0.0**	**0.0**
	$550 \leq \|S\| \leq 650$	5.7	**0.0**	**0.0**	7.2	**0.0**	**0.0**
1000	-	4.4	**0.0**	0.5	16.4	**0.0**	**0.0**
	$\|S\| \leq 500$	4.3	**0.0**	0.5	19.7	**0.0**	**0.0**
	$300 \leq \|S\| \leq 500$	13.5	**0.0**	0.1	16.6	**0.0**	**0.0**
	$650 \leq \|S\| \leq 750$	7.0	**0.0**	0.1	8.6	**0.0**	**0.0**
Average		3.0	**0.1**	0.3	7.5	**0.0**	0.1

Table 5 Percentage gaps (%) for the MEV instances, under small time budgets.

we use instances of 10, 15 and 20 products. For such instances, it is possible to enumerate all the feasible assortments and find the optimal ones, so we are able to compare solutions given by the BiTR and ADXopt to the optimal ones. Table 6 reports our numerical results, where ES (Exhaustive Search) is the method of enumerating and searching over all the feasible solutions. Interestingly, both BiTR and ADXOpt manage to return optimal solutions for all the instances. The BiTR and ADXOpt perform similarly in terms of computing time, and the ES approach is, expectedly, very slow as compared to the two other approaches.

In summary, the experiments show that our approach outperforms both existing exact and heuristic methods (i.e., CPLEX solving MILP formulations and the ADXOpt), in the sense that it is able to return better solutions in shorter computing times. This clearly indicates the benefits of taking into consideration the gradient information of the objective function for the search. In particular, the BiTR algorithm allows to return good solutions in a matter of seconds, which might be of interest from a retail company's viewpoint. That is, since the algorithm is general and fast, it opens the possibility for considering several discrete choice models, and quickly solving several

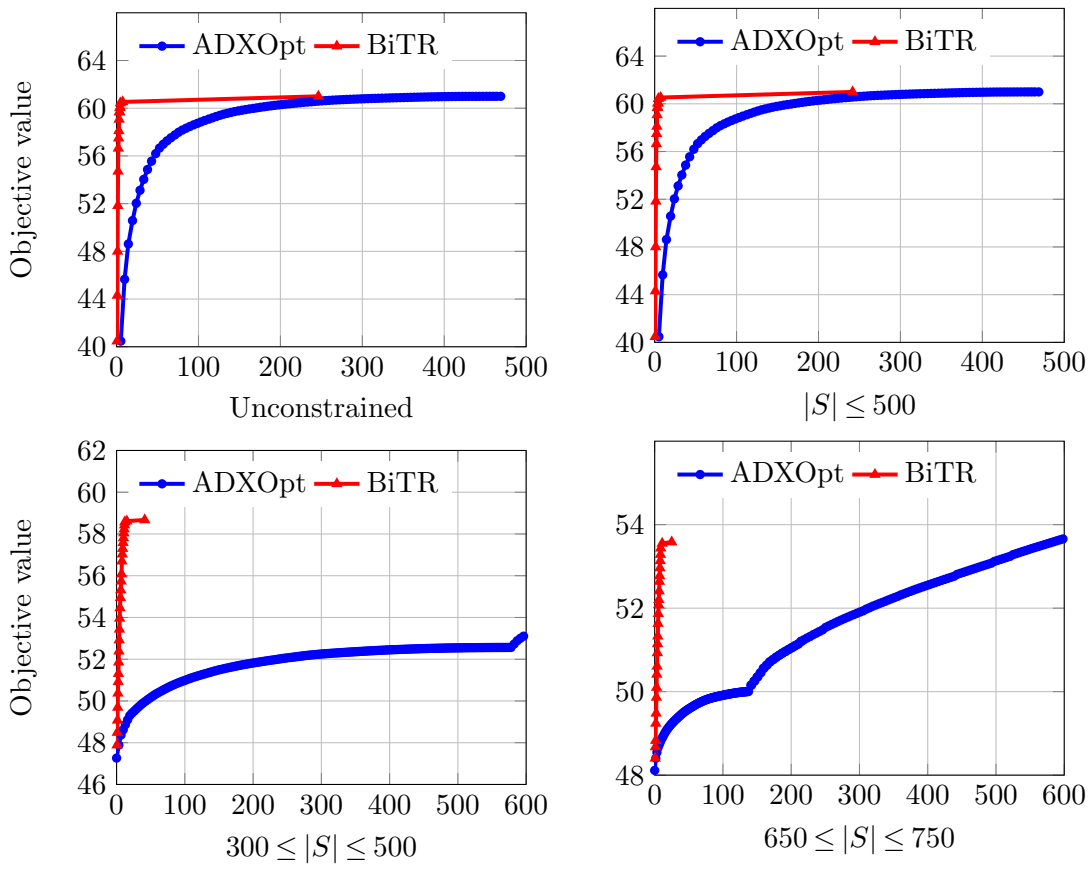

Figure 6 **Computing time and objective values found, given by the BiTR and ADXOpt algorithms for MEV instances of 1,000 products, and given a time budget of 600 seconds.**

m	Constraints	ES	ADXOpt	BiTR
10	-	1.0	**0.1**	0.1
	$\|S\| \leq 3$	0.2	**0.1**	**0.1**
	$3 \leq \|S\| \leq 5$	0.6	**0.1**	**0.1**
	$5 \leq \|S\| \leq 7$	0.6	**0.1**	**0.1**
15	-	33.8	**0.2**	**0.2**
	$\|S\| \leq 5$	5.2	**0.2**	**0.2**
	$5 \leq \|S\| \leq 8$	21.5	**0.3**	**0.3**
	$9 \leq \|S\| \leq 13$	10.2	0.2	**0.1**
20	-	1200.7	**0.2**	0.3
	$\|S\| \leq 10$	717.8	**0.7**	**0.7**
	$3 \leq \|S\| \leq 10$	705.5	**0.5**	**0.5**
	$10 \leq \|S\| \leq 15$	681.8	0.3	**0.2**

Table 6 **Computing time (seconds) for the MEV model and small-size instances, all the approaches return optimal solutions.**

assortment optimizations to find a collection of solution candidates, from which one can pick solutions that make sense. Moreover, our approach is not limited to simple constraints considered above, thus it provides great flexibility to deal with different business requirements.

6. Conclusion

In this paper, we proposed a new algorithm for the assortment optimization problem under different parametric choice models. The problem is challenging due to the fact that the expected revenue function is highly nonlinear and non-convex. Our approach is based on the idea that we can iteratively approximate the objective function by a linear one and perform a "local search" based on this approximate function. In the special but natural case in which the constraints on the assortment structure are lower and upper bounds on its size, we devised a polynomial-time algorithm that solve the subproblem in each iteration, thus allowing us to efficiently find candidate assortments. We also developed a greedy local search approach to further improve the solutions. In addition, several theoretical properties of the greedy algorithm were also discovered for the MNL special case. Those properties help to accelerate the search process.

We have tested our BiTR algorithm on instances under the MNL, MMNL and network MEV models. The results show that our algorithm, called BiTR, dominates the classical heuristic algorithm ADXOpt and it is practically effective with respect to CPLEX solving MILP formulations of the problems.

In summary, the extensive computational tests have shown that the BiTR is able to provide good solutions in short computing time. So, this method should be useful for some real-life applications, e.g., online retail businesses, where one needs demand models to accurately capture customers' demand, and a real-time solution method to quickly provide good assortment solutions under some relatively simple business constraints.

For the future search, we are interested in extending the BiTR to handle the assortment-prices planning problem, i.e., the problem of simultaneously selecting assortment and prices for products in order to maximize the expected revenue. This is also interesting to see how the BiTR can be applied for other data-driven optimization problems, e.g., the maximum captured problem in facility location, where the demand of users is modeled and predicted by a general parametric choice model.

References

Ben-Akiva M (1973) *The structure of travel demand models*. Ph.D. thesis, MIT.

Ben-Akiva M, Bierlaire M (1999) Discrete choice methods and their applications to short-term travel decisions. Hall R, ed., *Handbook of Transportation Science*, 5–34 (Kluwer).

Ben-Akiva M, Lerman SR (1985) *Discrete Choice Analysis: Theory and Application to Travel Demand* (MIT Press, Cambridge, Massachusetts).

Bertsimas D, Mišić V (2017) Exact first-choice product line optimization. *Forthcoming in Operations Research* .

Bront JJM, Méndez-Díaz I, Vulcano G (2009) A column generation algorithm for choice-based network revenue management. *Operations Research* 57(3):769–784.

Daly A, Bierlaire M (2006) A general and operational representation of generalised extreme value models. *Transportation Research Part B* 40(4):285 – 305.

Davis JM, Gallego G, Topaloglu H (2014) Assortment optimization under variants of the nested logit model. *Operations Research* 62(2):250–273.

Désir A, Goyal V (2014) Near-optimal algorithms for capacity constrained assortment optimization. *Available at SSRN 2543309* .

Feige U, Mirrokni VS, Vondrak J (2011) Maximizing non-monotone submodular functions. *SIAM Journal on Computing* 40(4):1133–1153.

Fischetti M, Lodi A (2003) Local branching. *Mathematical programming* 98(1-3):23–47.

Fosgerau M, McFadden M, Bierlaire M (2013) Choice probability generating functions. *Journal of Choice Modelling* 8:1–18.

Gallego G, Topaloglu H (2014) Constrained assortment optimization for the nested logit model. *Management Science* 60(10):2583–2601.

Jagabathula S (2014) Assortment optimization under general choice. *Available at SSRN 2512831* .

Jena SD, Lodi A, Palmer H (2017) Partially-ranked choice models for data-driven assortment optimization. Technical Report DS4DM-2017-011, Canada Excellence Research Chair.

Koppelman F, Wen CH (2000) The paired combinatorial logit model: properties, estimation and application. *Transportation Research Part B* 34:75–89.

Li G, Rusmevichientong P, Topaloglu H (2015) The d-level nested logit model: Assortment and price optimization problems. *Operations Research* 63(2):325–342.

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

Mai T, Frejinger E, Fosgerau M, Bastin F (2017) A dynamic programming approach for quickly estimating large network-based MEV models. *Transportation Research Part B: Methodological* 98:179–197.

McFadden D (1978) Modelling the choice of residential location. Karlqvist A, Lundqvist L, Snickars F, Weibull J, eds., *Spatial Interaction Theory and Residential Location*, 75–96 (Amsterdam: North-Holland).

McFadden D, Train K (2000) Mixed MNL models for discrete response. *Journal of applied Econometrics* 447–470.

Méndez-Díaz I, Miranda-Bront JJ, Vulcano G, Zabala P (2010) A branch-and-cut algorithm for the latent class logit assortment problem. *Electronic Notes in Discrete Mathematics* 36:383–390.

Nocedal J, Wright SJ (2006) *Numerical Optimization* (New York, NY, USA: Springer), 2nd edition.

Rusmevichientong P, Shen ZJM, Shmoys DB (2010) Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research* 58(6):1666–1680.

Rusmevichientong P, Shmoys D, Tong C, Topaloglu H (2014) Assortment optimization under the multinomial logit model with random choice parameters. *Production and Operations Management* 23(11):2023–2039.

Rusmevichientong P, Topaloglu H (2012) Robust assortment optimization in revenue management under the multinomial logit choice model. *Operations Research* 60(4):865–882.

Small KA (1987) A discrete choice model for ordered alternatives. *Econometrica* 55(2):409–424.

Talluri K, Van Ryzin G (2004) Revenue management under a general discrete choice model of consumer behavior. *Management Science* 50(1):15–33.

Train K (2003) *Discrete Choice Methods with Simulation* (Cambridge University Press).

van Ryzin G, Vulcano G (2014) A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science* 61(2):281–300.

van Ryzin G, Vulcano G (2017) An expectation-maximization method to estimate a rank-based choice model of demand. *Operations Research* 65(2):396–407.

Vovsha P, Bekhor S (1998) Link-nested logit model of route choice Overcoming route overlapping problem. *Transportation Research Record* 1645:133–142.

Wu TH (1997) A note on a global approach for general 0–1 fractional programming. *European Journal of Operational Research* 101(1):220–223.

Appendix. Proofs of Propositions 1-5

Proof of Proposition 1. We let S^* and x^* denote an assortment and the corresponding binary solution given by the local search algorithm. We will prove that for any assortment S such that $LB \leq |S| \leq UB$ we have $R(S^*) \geq R(S)$. First, we have

$$R(S^*) = \frac{\sum_{i\in S^*} V_i r_i}{\sum_{i\in S^*} V_i}.$$

Again, for notational simplicity we define $A = \sum_{i\in S^*} V_i r_i$ and $B = \sum_{i\in S^*} V_i$. We also have the fact that for any other assortment S, S can be obtained by exchanging and removing/adding products from/to S^*. More precisely, from S^* we can keep doing the exchanges until we get an assortment $\bar{S}$ such that $\bar{S} \subset S$ or $S \subset \bar{S}$, then if $\bar{S} \subset S$ we add more products to $\bar{S}$ to obtain S, otherwise we remove some products from $\bar{S}$. These operations can be expressed in a formal way as follows

$$S = \begin{cases} (S^* - i_1 + j_1 - \ldots - i_p + j_p) + j_{p+1} + \ldots + j_h & \text{if } |S^*| \leq |S| \\ (S^* - i_1 + j_1 - \ldots - i_p + j_p) - i_{p+1} + \ldots + i_l & \text{if } |S^*| > |S| \end{cases} \tag{13}$$

$i_1, \ldots, i_l \in S^* \backslash \{0\}$, and $j_1, \ldots, j_h \notin S^*$, and each operation "$-i_t + j_t$" stands for exchanging $i_t \in S^*$ with $j_t \notin S^*$. Now, we prove that $R(S^*) \geq R(S)$. Because S^* is a solution of the local search algorithm, we have

$$R(S^*) = \frac{A}{B} \geq \frac{A + V_j r_j - V_i r_i}{B + V_j - V_i}, \quad \forall i \in S^* \backslash \{0\}, j \notin S^*, \tag{14}$$

$$R(S^*) = \frac{A}{B} \geq \frac{A + V_j r_j}{B + V_j}, \quad \forall j \notin S^*, \text{if } |S| > |S^*|,$$
$$R(S^*) = \frac{A}{B} \geq \frac{A - V_i r_i}{B - V_i}, \quad \forall i \in S^* \backslash \{0\}, \text{if } |S| \leq |S^*|,$$

or equipvalently,

$$A(V_j - V_i) \geq B(V_j r_j - V_i r_i) \quad \forall i \in S^* \backslash \{0\}, j \notin S^*,$$
$$AV_j \geq BV_j r_j, \quad \forall j \notin S^*, \text{if } |S| > |S^*|,$$
$$A(-V_i) \geq B(-V_i r_i), \quad \forall i \in S^* \backslash \{0\}, \text{if } |S| \leq |S^*|.$$

So, if we incorporate the above inequalities with (13) we have

- If $|S| \geq |S^*|$, then

$$A(-V_{i_1} + V_{j_1} - \ldots - V_{i_p} + V_{j_p} + V_{j_{p+1}} + \ldots + V_{j_h}) \geq B(-V_{i_1} r_{i_1} + V_{j_1} r_{j_1} - \ldots - V_{i_p} r_{i_p} + V_{j_p} r_{j_p} + V_{j_{p+1}} r_{j_{p+1}} + \ldots + V_{j_h} r_{j_h}),$$

or equivalently,

$$\frac{A}{B} \geq \frac{A - V_{i_1} r_{i_1} + V_{j_1} r_{j_1} - \ldots - V_{i_p} r_{i_p} + V_{j_p} r_{j_p} + V_{j_{p+1}} r_{j_{p+1}} + \ldots + V_{j_h} r_{j_h}}{B - V_{i_1} + V_{j_1} - \ldots - V_{i_p} + V_{j_p} + V_{j_{p+1}} + \ldots + V_{j_h}},$$

or $R(S^*) \geq R(S)$.

- If $|S| < |S^*|$, then

$$A(-V_{i_1} + V_{j_1} - \ldots - V_{i_p} + V_{j_p} - V_{i_{p+1}} - \ldots - V_{i_l}) \geq B(-V_{i_1} r_{i_1} + V_{j_1} r_{j_1} - \ldots - V_{i_p} r_{i_p} + V_{j_p} r_{j_p} - V_{i_{p+1}} r_{i_{p+1}} - \ldots - V_{i_l} r_{i_l}),$$

or equivalently,

$$\frac{A}{B} \geq \frac{A - V_{i_1} r_{i_1} + V_{j_1} r_{j_1} - \ldots - V_{i_p} r_{i_p} + V_{j_p} r_{j_p} - V_{i_{p+1}} r_{i_{p+1}} - \ldots - V_{i_l} r_{i_l}}{B - V_{i_1} + V_{j_1} - \ldots - V_{i_p} + V_{j_p} - V_{i_{p+1}} - \ldots - V_{i_l}},$$

and this also leads to $R(S^*) \geq R(S)$.

Q.E.D.

Proof of Proposition 2. First, we write

$$R(S_k) = \frac{\sum_{i \in S_k} V_i r_i}{\sum_{i \in S_k} V_i}.$$

For notational simplicity, let us define $A = \sum_{i \in S_k} V_i r_i$ and $B = \sum_{i \in S_k} V_i$, so $R(S_k) = A/B$. Under the assumption of the proposition, we have

$$R(S_k) \geq \text{argmax}_{S \in (M_k^D \cup M_k^A) \cap M} R(S), \tag{15}$$

with a note that $M_k^D \subset M$ and $M_k^A \subset M$ because $|S_k|$ is strictly in the bounds. If we denote by $S + j$ the operation of adding product j to assortment S (i.e., $S \cup \{j\}$), and by $S - i$ the operation of removing product i from S (i.e., $S \backslash \{i\}$), then (15) can be written equivalently as

$$\begin{cases} R(S_k) \geq R(S_k + j), \forall j \notin S_k \\ R(S_k) \geq R(S_k - i), \forall i \in S_k \backslash \{0\}, \end{cases}$$

$$\Leftrightarrow \begin{cases} \frac{A}{B} \geq \frac{A+V_j r_j}{B+V_j}, & \forall j \notin S_k \\ \frac{A}{B} \geq \frac{A-V_i r_i}{B-V_i}, & \forall i \in S_k \backslash \{0\} \end{cases}$$

$$\Leftrightarrow \begin{cases} AV_j \geq BV_j r_j, & \forall j \notin S_k \\ A(-V_i) \geq B(-V_i r_i), & \forall i \in S_k \backslash \{0\}. \end{cases} \tag{16}$$

Now, for any assortment S such that $LB \leq |S| \leq UB$, we have the fact that S can be always obtained by removing/adding some products from/to S_k, i.e., there exist products $i_1, \ldots, i_p \in S_k \backslash \{0\}$ and $j_1, \ldots, j_q \notin S_k$ such that

$$S = S_k - i_1 - \ldots - i_p + j_1 + \ldots + j_q.$$

According to (16) we have

$$A(-V_{i_1} - \ldots - V_{i_p} + V_{j_1} + \ldots + V_{j_q}) \geq B(-V_{i_1} r_{i_1} - \ldots - V_{i_p} r_{i_p} + V_{j_1} r_{j_1} + \ldots + V_{j_q} r_{j_q}),$$

or equivalently,

$$\frac{A}{B} \geq \frac{A - V_{i_1} r_{i_1} - \ldots - V_{i_p} r_{i_p} + V_{j_1} r_{j_1} + \ldots + V_{j_q} r_{j_q}}{B - V_{i_1} - \ldots - V_{i_p} + V_{j_1} + \ldots + V_{j_q}},$$

meaning that $R(S_k) \geq R(S)$. So S_k is an optimal solution. Q.E.D.

Proof of Proposition 3. Similar to the proof of Proposition 2, we also have that $R(S_k) \geq R(S)$, $\forall S \in M_k^D \cup M_k^A$, is equivalent to

$$\Leftrightarrow \begin{cases} AV_j \geq BV_j r_j, & \forall j \notin S_k \\ A(-V_i) \geq B(-V_i r_i), & \forall i \in S_k \backslash \{0\}, \end{cases} \tag{17}$$

where $A = \sum_{i \in S_k} V_i r_i$ and $B = \sum_{i \in S_k} V_i$. So,

$$\begin{cases} A(V_{j_1} + V_{j_2}) \geq B(V_{j_1} r_{j_1} + V_{j_1} r_{j_1}), & \forall j_1, j_2 \notin S_k \\ A(-V_{i_1} - V_{i_2}) \geq B(-V_{i_1} r_{i_1} - V_{i_2} r_{i_2}), & \forall i_1, i_2 \in S_k \backslash \{0\} \end{cases}$$

$$\Leftrightarrow \begin{cases} \frac{A}{B} \geq \frac{A+V_{j_1} r_{j_1} + V_{j_1} r_{j_1}}{B+V_{j_1}+V_{j_2}}, & \forall j_1, j_2 \notin S_k \\ \frac{A}{B} \geq \frac{A-V_{i_1} r_{i_1} - V_{i_2} r_{i_2}}{B-V_{i_1}-V_{i_2}}, & \forall i_1, i_2 \in S_k \backslash \{0\} \end{cases} \Leftrightarrow \begin{cases} R(S_k) \geq R(S_k + j_1 + j_2), & \forall j_1, j_2 \notin S_k \\ R(S_k) \geq R(S_k - i_1 - i_2), & \forall i_1, i_2 \in S_k \backslash \{0\}. \end{cases}$$

This also means that $R(S_k) \geq R(S)$, $\forall S \in M_k^{2A} \cup M_k^{2D}$. Q.E.D.

Proof of Proposition 4. We have that, at Step 2.1, a product $i \in S_k \backslash \{0\}$ could be removed from S_k if $R(S_k) < R(S_k - i)$, meaning that

$$\frac{A}{B} < \frac{A - V_i r_i}{B - V_i} \Leftrightarrow A > B r_i, \text{ or equipvalently } R(S_k) > r_i$$

Similarly, at Step 2.2, a product $j \notin S_k$ can be added to S_k if

$$\frac{A}{B} < \frac{A + V_j r_j}{B + V_j} \Leftrightarrow A < B r_j \Leftrightarrow R(S_k) < r_j.$$

Q.E.D.

Proof of Proposition 5. We consider the case that $|S_k| = LB$. Because $R(\bar{S}) \leq R(S_k)$, there is no product that should be added to S_k. According to Proposition 4 we have

$$R(S_k) \geq r_j, \ \forall j \in \mathcal{U} \backslash S_k.$$

We now show, by contradiction, that if product $i_k \in S_k \backslash \{0\}$ is exchanged with $j_k \in \mathcal{U} \backslash S_k$ then $R(S_k) > r_{i_k}$. Indeed, if $R(S_k) \leq r_{i_k}$ then

$$\begin{cases} A/B \geq r_{j_k} \\ A/B \leq r_{i_k} \end{cases} \Leftrightarrow \begin{cases} AV_{j_k} \geq BV_{j_k} r_{j_k} \\ A(-V_{i_k}) \leq B(-V_{i_k} r_{i_k}) \end{cases}$$

$$\Rightarrow A(V_{j_k} - V_{i_k}) \geq B(V_{j_k} r_{j_k} - V_{i_k} r_{i_k}) \Rightarrow \frac{A}{B} \geq \frac{A + V_{j_k} r_{j_k} - V_{i_k} r_{i_k}}{B + V_{j_k} - V_{j_k}},$$

meaning that $R(S_k) \geq R(S_k + j_k - i_k)$. This contradicts the supposition that i_k is exchanged with j_k by an "exchange" step.

So, after the "exchange" step at iteration k, at the next iteration $k+1$ we have $\mathcal{U} \backslash S_{k+1} = \mathcal{U} \backslash S_k \backslash \{j_k\} \cup \{i_k\}$. Because $R(S_k) > r_{i_k}$ as stated above, so we have $R(S_k) \geq r_j$, $\forall j \in \mathcal{U} \backslash S_{k+1}$. Moreover, $R(S_{k+1}) > R(S_k)$, so in general we have

$$\begin{cases} |S_{k+1}| = LB \\ R(S_{k+1}) \geq r_j, \forall j \in \mathcal{U} \backslash S_{k+1}. \end{cases}$$

Hence, by induction, we complete the proof. Q.E.D.