Submitted to *Management Science*
manuscript MS-16-00074.R1

Assortment Optimization Under Consider-then-Choose Choice Models

Ali Aouad
Operations Research Center, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, aaouad@mit.edu

Vivek Farias
Sloan School of Management, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, vivekf@mit.edu

Retsef Levi
Sloan School of Management, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, retsef@mit.edu

Consider-then-choose models, borne out by empirical literature in marketing and psychology, explain that customers choose among alternatives in two phases, by first screening products to decide which alternatives to consider, before then ranking them. In this paper, we develop a dynamic programming framework to study the computational aspects of assortment optimization under consider-then-choose premises. Although non-parametric choice models generally lead to computationally intractable assortment optimization problems, we are able to show that for many practical and empirically vetted assumptions on how customers consider and choose, our resulting dynamic program is efficient. Our approach unifies and subsumes several specialized settings analyzed in previous literature. Empirically, we demonstrate the versatility and predictive power of our modeling approach on a combination of synthetic and real industry datasets, where prediction errors are significantly reduced against common parametric choice models. In synthetic experiments, our algorithms lead to practical computation schemes that outperform a state-of-the-art integer programming solver in terms of running time, in several parameter regimes of interest.

Key words: Assortment Planning, choice models, dynamic programming, consider-then-choose.

1. Introduction

What selection of products should an e-retailer display for each search query? How does a brick and mortar retailer determine the product assortment in each store? The challenge of finding a selection of products that maximizes revenue or customer satisfaction, in the face of heterogeneous customer segments, who have different preferences across products, has been recognized in several industries as a strategic and operational driver of success.

In a highly differentiated market, choosing an optimal assortment requires to model beforehand the customers preferences to predict accurately how the demand shares of products evolve in response to variations in the *offer set*, through what is called a *choice model*. Building an effective choice model strikes a delicate balance between several desired attributes. Indeed, as choice models become more detailed, both their estimation from data, and the resulting optimization problems face computational barriers, due to what is known as the curse of dimensionality. In fact, assortment optimization is combinatorial in nature and yields hard problems. The present paper demonstrates that a class of choice models, referred to in the literature as 'consider-then-choose' models, render assortment optimization tractable under a wide variety of modeling primitives. We present a unique dynamic programming formulation of the problem, and show that a state space collapse in this problem yields the aforementioned tractability in several practical cases. Outside of theory, we empirically demonstrate the predictive power of our modeling approach using both synthetic and real industry datasets. We illustrate the computational practicality of our approach through extensive comparisons with state-of-the-art integer programming solvers.

Choice modeling and assortment optimization. Generally speaking, choice models can be divided into *parametric* and *non-parametric* models, the latter of which are effectively general distributions over preference lists of products. Until recently, most of the work related to assortment optimization has focused on parametric choice models, primarily attraction-based models in which customer preferences are modeled through a relatively small number of parameters. The survey by Kök et al. (2009) and book by Talluri and Van Ryzin (2006) present excellent overviews on such topics, and our literature review in Section 1.2 will summarize the state of the art here. In a nutshell, the literature presents us with the following dichotomy: on the one hand, for simple parametric models such as the Multinomial Logit (MNL) and variants of the Nested logit (NL) model, we now have efficient algorithms available for assortment optimization. On the other hand, these same models impose structural assumptions on customer preferences that may prove unrealistic in practice (Debreu 1960, Ben-Akiva and Lerman 1985). In attempting to address this latter issue, one may consider further generalized models such as a mixture of MNL models (MMNL), but then assortment optimization is no longer easy with the best known algorithms having a complexity that scales exponentially in the cardinality of the mixture (Bront et al. 2009, Rusmevichientong et al. 2014, Désir and Goyal 2014). In addition this latter class of models is notorious for problems such as over-fitting to data.

In an attempt to construct a parsimonious approach to modeling choice, *non-parametric* choice models, where the choice probability arises from a *sparse* distribution over preference lists (Rusmevichiengtong et al. 2006, Farias et al. 2013, van Ryzin and Vulcano 2014) have also received some attention. Here, each customer type purchases the highest rank item in his preference list made available, or leaves without making any purchase. In this context, Farias et al. (2013) develop a robust estimation methodology, where the sparsity of the distribution scales with the amount of data available, allowing to attain better prediction accuracy than several common parametric models. On the other hand, there is relatively little known on the computational tractability of assortment optimization under these non-parametric models heretofore, beyond a few special cases of interest (Honhon et al. 2012). In fact, sparsity is generally insufficient to alleviate the computational hardness of assortment optimization, and the problem was shown to be NP-hard even to approximate by Aouad et al. (2015).

Consider-then-choose models. The aforementioned parametric and non-parametric models place extremely general conditions on the customer's decision making process, effectively requiring a customer to list *all* her options and then pick her most desirable from that list. In reality, one may naturally expect this process to be different with a customer using a set of simple rules to immediately *disregard* the vast majority of choices, and then rank (and select from) the small number of options left. We refer to such models as *consider-then-choose* models, wherein the *consideration set* is the (small) set of products considered. The history of these consider-then-choose idea originates in the marketing and psychology literature. The idea of whittling down choices into a consideration set was first posited by Campbell (1969) and formulated into a theory of the customer's behavior by Howard and Sheth (1969). In his seminal study, Hauser (1978) observed that the consideration set structure is in itself a significant explanatory factor of choice heterogeneity. We review the evolution of this approach to modeling choice in our literature review in Section 1.2. However, our objective with considering such models is twofold:

1. We believe that these models have the ability to model real-world data. This belief is motivated by empirical observations made in the antecedent literature on whether and how consideration sets are formed. Further, our modeling approach is borne out by several experiments on real industry datasets.
2. This consider-then-choose structure can be leveraged to mitigate the complexity of assortment optimization problems. In particular, we show that many empirically-vetted assumptions on how customers consider and choose lead to tractable assortment optimization problems.

1.1. Our results

Our main contribution is the development of a unified algorithmic framework to study the computational tractability of assortment problems under a family of preference-list based choice models that has been empirically vetted in the marketing literature, specifically, consider-then-choose models. Moreover, our framework allows a direct connection between modeling assumptions on the customers' choice behavior and the resulting computational complexity. Consequently, we show that several practical assumptions regarding how customers consider and choose lead to tractable assortment optimization models. Our dynamic programming algorithm, based on a divide-and-conquer approach in a specific graph representation, provides computationally efficient heuristics for more general preference list distributions, and outperforms a state-of-the-art integer programming solver (IP) for several class of instances. We demonstrate the predictive power of the proposed consider-then-choose modeling framework against common parametric models, using both synthetic experiments and real-world datasets. Our industry partner, Infoscout Inc, operates the largest purchase panel in the US, which provides longitudinal purchase information across retailers and product categories. In what follows, we provide a more detailed sketch of our contributions.

Dynamic program and graph representation. Motivated by the empirical observation that the structure of the consideration sets largely explains choice heterogeneity, we start by formulating in Section 3 a dynamic program for *unique-ranking* distributions, where customers consider arbitrary subsets of products, but their relative ranking preferences are derived from a common permutation. We introduce a *bipartite graph representation* of the problem, which is key to our approach and analysis. Indeed, the connected components of this graph capture a natural decomposition of the instance. Our dynamic program makes use of this decomposition procedure in a divide-and-conquer fashion. In contrast to standard dynamic programming, our algorithm relies upon a careful and exhaustive generation of the computational tree prior to solving the recursive equation. This approach allows for a state space collapse, which substantially reduces the complexity under structural assumptions regarding how customers consider and choose. Specifically, we show that the complexity analysis generally boils down to 'counting' the number of connected subgraphs induced by the graph traversal. We prove that our algorithm runs in polynomial time for very sparse distributions, when the number of preference-lists grows logarithmically in the number of products. Also, we show that even in the worst case, our algorithm dominates the brute force enumerative approach.

The extension to general preference list-distributions requires additional technicalities, which are described in Section 5. Also, our results naturally extend to capacitated assortment optimization, with a constraint on the size of the assortment. This result is described separately in Appendix F.

Tractable consider-then-choose models. In Section 4, we investigate several models of consideration sets that stem from documented assumptions on the customers' purchasing behavior. We derive polynomial running time guarantees for the corresponding dynamic program in the unique-ranking setting. In Section 6, we investigate more general classes of distributions that combine heterogeneous consideration sets along with ranking heterogeneity. Our results subsume and extend several models studied in previous literature. Our complexity results, and the corresponding structural assumptions, are summarized in Table 1.

Table 1 **Summary of results: polynomial running time guarantees for consider-then-choose choice models.**

Consideration sets	Ranking functions	Running time	Sections
Induced intervals	Neighborhood of a ranking function	$O(N^4K)$	4.1 and 6.1
Laminar properties	Neighborhood of a ranking function	$O(N^2K^2)$	4.2 and 6.1
Disjunction on d features	Neighborhood of a ranking function	$O(N^{4d-2}K)$	4.3 and 6.1
Intervals	Quasi-convex permutations	$O(N^4K^4)$	6.2
Two-feature compensatory	Two-feature compensatory	$O(N^4K^4)$	6.3

The parameter N describes the number of product alternatives, K denotes the number of preference lists (sparsity of the distribution), and d is a complexity parameter corresponding to the number of features considered by customers upon forming their consideration set in the disjunctive model. The notion of induced intervals means that there exists *some* arbitrary numbering according to which the consideration sets are intervals.

Empirical performance. Our numerical experiments on synthetic instances, described in Section 7.1, demonstrate that the algorithm is efficient in practice. We compare its performance against an integer programming formulation implemented using a state-of-the-art commercial solver (GUROBI v6.5). We demonstrate that the IP approach becomes intractable to solve large-scale instances of the quasi-convex model. Even under generic consideration set structures, our approach dominates the IP solver in several regime of parameters.

Finally, we demonstrate in Section 7.2 the versatility of our modeling approach against a benchmark formed by 'small' mixtures of Multinomial Logits (MMNL)[1]. The objective is to predict the relative purchase probabilities of products in various assortments. The predictive power of our approach is demonstrated by the experiments conducted on real-world datasets provided by our industry partner, in three distinct product categories. The errors in out-of-sample predictions of the purchase probabilities are reduced on average by 14% to 25% under various metrics. In synthetic experiments, our consider-then-choose model outperforms the benchmarks in the plurality of cases. Specifically, we use the following ground truth models: a large-mixture MMNL model and a simple consider-then-choose model.

1.2. Related literature

Our work relates to two streams of literature, namely the operational literature on choice modeling and assortment planning, and the marketing literature on consider-then-choose models.

Choice models and assortment optimization. In the last two decades, growing product proliferation and differentiation has motivated a paradigm shift in demand modeling from independent demand models to choice-based models, to capture the substitution effects in a given product category (Mahajan and Van Ryzin 2001, Kök and Fisher 2007, Ratliff et al. 2008, Vulcano et al. 2010). In this context, assortment optimization has received a great deal of attention in operations management literature. Most of the focus has been on variants of this problem under the

[1] In light of previous literature, assortment optimization is practical only for a mixture over a relatively small number of customer segments (a notion we will make precise in Section 7.2).

widespread attraction-based models such as the Multinomial Logit (MNL) model, the discrete Mixture of MNLs (MMNL), etc. Under MNL preferences, the problem is known to be polynomially solvable (Talluri and van Ryzin 2004, Rusmevichientong et al. 2010), and the solution methods were further advanced to handle more general settings (Rusmevichientong and Topaloglu 2012, Davis et al. 2013). However, the tractability of assortment optimization under the attraction-based models does not carry over to heterogeneous customer segments. That is, even with two segments the MMNL-based problem was shown to be NP-complete by Bront et al. (2009) and Rusmevichientong et al. (2014). For a fixed number of customer segments, Désir and Goyal (2014) developed a fully polynomial-time approximation scheme, but its computationally efficiency hinges on modeling few customer segments. Given these computational barriers, recent work in assortment optimization attempts to identify new probabilisitic models leading to tractable assortment optimization problems (Li et al. 2015, Blanchet et al. 2013, Davis et al. 2014).

On the other hand, there has been an emerging literature on preference list-based choice models (Rusmevichiengtong et al. 2006, Farias et al. 2013, Jagabathula and Rusmevichientong 2016). Here, the heterogeneity in choice is explicitly encoded through a distribution over preference lists. This approach to modeling choice is very general, e.g., the attraction-based models can be viewed as parametrized distributions over all potential preference lists. In this context, Farias et al. (2013) proposed an efficient methodology to make robust revenue predictions and derived recovery guarantees under certain technical conditions. To overcome the dimensionality of the estimation problem, van Ryzin and Vulcano (2014) proposed the 'market discovery' algorithm: starting from an initial collection of preference lists, the support of the distribution is enlarged iteratively by generating a preference list that increases the log-likelihood value, using dual information. While estimation methods have been investigated in this setting, assortment optimization remains mostly untapped. Aouad et al. (2015) characterized the complexity class of the problem under generic distributions, while Honhon et al. (2012) developed tailor-made dynamic programming ideas for several special cases, which are subsumed by our analytical results.

Consider-then-choose literature. A steady line of research in marketing and psychology has studied various aspects of the decision-making strategies employed by customers. This literature gives rise to the following key observations.

- *Cognitive simplicity.* To alleviate the cognitive burden in multi-alternative decision tasks, individuals apply simple decision heuristics (Tversky and Kahneman 1975, Payne et al. 1996). Hence, the consideration sets are justified by the need to balance search efforts with potential gains (Hauser and Wernerfelt 1990, Roberts and Lattin 1991). Screening heuristics were shown to be rational under limited time and knowledge (Gigerenzer and Goldstein 1996, Gigerenzer and Selten 2002).

- *Consideration set heuristics.* Consequently, numerous studies in marketing have validated a consider-then-choose decision process, where customers screen products to a smaller relevant set of products before making choice decisions. For example, Pras and Summers (1975), Brisoux and Laroche (1981) and more recently Gilbride and Allenby (2004) showed empirically that customers often form their consideration sets through a 'conjunction' of elimination rules; see also Parkinson and Reilly (1979), Belonax and Mittelstaedt (1978), Laroche et al. (2003). For a detailed view on the consideration set literature, we refer the reader to the surveys by Hauser et al. (2009), Payne et al. (1996), Bettman et al. (1998).
- *Predictive power.* It has generally been observed that the incorporation of a two-stage decision process enables more accurate predictions, for example in market share forecasting (Urban 1975, Silk and Urban 1978), in choice modeling (Roberts and Lattin 1991) or in risky decision-making (Brandstatter et al. 2006). In fact, in his seminal work, Hauser (1978) observed that the heterogeneity in choice decisions is largely explained by the consideration sets. Even with a crude assumption on the ranking decisions (formed uniformly at random), the consideration set structure still explains nearly 80% of the heterogeneity in choice captured by a richer model, which combines the consideration sets with logit-based rankings. This observation can be explained in that the first stage decisions eliminate a large fraction of the alternatives and the resulting consideration sets are comprised of a few products in most categories (Reilly and Parkinson 1985, Belonax and Mittelstaedt 1978, Hauser and Wernerfelt 1990).

Motivated by these findings, the modeling approach we develop subsequently is centered around the notion of consideration sets. We will show that this approach to adding 'structure' to a general distribution over preference lists buys us a great deal from a computational complexity standpoint, and still allows strong predictive power. Prior to our work, the paper by Jagabathula and Rusmevichientong (2016) also incorporates a choice model based on consideration sets. The optimization problem considered therein relates more closely to combinatorial pricing.

2. Modeling Approach and Problem Formulation

Assortment optimization problem. Throughout the paper we use the index $i \in \{1,\dots,N\} = [N]$ to denote one of N products, each is associated with a price P_i. In addition, we use the index $j \in \{1,\dots,K\} = [K]$ to denote one of K customer-types, each is associated with a consideration set $C_j \subseteq [N]$ that specifies the products she is willing to buy and a ranking function σ_j (that is, a permutation over products) that reflects her relative preferences. We let $(\lambda_1,\dots,\lambda_K)$ be the probability vector, where λ_j denotes the respective fraction of customer-type j in the population. The decision maker has to choose an assortment $\mathcal{A} \subseteq [N]$ that maximizes the total revenue. Specifically, let $\operatorname{Rev}(j,\mathcal{A})$ denote the revenue obtained from customer type j given that assortment $\mathcal{A}$

is stocked. Note that if $\mathcal{A} \cap C_j = \emptyset$ then $\text{Rev}(j, \mathcal{A}) = 0$ and otherwise $\text{Rev}(j, \mathcal{A}) = P_{i(\mathcal{A},j)}$, where $i(\mathcal{A}, j) = \arg\min_{i \in \mathcal{A} \cap C_j} \{\sigma_j(i)\}$ is the most preferred product of customer-types within $\mathcal{A}$. Therefore, the objective is to find an assortment $\mathcal{A}$ that maximizes the expected revenue: $\sum_{j \in [K]} \lambda_j \cdot \text{Rev}(j, \mathcal{A})$.

We let $\mathcal{C} = \{C_j : j \in [K]\}$ be the collection of consideration sets and $\Sigma = \{\sigma_j : j \in [K]\}$ be the set of the ranking functions. In contrast to generic preference list distributions, our approach captures consider-then-choose purchasing behaviors by imposing constraints on the sets $\mathcal{C}$ and Σ, respectively. Below, we provide a high-level description of the ingredients used to model $\mathcal{C}$ and Σ, while the precise mathematical definitions are stated in the corresponding parts of the paper.

Consideration set structure. We relate the collection of consideration sets $\mathcal{C}$ to the customers' cognitive process, where they screen products to form their consideration set. To this end, we build upon the survey by Hauser et al. (2009), which provides a unified framework to express the different consideration set models proposed in the marketing literature (see also Gilbride and Allenby (2004) for a similar mathematical formalism). Suppose that each product $i \in [N]$ is represented in by a vector $\mathbf{x}^{(i)} \in \mathbb{R}^d$ in latent d-dimensional feature space. A *screening rule* corresponds to a cut-off level $t_e \in \mathbb{R}$ on a given feature $e \in [d]$ that implies the elimination of all products $i \in [N]$ not satisfying $x_e^{(i)} \geq t_e$. Generally speaking, Hauser et al. (2009) explains that there are several families of cognitive processes whereby customers combine different screening rules to draw their consideration sets:

- *Conjunction of rules.* Here, a product is considered if *each one* of the specified screening rules are *all* satisfied. Namely, each consideration set $C \in \mathcal{C}$ is of the form:
$$C = \bigcap_{e \in [d]} \{i \in [N] : x_e^{(i)} \geq t_e\} \ .$$
- *Disjunction of rules.* A product is considered if *at least* one of specified screening rules is satisfied, leading to:
$$C = \bigcup_{e \in [d]} \{i \in [N] : x_e^{(i)} \geq t_e\} \ .$$
- *Compensatory models.* A product is considered based on a linear combination between different specified screening rules. In this case, there exists a utility vector $\mathbf{u} \in \mathbb{R}^d$ and a cut-off level $t \in \mathbb{R}$ such that the consideration set is of the form:
$$C = \{i \in [N] : \mathbf{u} \cdot \mathbf{x}^{(i)} \geq t\} \ .$$

In Sections 4 and 6, we investigate several classes of distributions over preference lists, where $\mathcal{C}$ is congruent with such combinations of screening rules. It is worth mentioning that, for a large enough d, each of the above models can replicate any arbitrary collection of consideration sets. Hence, for purposes of identifying tractable consideration set structures, we focus on a small number of screening rules – which is also consistent with the cognitive simplicity of customers' decisions.

Ranking decisions. Having explained how we model $\mathcal{C}$, it remains to describe Σ. To ease the exposition, our algorithmic framework is introduced in an incremental way. First, we focus on the heterogeneity of the consideration sets and start our discussion assuming that the collection of rankings Σ is a singleton. Here, we assume that there exists single ranking order common to all customer types, i.e., $\sigma_j = \sigma$, and the heterogeneity in preferences stems only from the heterogeneity of the consideration sets. We refer to this setting as the *unique-ranking model.* As shall be seen subsequently, the unique-ranking model already subsumes several choice models studied in previous literature, and even in this setting, assortment optimization is computationally intractable. Specifically, it was shown by Aouad et al. (2015) that the problem under the unique-ranking model is NP-hard to approximate within factor $O\left(N^{1-\epsilon}\right)$ for any $\epsilon > 0$.

Our algorithmic approach and analysis carry over in the presence of heterogeneity in ranking decisions. Specifically, in Section 6, the unique-ranking assumption is relaxed in two ways: *(i)* by assuming that Σ is formed by similar rankings arising from the local perturbations of a central permutation, or *(ii)* by studying ranking structures motivated by behavioral assumptions (e.g., quasi-convex permutations).

3. Dynamic Program Under Unique-Ranking Distributions

In this section, we present a dynamic programming (DP) formulation under unique-ranking distributions. As some obstacles must be surmounted to consummate our approach in the general case, the algorithm for arbitrary preference list distribution is described separately in Section 5, to ease the exposition. We formulate the dynamic program in two parallel ways, the first corresponds to a traditional recursive formulation and the second is an appropriate graph representation.

Preliminaries. Recall that an instance of the assortment problem is described by the set of parameters $N, K, \Sigma, \mathcal{C}, \sigma_j, C_j, \lambda_j$. Assuming that $\Sigma = \{\sigma\}$, without loss of generality the product indices can be rearranged according to the σ-ordering. That is item 1 is the most preferred product and N is the least preferred one – to lighten the notation, the reference to σ is omitted hereafter. In what follows, we use $[X]$ to denote the set $\{1, \ldots, X\}$.

State space and objective function. The state space is formed by all pairs of subsets (S, T), where S is a subset of products in $[N]$ and T is a subset of types in $[K]$. Specifically, we let $J(S, T)$ be the maximum expected revenue that can be obtained from choosing an assortment of products within S to satisfy the customer-types in T. In the subproblem, we assume that only customers in T can occur and the arrival probabilities are directly induced from the original instance without a renormalization of the corresponding sub-vector. Formally, the subproblem (S, T) is formulated as follows:

$$J(S,T) = \max_{\mathcal{A} \subseteq S} \sum_{j \in T} \lambda_j \cdot \mathrm{Rev}\left(j, \mathcal{A}\right) \ .$$

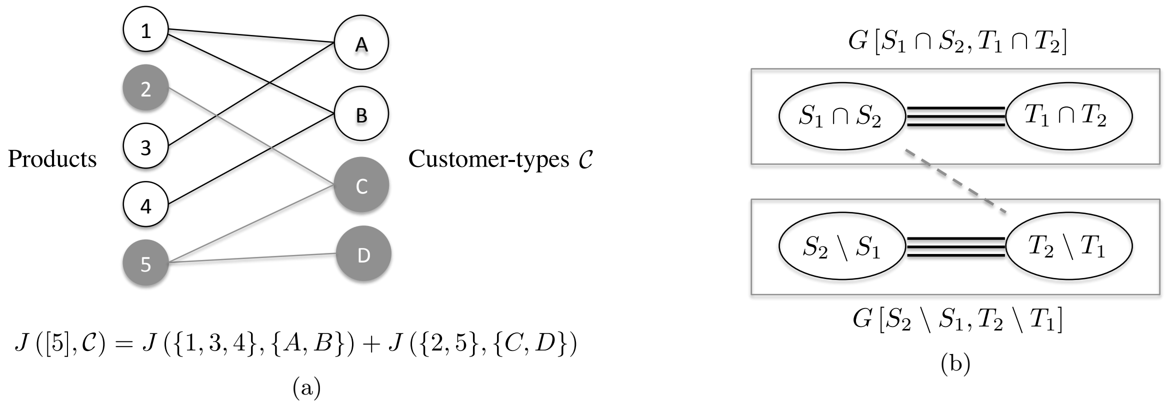

1
2
3
4
5
A
B
C
D
Products
Customer-types $\mathcal{C}$
$J([5],\mathcal{C}) = J(\{1,3,4\},\{A,B\}) + J(\{2,5\},\{C,D\})$
(a)
$G[S_1 \cap S_2, T_1 \cap T_2]$
$S_1 \cap S_2$
$T_1 \cap T_2$
$S_2 \setminus S_1$
$T_2 \setminus T_1$
$G[S_2 \setminus S_1, T_2 \setminus T_1]$
(b)

Figure 1 **Graph Representation. (a) Decomposition of the instance according to the connected components of the graph representation. (b) Representation of the proof of Proposition 1.**

Graph Representation. We next introduce a bipartite graph representation G associated with each instance of the problem. The partite sets are formed by *(i)* the set of products, each of which is represented by a node, and *(ii)* the set of customer-types $[K]$. There is an edge between a customer-type node and a product node if the latter is included in the consideration set of the former. That is, we define the graph $G = ([N],[K],E)$, where $E = \{(i,j) \in [N] \times [K] : \ i \in C_j\}$. This graph induces the family of subgraphs $G[S,T]$ associated with each subproblem (S,T), that is, $G[S,T] = (S,T,E_{S,T})$, where $E_{S,T} = \{(i,j) \in E : \ i \in S \text{ and } j \in T\}$. The lemma below asserts that the partition of $G[S,T]$ into its connected components captures a decomposition of the instance (S,T) into several subproblems that can be solved independently. This decomposition scheme, represented in Figure (1a), is key to our recursion.

LEMMA 1. *Assuming that the connected components of $G[S,T]$ are described by the collection of subgraphs $(G[S_u,T_u])_{u\in[r]}$, where S_u denotes a subset of product nodes in S and T_u is a subset of type nodes in T, then $J(S,T) = \sum_{u=1}^{r} J(S_u,T_u)$.*

Proof It is sufficient to prove that the expected revenue generated in (S,T) by any assortment $\mathcal{A} \subseteq S$ decomposes into the sum over $u \in [r]$ of the revenues generated in each subproblem (S_u,T_u) by the respective assortment $\mathcal{A}_u = \mathcal{A} \cap S_u$. Let j be a customer-type in T_u. The main observation is that customer-type j's most preferred product within the assortment $\mathcal{A}$ is the same as the one he prefers when faced with the assortment $\mathcal{A}_u$. Indeed, by connectivity, any product in $C_j \cap S$ that is considered by customer j, necessarily belongs to S_u. Since $(S_u)_{u\in[r]}$ forms a partition of S, an optimal assortment $\mathcal{A}$ for subproblem (S,T) is the union of optimal assortments $\mathcal{A}_u \subseteq S_u$ for each subproblem (S_u,T_u). $\square$

Base case. If $S = [N]$ and $T = [K]$, then $J(S,T)$ corresponds to the original problem we wish to solve. Using Lemma 1, this problem can be broken-down into separate optimization problems according to the connected components partition. From this point on, we may assume without loss of generality that connectivity is an invariant of the subgraphs examined by the recursion.

Recursive step. We consider the subproblem (S,T) such that $G[S,T]$ is a connected subgraph. We define i as the most preferred product among product nodes S, i.e., $i = \min(S)$. The decision (or action) made by the dynamic program for state (S,T) is whether to stock product i in the assortment or not. Next, we describe graph operations on $G[S,T]$ that correspond to each alternative. As these graph operations decompose $G[S,T]$ into more refined connected components, a natural recursion consists in examining the immediate reward and reward-to-go induced by each stocking decision.

Case 1: Product i is stocked. Let $T(i)$ be the customer-types whose consideration sets contain product i. The unique-ranking order implies that any product added to the assortment at some later point of the recursion is less preferred than i by any customer-type in $T(i)$. As a result, we can compute the expected revenue generated by their purchase of product i. In addition, since i is more preferred than any product that is stocked at some later point of the recursion, the customer-types $T(i)$ can be discarded from this point on. Thus we represent the reward-to-go as the optimal expected revenue associated with the *residual* subproblem $(S \setminus \{i\}, T \setminus T(i))$. In the graph representation, the decision to include i in the assortment corresponds to removing node i and its adjacent edges from the graph as well as deleting all nodes in $T(i)$ and their adjacent edges. Due to these graph operations, the residual subgraph $G[S \setminus \{i\}, T \setminus T(i)]$ is potentially not connected anymore. By Lemma 1, the subproblem can be broken-down according to the connected components partition. If $(S_u^+, T_u^+)_{u \in [r(+)]}$ are the resulting subproblems, the expected revenue is $P_i \cdot \sum_{j \in T(i)} \lambda_j + \sum_{u=1}^{r(+)} J(S_u^+, T_u^+)$.

Case 2: product i is not stocked. All the customers in T remain unsatisfied and the reward-to-go is that associated with subproblem $(S \setminus \{i\}, T)$. The corresponding graph operation is the deletion of node i and its outgoing edges, and consider the residual subgraph $G[S \setminus \{i\}, T]$. Let $(G[S_u^-, T_u^-])_{u \in [r(-)]}$ describe the connected components of the residual subgraph. Then, by Lemma 1 it follows that the expected revenue is $\sum_{u=1}^{r(-)} J(S_u^-, T_u^-)$.

Combining the two decisions, the dynamic programming recursion is:

$$J(S,T) = \max \left[P_i \cdot \sum_{j \in T(i)} \lambda_j + \sum_{u=1}^{r(+)} J(S_u^+, T_u^+) \ , \ \sum_{u=1}^{r(-)} J(S_u^-, T_u^-) \right] . \tag{1}$$

State space collapse. In a naive implementation of the algorithm, one could solve the problem for all possible pair (S,T). However, this approach is inherently intractable and could be in the worst case as bad as 2^{N+K}. However, the dynamic program does not need to examine all corresponding subproblems to solve the initial instance. In fact, the recursion formula provides an algorithmic procedure to determine the precise 'minimal' number of subproblems needed to be solved. In contrast to a standard dynamic program, we will not assume that the state space is known a priori, and carefully generate a computational tree by processing the products from 1 to N (i.e., according to the unique order σ) adding nodes to the tree based on the recursion described above. The algorithm requires a two-pass implementation: first the computational tree is built by generating all subproblems of interest, using the recursive formula (1), then an optimal assortment is obtained by a backward induction.

Complexity analysis. We let $\mathcal{S}$ denote henceforth the exact state space that proceeds from the previous observation. Namely, $\mathcal{S}$ represents a collection of distinct subproblems, each of which belongs to the computational tree generated by the recursion. We now argue that the running time complexity is $O(NK \cdot |\mathcal{S}|)$. Indeed, building each node of the computational tree requires at most $O(NK)$ operations. This is the number of operations required to update the graph, compute the new connected components in $O(NK)$ operations and check whether each new subproblem already belongs to the computational tree in $O(N+K)$ operations using an appropriate search data structure, where each subproblem is encoded by an $N+K$-binary string. Then, the subproblems are solved backwards using the recursive formula with a total running time complexity of $O(K \cdot |\mathcal{S}|)$, taking $O(K)$ operations at each step to solve Equation (1). As a result, the complexity analysis boils down to estimating the size of the state space $\mathcal{S}$.

In the worst case, the number of connected subgraphs is still exponential. However, we establish in the next theorem that the state space is at most $\min(2^N, N \cdot 2^K)$, instead of the naive 2^{N+K}. Hence, the algorithm is efficient for applications in which the distribution over preference lists has a sparse support. It is worth noting that the running time is polynomial for $K = O(\log(N))$. Also, for $K = O(1)$, the running time is quadratic in the number of products, instead of the brute force approach in time $O(N^K)$.

THEOREM 1. *The size of the state space is at most* $\min(2^N, N \cdot 2^K)$. *The running time complexity is quadratic in* N *for a constant number of types* K*, and polynomial for* $K = O(\log(N))$.

To the end of proving Theorem 1, we introduce a characterization of the state space, which considerably simplifies the analysis of the algorithm. We define a projection Φ from the collection of subproblems $\mathcal{S}$ (i.e., subproblems in the computational tree) onto the collection of subsets of $\{1 \dots N\}$ as follows:

$$\begin{aligned}\Phi : \mathcal{S} &\to \mathscr{P}([N]) \\ (S,T) &\mapsto [\min(S)] \bigcap \left(\bigcup_{u \in T} C_u \right) .\end{aligned}$$

We establish below that Φ is injective, meaning that the size of the state space is equal to $|\Phi\langle\mathcal{S}\rangle|$.

PROPOSITION 1. *Φ is injective, and as a result: $|\mathcal{S}| = |\Phi\langle\mathcal{S}\rangle|$.*

Proof We assume (S_1, T_1), (S_2, T_2) are two subproblems that are generated by the recursion, such that $\Phi(S_1, T_1) = \Phi(S_2, T_2)$. By construction, $G_1 = G[S_1, T_1]$ and $G_2 = G[S_2, T_2]$ are two connected subgraphs.

Because G_1 is connected, there exists $u \in T_1$ such that $(\min(S_1), u)$ is an edge of G_1, meaning that $\min(S_1) \in C_u$. As a result, $\min(S_1) = \max(\Phi(G_1))$. By symmetry, we obtain:

$$\min(S_2) = \max(\Phi(G_2)) = \max(\Phi(G_1)) = \min(S_1) \ .$$

We infer from the connectivity of the subgraph $G[S_1, T_1]$ that $S_1 \subseteq \cup_{u \in T_1} C_u$. Since the set of products examined at previous steps of the recursion is exactly $[\min(S_1) - 1]$, we infer the equality $S_1 = \cup_{u \in T_1} (C_u \cap [\min(S_1), N])$. By a symmetric argument, $S_2 = \cup_{u \in T_2} (C_u \cap [\min(S_2), N])$.

As a result, what remains to be proven is simply that $T_1 = T_2$. Assume ad absurdum that $T_2 \setminus T_1 \neq \emptyset$ and let $u' \in T_2 \setminus T_1$. Under this assumption, we establish the following property.

CLAIM 1. *$C_{u'} \cap S_1 = \emptyset$.*

Proof of Claim 1 We assume otherwise and prove a contradiction. Because the two subgraphs G_1, G_2 both contain product node $\min(S_1)$, they initially lied in the same connected component of G. As a result, by looking at the sequence of algorithm iterations that generates G_1, we can define i as the minimal product examined by the algorithm after which u' gets disconnected from G_1. Then, product i has necessarily been added to the assortment, while node u' has been removed from the graph. Indeed, otherwise u' would still be connected to S_1 by hypothesis. Therefore, we obtain that $i \in C_{u'}$. It follows that $i \in \Phi(S_2, T_2)$ and thus $i \in \Phi(S_1, T_1)$. On the other hand, it is clear that i does not belong to $\cup_{u \in T_1} C_u$ otherwise some customer-types in T_1 would be discarded when i is selected in the assortment. This yields the desired contradiction. $\square$

The latter claim implies that there exists no edge between customer-types in $T_2 \setminus T_1$ and product nodes $S_1 \cap S_2$. In addition, there exist no edges between customer-type nodes $T_2 \cap T_1$ and product nodes $S_2 \setminus S_1$. Indeed, there would exist otherwise $u \in T_1$ and $i \in C_u \cap S_2$, such that $i \notin S_1$. Because

$\min(S_2) = \min(S_1)$ we infer that $i \in C_u \cap [\min(S_1), N]$. By construction of our recursion, we obtain $i \in S_1$, which gives a contradiction. To conclude, as shown in Figure (1b) we observe that $G[S_2 \setminus S_1, T_2 \setminus T_1]$ and $G[S_2 \cap S_1, T_2 \cap T_1]$ are distinct connected components of $G[S_2, T_2]$, contradicting the connectivity of the latter subgraph. $\square$

Finally, we derive a parametric bound on the state space, as a function of the consideration sets *diameter*. To this end, we define $\mathrm{Diam}(\mathcal{C})$ as the maximal diameter of a consideration set in $\mathcal{C}$: $\mathrm{Diam}(\mathcal{C}) = \max\{\max C - \min C : C \in \mathcal{C}\}$. The next claim comes as an immediate consequence of Proposition 1.

COROLLARY 1. *The size of the state space is at most* $2^{\mathrm{Diam}(\mathcal{C})}$. *Hence, the running time complexity is polynomial when* $\mathrm{Diam}(\mathcal{C}) = O(\log(N))$.

4. Modeling the Consideration Sets

In this section, we identify several consideration set models that stem from behavioral assumptions, for which $|\mathcal{S}|$ is polynomial in the input size.

4.1. Induced Intervals Structure

DEFINITION 1. A collection of consideration sets $\mathcal{C}$ is a family of *induced intervals* if it forms a collection of intervals when numbered according to some arbitrary permutation $\pi : [N] \to [N]$, i.e., $\pi\langle C_j \rangle$ is an interval for any customer-type $j \in [K]$.

Using the screening rule formalism of Section 2, it can be verified that this property arises when the consideration sets are formed as a conjunction of two screening rules, meaning that each consideration set in $\mathcal{C}$ is of the form $\{i \in [N] : x_1^{(i)} \geq t_1 \wedge x_2^{(i)} \geq t_2\}$ for some cut-off levels t_1, t_2, and the corresponding features are inversely related, i.e., for any products $i_1, i_2 \in [N]$, $x_1^{(i_1)} \geq x_1^{(i_2)}$ implies that $x_2^{(i_1)} \leq x_2^{(i_2)}$. As a practical example, *price* and *quality* are significant drivers of the customers' choices, who might use the following screening rules:

- *Budget constraint:* Customers would eliminate at an early stage of the purchasing process the products that they cannot afford (Fisher and Vaidyanathan 2009, Jagabathula and Rusmevichientong 2016).
- *Perceived quality cut-off:* There is empirical evidence that price is used as a cue for quality (Zeithalm 1988, Posavac et al. 2005), hence customers would eliminate all products cheaper than the given cut-off level.

The consideration sets emanating from a conjunction between budget constraints and perceived quality cut-offs are intervals with respect to the price order. Also, it is worth noting that induced interval consideration sets with unique-ranking subsumes the *downward substitution* model proposed by Pentico (1974) and Honhon et al. (2012) as the special case where the preference order

$\Sigma = \{\sigma\}$ coincides with $\{\pi\}$. In contrast, for the induced intervals in question, the preference order σ is generally distinct from the inducing permutation π. We now prove that the dynamic programming algorithm runs in polynomial time under this class of distributions, by bounding the number of connected subgraphs generated along the dynamic program. Intuitively, our counting argument utilizes the observation that a union of overlapping intervals is itself an interval.

THEOREM 2. *Under induced intervals consideration sets, the dynamic program has a running time of $O(N^4K)$. Using a specific data-structure, the running time complexity is $O(N^2K \cdot \log(K))$ in the special case of the downward substitution model.*

Proof Given that the function Φ is injective according to Proposition 1, it is sufficient to upper bound $|\Phi\langle\mathcal{S}\rangle|$. To this end, we let (S,T) designate a subproblem of $\mathcal{S}$. The key observation is that, due to the connectivity of $G[S,T]$, the union of the consideration sets in T is itself an interval according to the ordering π. Indeed, assume ad absurdum that there exists products $i,j \in S \cap [N]$ and a product $\alpha \in [N] \setminus S$ such that $\pi(i) < \pi(\alpha) < \pi(j)$. Then, for any customer-type j in T, since $\pi\langle C_j\rangle$ is an interval, we infer that either $\pi\langle C_j\rangle \subseteq [\pi(\alpha)-1]$ or $\pi\langle C_j\rangle \subseteq [\pi(\alpha)+1,N]$. Denoting by T_1 the customer-types that satisfy the former inclusion, and T_2 the latter one, we conclude that the subgraph $G[S,T]$ decomposes into distinct connected components $G[S_1,T_1]$ and $G[S_2,T_2]$, where S_1 is the subset of products whose π-indices belong to $[\pi(\alpha)-1]$ and S_2 corresponds to the π-indices in $[\pi(\alpha)+1,N]$. This contradicts the connectivity of $G[S,T]$.

Since $\Phi(S,T) = [\min(S)] \cap \cup_{j\in T} C_j$, and we have proven that $\pi\langle\cup_{j\in T} C_j\rangle$ is an interval, we conclude that the image of Φ is a collection at most N^3 distinct subsets of $[N]$. For the special case of downward substitution, i.e., $\Sigma = \{\pi\}$, $\Phi(S,T)$ is an interval of the form $[\alpha, \min(S)]$, meaning that the state space has a cardinality of $O(N^2)$. In this case, using an interval tree data-structure, it is known that the connected components can be computed in a running time of $O(K\log(K))$ (Samet 1990), leading to a total running time of $O(N^2K\log(K))$. $\square$

4.2. Laminar properties

DEFINITION 2. A collection of consideration sets $\mathcal{C}$ is said to be *laminar* if, for any two customer-types $j,j' \in [K]$ such that $C_j \cap C_{j'} \neq \emptyset$, the consideration sets are nested, i.e., $C_j \subseteq C_{j'}$ or $C_{j'} \subseteq C_j$.

Elimination-by-Aspect. Such laminar structures arise in Elimination-by-Aspect (EBA) choice-making processes, which were first introduced by Tversky (1972a,b). To this end, we assume that the feature space is discrete, i.e., without loss of generality $x^{(i)} \in \{0,1\}^d$, and each screening rule on feature $e \in [d]$ is expressed as a constraint of the form $x^{(i)}_e = t$ with a cut-off level $t \in \{0,1\}$. EBA models assume that a customer picks features iteratively, entailing a random sequence $e_1,\ldots,e_M \in [d]^M$, where M is random. At each step $k \in [M]$, the customer selects a level t_{e_k}, and eliminates all products i not satisfying $x^{(i)}_{e_k} = t_{e_k}$. The sequence of features could be deterministic (this is known as the lexicographic order) or random. One probabilistic structure used to describe these processes in related models rests on a tree structure (Tversky and Sattath 1979): the next feature chosen by an individual in the sequence of eliminations is *deterministic* conditional to the prefix of levels that he chose prior. That is, e_k is a deterministic function of $(e_1, t_{e_1}), \ldots, (e_{k-1}, t_{e_{k-1}})$. Assuming this property, it can be verified that the corresponding distributions over consideration sets necessarily have a laminar support (see Figure 2 for a pictorial representation).

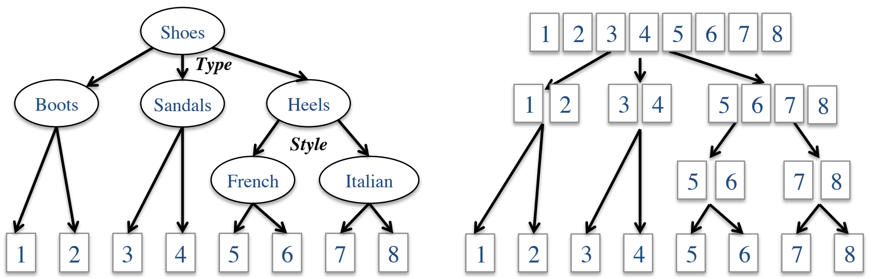


Figure 2 An Elimination-by-Aspect screening process and the corresponding laminar tree, for the example of purchasing 'shoes', with two features: 'type' and 'style'.

The following theorem suggests that the induced intervals structure defined in Section 4.1 is rather general as it subsumes laminar consideration sets as a special case.

Theorem 3. *The class of laminar consideration sets is a special case of induced intervals. The corresponding running time complexity is $O(N^2K^2)$.*

Proof Let $\mathcal{C}$ denote a laminar consideration set system. Without loss of generality, we may assume that each consideration set corresponds to a unique customer-type (otherwise if two customer-types share the same consideration set, we represent them by a single type and aggregate their arrival probabilities) and there exists a consideration set comprised of all products $[N]$ in $\mathcal{C}$ (its arrival probability can be set to 0). We seek to prove that $\mathcal{C}$ is a family of intervals if the products are numbered according to some appropriate permutation $\pi : [N] \to [N]$.

Laminar tree. It is known that any laminar collection of subsets admits a rooted tree representation (Edmonds and Giles 1977). That is, we can build an directed tree (V, E), wherein each customer-type is represented by a single node, i.e., $V = [K]$, and there exists a directed edge $(j,k) \in E$ if k is the customer-type with a maximal consideration set contained in C_j. In other terms, we have $C_k \subset C_j$ and there exists no other $l \in [k]$ such that $C_k \subset C_l \subset C_j$. The root corresponds to the customer-type with consideration set $[N]$.

Depth first order. Now, for any $j \in K$, we define $o(j)$ as the offspring of node j in (V, E). Also, we introduce the list of products $s(j)$ formed by the difference between C_j and the products associated with the children of j, i.e. $s(j) = C_j \setminus \cup_{j' \in o(j)} C_{j'}$. Next, the permutation π is defined through the ranked list of products obtained as a concatenation of the lists $s(j)$ in a depth-first traversal of the laminar tree. It can be proven inductively that $\pi\langle C_j\rangle$ is an interval for any $j \in [K]$. Indeed, if a node j is a leaf of the laminar tree, then $s(j) = C_j$, and the concatenation procedure preserves the connectivity of $s(j)$. The inductive argument proceeds from the following observations.

1. By definition of s, for any given customer-type $j \in [K]$, $C_j = (\cup_{j' \in o(j)} C_{j'}) \cup s(j)$.
2. The collections of products associated with the children nodes are examined consecutively in a depth-first traversal. Hence, the inductive hypothesis implies that $\pi\langle \cup_{j' \in o(j)} C_{j'}\rangle$ is an interval.
3. Since this interval gets concatenated to $s(j)$, observation 1 above implies that $\pi\langle C_j\rangle$ is an interval. $\square$

Weakly laminar consideration sets. To demonstrate the generality of our algorithmic approach, we now introduce an extension of this model, dubbed *weakly laminar* consideration sets, which does not reduce to induced intervals, but still admits a polynomial running time guarantee.

Definition 3. A collection of consideration sets $\mathcal{C}$ is said to be *weakly laminar* if any two consideration sets that intersect are nested up to the maximal product of their intersection. That is, for any customer-types $a, b \in [K]$ such that $C_a \cap C_b \neq \emptyset$, if $i = \max(C_a \cap C_b)$, then, either $C_a \cap [i] \subseteq C_b \cap [i]$ or $C_b \cap [i] \subseteq C_a \cap [i]$.

This model captures the *conjunction* of any laminar consideration sets with any arbitrary screening rule, such as the budget and quality constraints mentioned in Section 4.1. In addition, it subsumes (strictly) other choice models in related literature, notably the above-mentioned *downward substitution model*, as well as the *out-tree* model proposed by Honhon et al. (2012).

Theorem 4. *Under weakly laminar consideration sets, the dynamic program runs in time $O(N^2K^2)$.*

Proof To analyze the size of the state space $|\mathcal{S}|$ under this model, we first exhibit a structural property satisfied by the recursion, namely the existence of a 'maximal' consideration set with respect to some well-chosen inclusion order, in each connected subgraph examined by the recursion.

Lemma 2. *Assume that $(S, T) \in \mathcal{S}$ is a subproblem generated by the recursion. Then, there exists a customer-type $j^* \in T$ such that $C_j \cap [\min(S)] \subset C_{j^*} \cap [\min(S)]$ for any customer-type $j \in T$.*

This property is proven in Appendix A. Consequently, we can upper bound $|\Phi\langle\mathcal{S}\rangle|$. Let (S, T) be a subproblem in the state space $\mathcal{S}$. By Lemma 2, we obtain

$$\Phi(S,T) = \bigcup_{j \in T} \left(C_j \bigcap [\min(S)]\right) = C_{j^*} \bigcap [\min(S)]$$

For a fixed value of $\min(S)$, there are at most K subsets $\Phi(S, T)$, meaning that $|\Phi\langle\mathcal{S}\rangle| \leq NK$. $\square$

4.3. Disjunctive consideration sets

The consideration set models discussed in the previous sections proceed from a conjunction of screening rules. As mentioned in Section 2, another decision-making model proposed in the marketing literature posits that the consideration sets are formed in a *disjunctive* fashion.

Definition 4. For any $d \in \mathbb{N}$, a collection of consideration sets $\mathcal{C}$ is said to be *d-disjunctive* if the feature space is d-dimensional and all consideration sets in $\mathcal{C}$ are generated as a disjunction of screening rules. That is, each customer-type $j \in [K]$ is characterized by a cut-off vector denoted by $\mathbf{t}^{(j)} \in \mathbb{R}^d$, such that

$$C_j = \left\{ i \in [N] : \left(x_1^{(i)} \geq t_1^{(j)} \right) \vee \cdots \vee \left(x_d^{(i)} \geq t_d^{(j)} \right) \right\} \ .$$

We now prove that the size of the state space $|\mathcal{S}|$ is polynomially bounded for a fixed parameter d. Since the d-disjunctive model can replicate arbitrary consideration set structure $\mathcal{C}$ for a large enough d, the next theorem expresses an explicit tradeoff between modeling power and tractability. In practice, one would expect that customers make use of few screening rules (Hauser et al. 2009).

Theorem 5. *Under d-disjunctive consideration sets, the dynamic program has a running time of* $O(N^2 K^{d+1})$.

5. The General Dynamic Program

In this section, we relax the assumption that Σ is a singleton and describe a dynamic program that applies to arbitrary preference list distributions. The key ingredients of the algorithm remain unchanged. Specifically, products are processed sequentially, which entails a decomposition of the graph representation into increasingly refined connected components, in a divide-and-conquer fashion. However, unlike the unique-ranking case, the processing order does not necessarily coincide with the customer's preference order. Thus, it is not immediate which subset of customer-types gets allocated to a given product at the time a DP decision is made. As a result, the DP action space needs to be enlarged to account for any feasible allocation of a product to a subset of customer-types. At face value, there are exponentially many potential allocations, and the approach appears to be subject to the curse of dimensionality. We work around this difficulty by proposing an auxiliary algorithm, used at each step of the recursion, that can yield tractable solutions.

Processing order. We begin by defining a *processing order* σ on the products, according to which the dynamic program makes sequential decisions (or actions). The correctness of the dynamic program does not depend on σ although, as shown in next subsections, an appropriate choice of σ may significantly reduce the running time complexity. Here, σ is chosen as an arbitrary permutation and the products are numbered accordingly (i.e., product 1 is processed first, and so on) such that the reference to the processing order is made implicit throughout the present section.

State space and value function. The state space is described by the parameters $(S, T, \boldsymbol{L})$, where S is a subset of products in $[N]$, T is a subset of customer-types in $[K]$ and $\boldsymbol{L} \in \mathbb{Z}_{+}^{K}$ is a nonnegative integer valued vector, named the *truncation vector*. We let $J(S, T, \boldsymbol{L})$ be the maximum expected revenue that can be attained from customer-types in T using an assortment within products in S, and assuming that, each customer-type $j \in T$ is willing to purchase only products of rank at most L_j within his consideration set. That is, customer-type j will only purchase products in the set $C_j(L_j) = \{i \in C_j : \sigma_j(i) < L_j\}$. (Recall that each customer-type j is associated with a ranking function $\sigma_j \in \Sigma$.) We note that only the T-coordinates of $\boldsymbol{L}$, i.e., the sub-vector $\boldsymbol{L}[T]$, matter in the definition of $J(S, T, \boldsymbol{L})$. However, to lighten the notation we use the entire vector and assume that the unnecessary coordinates are set to 0.

DP bipartite graph. Similar to the unique-ranking case, we define the bipartite graph G that has a node, for each product $i \in [N]$, on one side, and a node, for each customer-type $j \in [K]$, on the other side. There is an edge between a product node and customer-type node if the preference list of the latter contains the former. Each subproblem of the state space $(S, T, \boldsymbol{L})$ is associated with the subgraph $G_{\boldsymbol{L}}[S, T]$ with *(i)* product nodes in S; *(ii)* customer-type nodes in T; *(iii)* there exists an edge between any $i \in S$ and $j \in T$ if $i \in C_j(L_j)$. Similar to the unique-ranking case, the connected components of the subgraph capture a decomposition into independent instances. The proof, in the same spirit as that of Lemma 1, is omitted.

Lemma 3. *For each subproblem $(S, T, \boldsymbol{L})$, assuming that the connected components of $G_{\boldsymbol{L}}[S, T]$ are described by the collection of subgraphs $(G_{\boldsymbol{L}}[S_u, T_u])_{u \in [r]}$ where S_u denotes a subset of product nodes in S and T_u is a subset of type nodes in T, then $J(S, T, \boldsymbol{L}) = \sum_{u=1}^{r} J(S_u, T_u, \boldsymbol{L})$.*

Dynamic programming decisions and graph operations. We consider a subproblem $(S, T, \boldsymbol{L})$ and i is the next product to be processed $i = \min(S)$. We define $T(i) \subseteq T$ as all customer-types whose preference list contains i, i.e., $T(i) = \{j \in T : i \in C_j(L_j)\}$.

Assume we decide to allocate product i to a subset of customers $V \subseteq T(i)$, meaning that i is the most preferred product made available to the customer-types in V. We describe below some natural operations on the graph $G_{\boldsymbol{L}}[S, T]$ to enforce the decision of allocating product i to customer-types V. In particular, we make sure that the decision to satisfy V with product i is consistent with future decisions, and that it is feasible irrespective of the subsequent decisions. Specifically, we perform the following operations on the bipartite graph:

1. *T-nodes deletion.* Because they are already satisfied with a product, the nodes corresponding to consumer-types V should be discarded, we define $T^{(V)} = T \setminus V$ as the remaining customer-types.

2. *S-nodes deletion.* For each satisfied customer-type $j \in V$, we should remove from S the nodes of all items he prefers more; indeed, these products cannot be stocked in the assortment otherwise they would have been chosen by customer-type j over product i. Thus we define $S^{(V)}$ as the residual set of products:

$$S^{(V)} = S \setminus \bigcup_{j \in V} \{x \in C_j(L_j) : \sigma_j(x) < \sigma_j(i)\} \ , \tag{2}$$

3. *Edges deletion.* Finally, if a customer-type whose preference list contains product i, is not allocated to this product, he can only purchase a product product he prefers more at some later point of the recursion. As a result, we need to truncate his preference list by updating the vector $\boldsymbol{L}$:

$$\forall j \in T^{(V)}, \quad L'_j = \begin{cases} L_j & \text{if } j \notin T(i) \\ \min(L_j, \sigma_j(i)) & \text{otherwise} \end{cases} . \tag{3}$$

We can easily verify the correctness of the above graph operations. That is, the expected revenue obtained by summing the immediate-reward, formed by the allocation of product i to customer-types V, with the reward-to-go, generated by the subproblem associated with the residual subgraph $G_{\boldsymbol{L}'}[S^{(V)}, T^{(V)}]$, is feasible. We now formally describe the recursion.

Base case. If we set $S = [N]$, $T = [K]$ and $L_j = N + 1$, then $J(S, T, \boldsymbol{L})$ reflects the original problem we are interested to solve. Using the Lemma 3, an optimal assortment is obtained by solving independently the subproblems associated with each connected component of G. From this point on, connectivity is an invariant of the subproblems examined by the recursion.

Recursive formula. We consider the subproblem $(S, T, \boldsymbol{L})$ such that $G_{\boldsymbol{L}}[S, T]$ is a connected subgraph. Recall that i denotes the next product to be processed (the minimal element of S) and $T(i)$ are all customer-types who consider product i. The decision made by the dynamic program consists in the subset of customer-types $V \subseteq T(i)$ allocated to product i. Without loss of generality, we can assume that an empty allocation $V = \emptyset$ means that he product i is not stocked in the assortment. In this case, the residual subgraph is $G_{\boldsymbol{L}}[S \setminus \{i\}, T]$, which decomposes into the connected components $(G_{\boldsymbol{L}}[S_u, T_u])_{u \in [r]}$. Each corresponding subproblems is solved independently according to Lemma 3, generating a total revenue of $\sum_{u=1}^{r} J(S_u, T_u, \boldsymbol{L})$.

For each choice of $V \subseteq T(i)$, where $V \neq \emptyset$, the allocation generates an immediate-reward $P_i \cdot \sum_{j \in T} \lambda_i$. Next, we consider the residual subgraph $G_{\boldsymbol{L}'}[S^{(V)}, T^{(V)}]$ after performing the operations previously described. Namely, we remove the most preferred products according to Equation (2) while we delete edges according to $\boldsymbol{L}'$ defined in Equation (3), where the vector $\boldsymbol{L}'$ does not depend on the choice of the allocation $V \subseteq T(i)$. Using Lemma 3, the subgraph $G_{\boldsymbol{L}'}[S^{(V)}, T^{(V)}]$ can be decomposed into its connected components $(G_{\boldsymbol{L}'}[S_u^{(V)}, T_u^{(V)}])_{u\in[r(V)]}$. Therefore, the optimality conditions yield the following the recursive formula:

$$J(S,T,\boldsymbol{L}) \;=\; \max\left(\sum_{u=1}^{r} J(S_u, T_u, \boldsymbol{L}) \;,\; \max_{V\subseteq T(i)} \;\; P_i \cdot \sum_{j\in V} \lambda_j \;+ \sum_{u=1}^{r(V)} J\left(S_u^{(V)}, T_u^{(V)}, \boldsymbol{L}'\right) \right) \tag{4}$$

Preliminary complexity analysis. In a naive implementation of the algorithm, one would solve the problem for all possible tuple $(S,T,\boldsymbol{L})$. Similar to the dynamic program presented in Section 3, the effective computational tree is in fact comprised of a much smaller fraction of the state space. However, in contrast to the unique-ranking case, the recursive formula (4) describes a maximization problem over exponentially many allocations $V \subseteq T(i)$, each associated by a family of descendant subproblems of the form $(S_u^{(V)}, T_u^{(V)}, \boldsymbol{L}')$. As a result, we cannot readily leverage this formula to build the computational tree. In addition, even if one tightly characterizes the state space, it is still not obvious how to solve efficiently the optimization problem described by (4).

The marginalized algorithm. We address the difficulty raised at the end of the previous section by proposing an efficient algorithm to generate the computational tree and solve the recursive formula (4), while avoiding an enumeration over all allocations $V \subseteq T(i)$. Because the detailed exposition is rather technical, we only provide the high-level idea, and state the resulting complexity analysis. The specifics of this auxiliary algorithm are detailed in Appendix B. The crux of our approach is to observe that the descendant subproblems in equation (4) are not necessarily distinct and may very well be equivalent: there is potentially overlap in the offspring generated across all choices of $V \subseteq T(i)$. Most of the redundancy that happen in a brute force enumeration can in fact be eliminated (up to a polynomial factor) by *marginalizing* the allocation decision. That is, the choice of the allocation $V \subseteq T(i)$ is broken-down into a sequence of binary decisions, each of which applies to a single customer-type in $T(i)$. These binary allocation decisions are made sequentially, entailing refined connected subgraphs and potentially new descendant subproblems in the computational tree. By constructing and updating an appropriate data-structure, we avoid the unnecessary exploration of equivalent subproblems. As a result, the marginalized algorithm runs in time polynomial in N, K and $|\mathcal{S}|$, yielding the following complexity result.

PROPOSITION 2. *The running time complexity of the marginalized dynamic program is polynomial in the size of the state space $\mathcal{S}$ and the input size. In addition, in the worst case, we have $|\mathcal{S}| \leq N \cdot 2^N$.*

6. Consider-then-Choose Models with Ranking Heterogeneity

We now investigate models which combine consideration set heterogeneity along with ranking heterogeneity, thereby extending the computational settings studied in Section 4.

6.1. Similar rankings

In this section, we relax the unique-ranking assumption, and derive parametric computational bounds for the consideration set models studied in Section 4 when the rankings are similar, i.e., Σ is formed by small perturbations of a central permutation σ. Namely, assuming that S_N designates the set of all permutations of $[N]$, we let $B(\sigma,d)$ designate the L_∞-ball of radius d centered on σ. That is,

$$B(\sigma,d) = \{\sigma' \in S_N : \forall i \in [N] |\sigma'(i) - \sigma(i)| \leq d\} \ .$$

This definition implies that for any permutations $\sigma_1, \sigma_2 \in B(\sigma,d)$, two products $i,j \in [N]$ such that $|\sigma(i) - \sigma(j)| \geq 2d$ necessarily have the same relative order in σ_1 and σ_2. In other terms, only local 'swaps' may occur between products at distance less than $2d$. This structure is somewhat similar to the d-sorted pricing structure proposed by Jagabathula and Rusmevichientong (2016).

The next theorem, proven in Appendix C, asserts that, for a fixed parameter d, the state space complexity associated with unique-ranking $\Sigma = \{\sigma\}$ is preserved up to a polynomial factor under the generalization $\Sigma = B(\sigma,d)$. In particular, the polynomial running time guarantees established in Section 4 carry over to $\Sigma = B(\sigma,d)$. Again, this result permits a parametric tradeoff between modeling power and tractability.

Theorem 6. *Let $\mathcal{S}(\mathcal{C},\sigma)$ denote the state space under a collection of consideration sets $\mathcal{C}$ and a unique-ranking function $\Sigma = \{\sigma\}$. Then, the size of the state space of the general dynamic program with processing order σ under the consideration sets $\mathcal{C}$ with rankings $\Sigma = B(\sigma,d)$ is at most $2^{4d-2} \cdot |\mathcal{S}(\mathcal{C},\sigma)|$.*

6.2. Quasi-convex preference lists

We now study a class of preference list distributions that allows for high levels of heterogeneity in the ranking decisions, but the ranking functions exhibit a *quasi-convex* structure.

Definition 5. Suppose that the product indices are numbered according to a *central* permutation σ over products. A distribution over preference lists belongs to the *quasi-convex model* if the consideration sets $\mathcal{C}$ are intervals and the ranking functions Σ are quasi-convex. That is, for all $j \in [K]$, there exists $i \in C_j$ such that σ_j is decreasing over $\{1,\dots,i\} \cap C_j$ and increasing over $[i,N] \cap C_j$.

The quasi-convex property captures several common preference patterns. To flesh out this model with practical examples, suppose that the consideration sets are formed as a conjunction of the screening rules relative to price and perceived quality described in Section 4.1. The quasi-convex family simultaneously captures a variety of ranking behaviors: customers can be price-driven, or quality-driven, or maximize any price/quality ratio function which is quasi-convex in price.

It is worth noting that the quasi-convex model subsumes the one-dimensional *locational choice model* (Lancaster 1966, 1975). In the latter model, customer-types and products are each represented by a scalar value, and a customer-type picks the closest product to him made available in the assortment (i.e., with minimal absolute distance between their respective scalars). It is not difficult to show that the ranking functions arising from this model are quasi-convex with respect to the central permutation formed by increasing scalars.

Observe that the quasi-convex model substantially 'enriches' the degree of freedom of the distributions up to $O(2^N)$ - in comparison to the $O(N^2)$ parameters of the intervals model or the $O(N^3)$ parameters associated with the locational model (see Claim 2 proven in Appendix D).

THEOREM 7. *Under the quasi-convex model with central permutation σ, the dynamic program with processing order σ has a state space of size $O(N^3)$.*

Proof We construct an injective mapping from any connected subgraph generated along the recursion, i.e., belonging to the computational tree, onto 3-tuples of products. Specifically, Ψ maps any subproblem $(S, T, \boldsymbol{L})$ to the tuple (a, b, c) where (a, b) is the ordered pair of the last products stocked along the recursion before generating $(S, T, \boldsymbol{L})$ while c is the next product to be processed in S, i.e., $c = \min(S)$. To prove that this mapping is injective, we consider $(S_1, T_1, \boldsymbol{L})$ and $(S_2, T_2, \boldsymbol{L}') \in \mathcal{S}$ two subproblems of the computational tree such that

$$(a, b, c) = \Psi(S_1, T_1, \boldsymbol{L}) = \Psi(S_2, T_2, \boldsymbol{L}') \ . \tag{5}$$

Now, assume ad absurdum that $T_1 \setminus T_2 \neq \emptyset$. Without loss of generality, we can pick a customer-type j in $T_1 \setminus T_2$ whose consideration set C_j has been truncated, meaning that $L_j < N + 1$. Indeed, $T_1 \setminus T_2$ would otherwise be comprised of customer-types not affected by the decisions made at the parent nodes of the computational tree, relative to products in $[b]$. But this contradicts that at least one node in $T_1 \setminus T_2$ gets disconnected from the subgraph $G_{\boldsymbol{L}'}[S_2, T_2]$ along the recursion.

Since $L_j < N + 1$, there exists a product in $C_j \cap [c - 1]$, which is processed before attaining subproblem $(S_1, T_1, \boldsymbol{L})$. By connectivity, $C_j(L_j) \cap S_1$ is not empty, meaning that C_j contains a product in $[c, N]$ as well. Given that C_j is an interval, this implies that $c \in C_j$. Also, the preference order between the products $\{a, b, c\}$ is given by $\sigma_j(c) < \sigma_j(b) < \sigma_j(a)$. Indeed, the ranking function being quasi-convex, any product with index larger than b would otherwise be less preferred than a and b, and as a result the truncated consideration set $C_j(L_j)$ would not intersect with S_1. This shows that customer-type j prefers product c over all products in $[b]$. Since $j \notin T_2$ and $c \in C_j$, the customer-type node j necessarily gets disconnected from the subgraph $G_{\boldsymbol{L}'}[S_2, T_2]$ along the recursion through an allocation to some product $i \in [b]$. However, had this allocation been decided at a parent node of the computational tree, all products that customer-type j prefers over i would have been discarded by now. In particular, product c would not belong to S_2, which contradicts equation (5).

Finally, we remark that the truncation of preference lists only depends on the last product being stocked. Indeed, as previously shown, for any remaining preference lists $j \in T_1$ such that $L_j < N+1$, then product b lies in C_j and $\sigma_j(b) < \sigma_j(a)$. Quasi-convexity induces that b is preferred over any other product stocked before a. Thus, $L_j = \sigma_j(b) = L'_j$. We conclude by observing that

$$S_1 = \bigcup_{j \in T_1} C_j(L_j) = \bigcup_{j \in T_2} C_j(L'_j) = S_2 \ . \quad \square$$

6.3. Two-feature compensatory model

We consider a preference list-based model where the screening rules are combined in a compensatory fashion (Einhorn and Hogarth 1975, Dawes 1979). Here, low levels on a given feature can be offset by high levels on other features as discussed in Section 2. Specifically, preference lists are formed according to utility maximization, as illustrated by Figure 3.

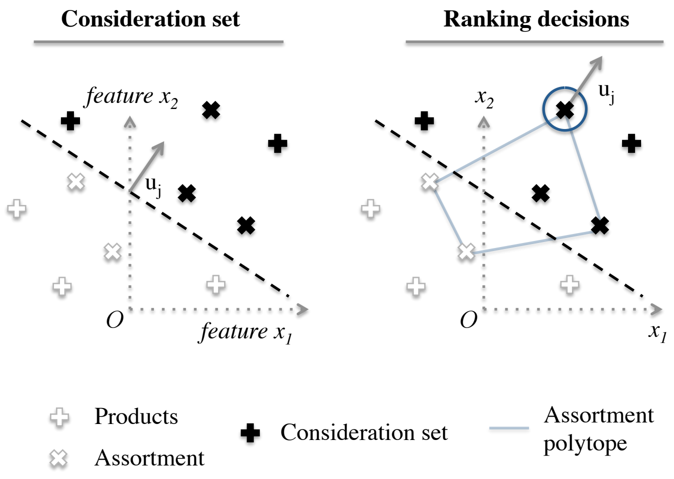


Figure 3 Consideration sets and ranking decisions driven by linear utility maximization in a two-featureal feature space.

Definition 6. Suppose that the feature space has dimension $d = 2$. An instance belongs to the two-feature compensatory model if each customer-type $j \in [K]$ can be described by a utility vector $\boldsymbol{u}^{(j)} \in \mathbb{R}^2$ and a cut-off level t_j such that *(i)* C_j contains all products which utility is above the cut-off t_j, i.e., $C_j = \{i \in [N] : \boldsymbol{u}^{(j)} \cdot \mathbf{x}^{(i)} \geq t_j\}$, and *(ii)* for any pair of products $i, k \in [N]$, $\sigma_j(i) < \sigma_j(k)$ if and only if $\boldsymbol{u}^{(j)} \cdot \boldsymbol{x}_k < \boldsymbol{u}^{(j)} \cdot \boldsymbol{x}_i$ (we assume there are no ties between products).

By exploiting the geometric structure of this model, we prove that the state space is of polynomial size under the class of preference list distributions described by the two-feature linear model. The proof, detailed in Appendix E, is of same spirit as that of Theorem 7: we construct an injective mapping of subproblems in $\mathcal{S}$ onto the last dynamic programming decisions in the computational tree.

THEOREM 8. *Under the two-feature compensatory model, for any arbitrary processing order, the size of the state space is of* $O(N^3K^2)$.

7. Empirical performance

In this section, we demonstrate that our approach yields superior predictive and computational performance against several benchmarks. In Section 7.1, we demonstrate that the dynamic programming approach is computationally efficient, even in comparison to a state-of-the art integer programmer solver. In Section 7.2, we show that the quasi-convex model has the ability to accurately replicate and predict the choice outcomes on synthetic and real industry datasets, against several parametric choice models.

7.1. Computational performance

7.1.1. Computational benchmark: integer programming. The assortment optimization problem can be formulated as $0-1$ binary program. We define the binary decision variables y_i to decide whether a product is added to the assortment, $x_{i,j}$ encodes the assignment of product $i \in C_j$ to customer-type $j \in [K]$. The problem is formulated as follows:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} P_i \cdot \lambda_j \cdot x_{i,j} && \\
\text{s.t.} \quad & x_{i,j} \le y_i && \forall (i,j) \in [N]\times[K] \qquad (6)\\
& x_{i,j} + y_l \le 1 && \forall j \in [K], l \in C_j \text{ and } i \in \{x \in C_j : \sigma_j(l) < \sigma_j(i)\} \qquad (7)\\
& \sum_{i \in C_j} x_{i,j} \le 1 && \forall j \in [K] \qquad (8)\\
& x_{i,j}, y_i \in \{0,1\}\ ,
\end{aligned}$$

where the coupling constraints (6) enforce that a customer may only pick a product made available in the assortment while the inequalities (7) ensure that a given customer-type could only choose the highest rank product made available to him. Finally, the constraints (8) mean that at most one product is assigned to each customer. The additional constraints (8) tighten the relaxation of the binary program. It is worth noting that that similar formulations were introduced prior to this work by McBride and Zufryden (1988) and Anupindi et al. (2009). This integer program (IP) is implemented on a commercial solver GUROBI (Gurobi Optimization 2015), which arguably combines state-of-the-art methodologies and implementation.

7.1.2. Computational set-up. The experiments are conducted using a MacBook Pro with processor 2.5 GHz Intel Core $i5$ (two cores). Our dynamic program is implemented using the programming language Julia. The commercial solver GUROBI (v.6.5) is run in parallel mode. We impose termination when the incumbent solution has an optimality gap of 1%, or after the running time reaches 1000 seconds for computational convenience. In contrast, our algorithm provides *exact solutions* for all instances. We run two series of experiments with different generative models. In the former, we generate instances of the quasi-convex model described in Section 6, arguably one of the 'richest' consider-then-choose model discussed in previous sections that admits a provable polynomial running time guarantee. In the latter, we compare the algorithms on generic instances with unique-ranking preferences, not pertaining to any specific structure of consideration sets. The consideration sets arise from i.i.d Bernoulli trials with a parameter $\alpha \in (0,1)$. In Appendix G, we describe more precisely our generative models and provide additional details on the implementation.

Table 2 Runtime of our algorithm (DP) against the commercial solver (IP) under the quasi-convex preference model.

Parameters		Average runtime (s)		Coeff. of var (%)	
N	K	DP	IP	DP	IP
50	500	0.9	45.9	17.0	47.4
50	1000	1.2	398.5	6.2	52.4
50	2000	1.8	777.1	$< 10^{-3}$	46.2
50	2500	2.4	$> 10^3$	$< 10^{-3}$	-
100	2500	16.8	$> 10^3$	$< 10^{-3}$	-
200	2500	138.9	$> 10^3$	$< 10^{-3}$	-

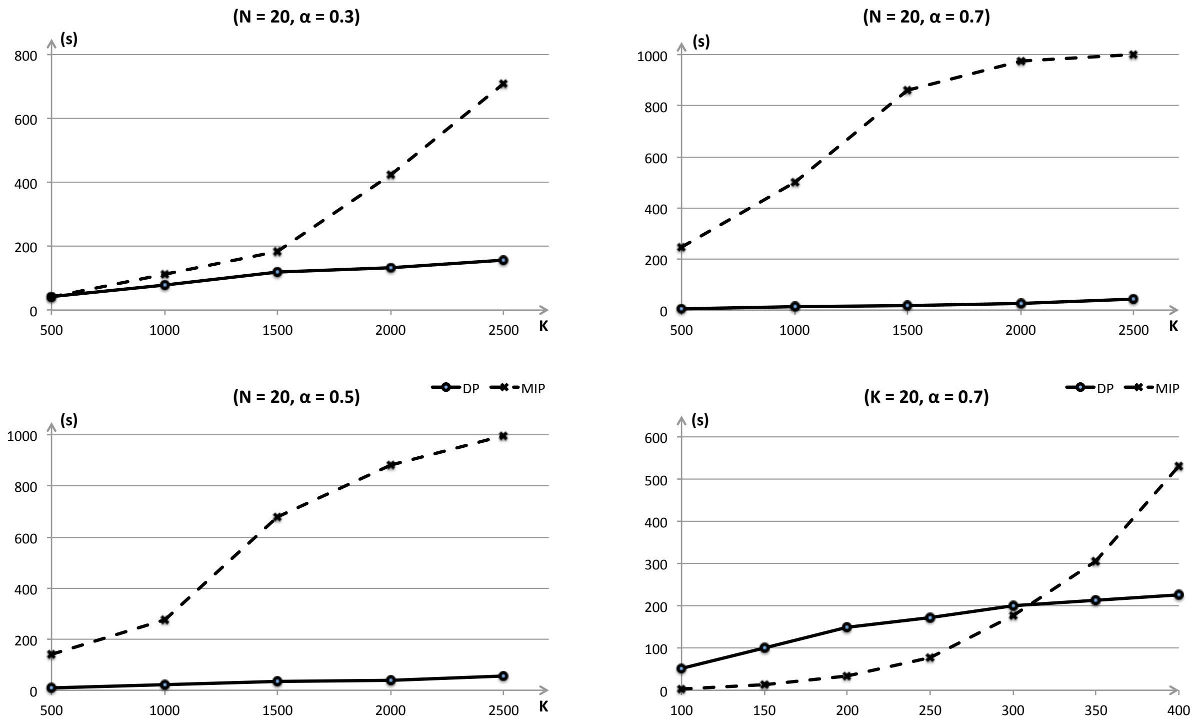
The IP is terminated after 1000 seconds. Each entry is obtained by sampling 50 instances unless the average running time exceeds 800 seconds, in which case we sample 20 instances.

7.1.3. Numerical results. Our numerical results indicate that our algorithmic approach substantially outperforms the IP solver in several regimes of parameters (see Table 2 and Figure 1). The dynamic programming approach shows more running time stability across instances due to its combinatorial nature, while one potential shortcoming of the IP approach for practitioners resides in the large variability of running time across instances.

Under the quasi-convex model. As shown in Table 2, the dynamic program dominates the commercial solver by an order of magnitude. The IP approach scales unfavorably with the number of customer-types K and it becomes intractable for large scale instances (e.g., with 200 products) where the dynamic program is still very efficient.

Under arbitrary consideration sets. Recall that in this general setting the problem is NP-hard even to approximate within $O(N^{1-\epsilon})$. In several cases, our approach dominates the IP as shown in Figure 4.

Figure 4 Average runtime of our algorithm (DP) against the commercial solver (IP) on synthetic instances.



Note. Note that the asymptotic complexity of the IP is not captured here since we impose termination after 1000 seconds. The running time is averaged over 50 instances. Recall that α is the Bernoulli parameter that controls the size of the consideration sets.

Similar to the quasi-convex model, the IP solver scales poorly with the number of customer types. That is, for a fixed number of products ($N = 20$), the running time of the IP is highly affected by the number of customer types K. The difference between the algorithms is more pronounced for larger consideration sets (larger Bernoulli parameter α). On the other hand, as one would expect, the dynamic program is less efficient when $N >> K$, since our algorithm enumerates over product stocking decisions. The results obtained for large consideration sets ($\alpha = 0.7$) suggest that the dynamic program could asymptotically dominate the IP solver in this regime.

The observed computational efficiency proceeds from the state space collapse performed by our algorithm. When comparing our approach to a “naive” recursion, the state space is reduced by a factor ranging between 75% to over 99% (see Table 5 in Appendix I).

7.2. Modeling Power

Practical applications of choice modeling, such as the assortment optimization problem studied here, begin with *transactional data*. The generic approach is to fit a specific type of choice model to this data and then employ an assortment optimization algorithm designed for that choice model. As such, the choice model employed must strike a balance between its ability to fit the data on the one hand, and admit efficient algorithms for assortment optimization on the other. In this regard it is well known that the MMNL model has the ability to represent any choice model satisfying the strong axiom of revealed preferences, that is, arbitrary distributions over preference lists (McFadden and Train 2000). Of course, this expressive power comes at a price: assortment optimization under the MMNL model is difficult in all but a restricted set of cases. Specifically, Désir and Goyal (2014) provide an algorithm for assortment optimization under the MMNL model whose complexity scales exponentially with the number of customer segments[2]. Consequently, optimization is practical only for a mixture over a relatively 'small' number of customer segments (a notion we will make precise shortly). In summary, one may regard MMNL models with a small number of customer segments as a valid alternative to the models (and corresponding algorithms) we consider in this paper. The goal of this section is to flesh out this comparison. Specifically, we consider the following experiments on synthetic and industry data:

1. *Synthetic data from an MMNL model:* Using a synthetic dataset generated from an MMNL model with a relatively large number of customer segments, we fit two types of models to this data: *(i)* an MMNL model with a small number of customer segments and *(ii)* the quasi-convex consider-then-choose model studied in Section 6.2. We show that in several cases the latter model provides a better fit to the data (out-of-sample) under a variety of metrics. In particular, we show that the quasi-convex model does an excellent job of capturing choice patterns that may have arise from an MMNL model.
2. *Synthetic data from a consider-then-choose model:* As a counterpart to synthetic data from an MMNL model, we consider fitting both types of model in the experiment above to synthetic data generated this time from a simple, intervals-based consider-then-choose model. As one would expect, the quasi-convex model provide a better fit by a large margin. In particular, we show that MMNL models with a small number of customer segment do not do an adequate job of fitting choice data arising from consider-then-choose behavior.
3. *Real industry data:* Using transactional data across a panel of hundreds of thousands of customers in three distinct product categories (containing hundreds of products) collected by an industry partner, we again run the same experiment, and evaluate predictive power on a hold-out sample. Again we show that the quasi-convex consider-then-choose model provides a very significant improvement in predictive accuracy on the hold out set. This improvement can be as high as 60% in certain categories, and never lower than 4% – a striking improvement.

In the sequel, we designate by MMNL(c) the class of mixtures with c customer segments. The MMNL instances (and the MNL as a special case) are parametrized by the preference weights

[2] Their algorithm constitutes a *fully polynomial-time approximation scheme* for a fixed number of customer segments

$w_{i,j} \in \mathbb{R}^+$ where $i, j \in [N] \times [c]$, along with the probability vector $(\mu_1, \ldots, \mu_c)$ of the mixture. Here N is the number of products and c is the number of customer segments. With this definition at hand, the purchase probability for product i in an assortment $\mathcal{A} \subseteq [N]$ is, under the MMNL(c) model, expressed as:

$$\Pr(i|\mathcal{A}) = \sum_{\alpha=1}^{c} \mu_\alpha \cdot \frac{w_{i,\alpha}}{1 + \sum_{j \in \mathcal{A}} w_{j,\alpha}} .$$

7.2.1. Synthetic data. In this section, we evaluate the predictive power of the quasi-convex model on synthetic datasets, against MMNL models with up to 3 customer segments. Assortment optimization with a larger number of segments is effectively impractical as noted earlier (the computational complexity scales exponentially with the number of segments). Against such benchmarks, we demonstrate the predictive power of the quasi-convex model. We first explain how the synthetic datasets are generated, and then describe our numerical results. The experiments are conducted on random instances, with a fixed number of products $N = 50$. The dataset is constructed by randomly generating 100 assortments $\mathcal{A}_1, \ldots, \mathcal{A}_{100}$, each formed by drawing N independent Bernoulli trials with probability of success 0.5. To generate the purchase probability data, we make use of the following ground truth models:

- *MMNL models:* For our first set of synthetic data, we generate random MMNL(5) instances. It is worth noting that assortment optimization against a 5 segment model is effectively impractical, and as such we will eventually fit an MMNL model with a smaller number of segments to this data. The preference weights $w_{i,j}$ are drawn independently from a log-normal distribution of scale σ, where σ is varied in the set $\{1, 10, 20, 40\}$. Each customer segment occurs with probability $\mu_1 = \cdots = \mu_5 = 1/5$. Here, σ intuitively controls the amount of heterogeneity in choice behavior across segments; we will momentarily see that predictive performance is sensitive to this parameter.
- *Consider-then-choose model:* For our second set of synthetic data, the purchase probabilities arise from the intervals model introduced in Section 4.1. Indexing the set of all possible intervals by $k \in \{1, 2, \ldots, K\}$, the probability vector $(\lambda_1, \ldots, \lambda_K)$ is drawn uniformly at random from the unit simplex.

Given the above generative settings, our datasets takes the form of a random matrix $(p_{ij})_{i\in[N], j\in[100]}$, where the entry p_{ij} is the empirical probability of purchases of product i within the assortment $\mathcal{A}_j$ according to the ground truth choice model. As discussed earlier, we fit to these datasets a suite of choice models, including MMNL models with up to 3 customer segments, as well as the quasi-convex model. To calibrate these models with the data, we use known estimation methods in the literature (McFadden 1973, Talluri and Van Ryzin 2006, van Ryzin and Vulcano 2014, Bertsimas and Mišic 2015). Specifically, we make use of a column generation algorithm to

Table 3 **Prediction errors of the models estimated from the synthetic data, in different generative settings.**

Ground truth		Quasi-convex		MNL		MMNL(2)		MMNL(3)	
		MSE	MAPE	MSE	MAPE	MSE	MAPE	MSE	MAPE
MMNL(5)	$\sigma = 1$	0.059	0.204	0.005	0.050	**0.002**	0.048	0.004	**0.048**
	$\sigma = 10$	**0.169**	**0.463**	0.207	0.529	0.204	0.510	0.197	0.511
	$\sigma = 20$	**0.197**	**0.489**	0.224	0.520	0.217	0.518	0.216	0.514
	$\sigma = 40$	**0.199**	**0.472**	0.245	0.533	0.240	0.524	0.226	0.519
Intervals		**0.003**	**0.010**	0.286	0.509	0.229	0.442	0.225	0.437

Recall that σ is the scale of the log-normal generator used to draw the preference weights in the MMNL instance. Each entry is obtained by sampling 10 random instances. After learning the quasi-convex lists through column generation, the final model is calibrated using the ℓ_1 or the ℓ_2-norm, based on cross-validation.

generate the quasi-convex preference lists and use standard maximum likelihood estimators to learn the MMNL family; the description of the estimation algorithms is deferred to Appendix H.

Prediction task and error metrics. For each dataset thus formed, we carried out a 5-fold cross-validation to estimate the prediction accuracy of the different models out-of-sample. We report two prediction error metrics: the mean square error (MSE), expressed in normalized form as a percentage of the total variance of the data and the mean absolute percentage error (MAPE). Specifically, letting $\mathcal{OS} \subseteq [100]$ designate the collection of out-of-sample assortments and $(\hat{p}_{ij})_{i\in[N] j\in\mathcal{OS}}$ be the prediction matrix, we have:

$$\text{MSE} = \frac{\sum_{j\in\mathcal{OS}} \sum_{i\in\mathcal{A}_j} (\hat{p}_{ij} - p_{ij})^2}{\sum_{j\in\mathcal{OS}} \sum_{i\in\mathcal{A}_j} p_{ij}^2} \ ,$$

and

$$\text{MAPE} = \sum_{j\in\mathcal{OS}} \frac{1}{|\mathcal{OS}|\cdot|\mathcal{A}_j|} \cdot \sum_{i\in\mathcal{A}_j} \frac{|\hat{p}_{ij} - p_{ij}|}{0.01 + p_{ij}} \ .$$

Results. The numerical results, show that the quasi-convex family has relatively accurate predictions in *all* generative settings, and outperforms the parametric models in the plurality of cases. As one might expect, when the intervals model is posited as ground truth, the estimated quasi-convex instances provide low prediction errors out-of-sample. On the other hand, the out-of-sample errors incurred under any of the MMNL models is substantially larger.

More interestingly, when data is generated according to an MMNL model, we see that as the scale parameter that controls customer heterogeneity across segments, σ, grows (i.e., $\sigma \geq 10$), the quasi-convex model provides more accurate out of sample predictions than the estimated MNL, MMNL(2) and MMNL(3) instances. For example, for $\sigma = 10$, the MSE is smaller by a factor of between 14 and 22%, while the MAPE is smaller by a factor of 10% to 12%. The cases where the MMNL models provide an improvement in prediction error are when $\sigma \leq 5$. In such cases the heterogeneity in choice across distinct segments is so small that an MNL model would be expected to provide a good fit to the data – a fact that is borne out in the experiment. In this setting the *absolute* prediction errors are relatively small across all models, including the quasi-convex model.

Table 4 **% Improvements in the predictive accuracy of our quasi-convex model against the chosen benchmarks.**

(% Improvement)		MNL		MMNL(2)		MMNL(3)	
Datasets		MSE	MAPE	MSE	MAPE	MSE	MAPE
Dog Food & Dog Treats	$k=2$	12%	28%	7.5%	14%	6.8%	14%
	$k=5$	21%	23%	13%	15%	15%	12%
Bath tissue	$k=2$	79%	59%	71%	50%	65%	42%
	$k=5$	67%	35%	58%	31%	51%	26%
Shampoo & Conditioners	$k=2$	15%	22%	9.3%	13%	11%	13%
	$k=5$	12%	18%	8.2%	14%	4.0%	10%

Note: k is the cross-validation parameter. Each entry is computed by averaging over 10 cross-validation estimates.

7.2.2. Purchase panel data. Our industry partner tracks the daily transactions made by hundreds of thousands of consumers across several product categories and retailers. In order to form our input datasets, we had access to three product categories with frequent purchases, namely *Bath tissue*, *Shampoo and Conditioners*, and *Dog Food and Treats*. In each category, the time horizon considered varies from 2 to 5 months to obtain around 1 million transactions. Transactions are aggregated at the brand level. Each assortment corresponds to the combination of a retail chain and a US state [3]: it is defined as the collection of products with at least one transaction. Having specified the assortments $\mathcal{A}_1, \ldots, \mathcal{A}_s$, the purchase probabilities p_{ij} are obtained by computing the market shares of products according to the observed transactions. For any given state and product category, we only consider those retailers with greater than 500 transactions in that state for that category. Hence, the *Bath tissue*, *Dog Food and Treats*, and *Shampoo and Conditioners* datasets are respectively formed by 58, 211 and 257 assortments. To verify the robustness of our conclusions, multiple experiments were conducted by randomly sub-sampling 50 products in each category.

Results. Table 4 reports the percentage improvement in predictive accuracy for each metric and product category. For example, denoting by MSE_Q the chi-square errors on the predictions of the quasi-convex model and MSE_{MNL} those associated with the MNL model, the percentage improvement relative to the MNL model is given by

$$\frac{\text{MSE}_{MNL} - \text{MSE}_Q}{\text{MSE}_{MNL}} .$$

The table reports these quantities averaged across multiple experiments (each experiment obtained via a random sampling of products), and each cross-validation fold. Our quasi-convex modeling approach outperforms the parametric benchmark models by providing smaller out-of-sample prediction errors in *all* cases. The gain is smallest (as expected) for the MMNL(3) model, but remains substantial in absolute terms – as large as 60% for certain product categories. Our results are

[3] Retailers in the dataset carry near identical assortments in a given state over the time periods considered.

robust to using other metrics (not reported here) such as chi-square. In summary, we conclude that the quasi-convex consider-then-choose model provides strong predictive power on real-world choice data.

8. Concluding Remarks

General objective function. A close examination of the algorithm reveals that our results apply in fact to a more general class of objective functions that we describe below. We introduce the *pay-off* function $f: [N] \times [K] \to \mathbb{R}$, where $f(i,j)$ is the contribution to the objective due to the purchase of product i by the preference list j. Letting $i(\mathcal{A}, j)$ denote the product purchased by preference list j when faced with assortment $\mathcal{A}$, the objective value for the assortment $\mathcal{A}$ is given by the expected pay-off: $\sum_{j\in[K]} \lambda_j \cdot f(i(\mathcal{A}, j)$. We may mention two application of practical interest captured by this more general family of objective functions. It accounts for potentially heterogeneous per-selling price over customer-types, e.g., targeted promotions or loyalty programs. Another interesting problem formulation consists in maximizing the customers' utility. In this case $f(i,j)$ is interpreted as the utility garnered by customer $j \in [k]$ when purchasing product $i \in [N]$.

Future work. This work opens exciting perspectives for future research. A natural lead is to further investigate the interplay between the behavioral heuristics identified by the marketing literature and the running time complexity of our dynamic program. In addition, the implementation allows for several refinements, such as using heuristics to prune the computational tree or exploring the subproblems in parallel. Another important question is to investigate the identifiability of the models discussed in this work from data, in particular the quasi-convex model, and study the computational and sample complexity associated with the estimation problem.

Appendix A: Modeling the Consideration Sets

Proof of Lemma 2 To ease the exposition, we define i as the minimal product in S, and let v designate a customer-type in T. By definition, there exists a customer-type $u \in T$ such that $i \in C_u$. Also, since $G[S,T]$ is a connected subgraph,give there exists a path between v and u. We now define $v^* \in T$ as the customer-type in T which satisfies $i \in C_{v^*}$ and has the shortest path with v. In other terms, v^* minimizes the length of a path between v and x over all $x \in T$ such that $i \in C_x$. This set is not empty because it contains customer-type u. We are going to prove that $C_v \cap [i] \subset C_{v^*} \cap [i]$.

Let j_1, j_2,... j_l be sequence of customer-type nodes in $G[S,T]$ corresponding to the shortest path between v and v^*:

$$\begin{cases} j_1 = v^* \text{ and } j_l = v \\ \forall r \in [l-1],\ \exists a \in S \ \text{ s.t. } (j_r, a, j_{r+1}) \text{ is a path of } G[S,T] \end{cases}$$

Let a_1, a_2...,a_{l-1} be the corresponding sequence of maximal intersections of the consideration set of each two subsequent customer-types along this path:

$$\forall 2 \le r \le l,\ a_r = \max[C_{j_r} \cap C_{j_{r-1}}]$$

By convention, we set $a_1 := i$. We now prove by induction over r, $2 \le r \le l$, that $a_r > a_{r-1}$ and $C_{j_r} \cap [a_r] \subset C_{j_{r-1}} \cap [a_r]$.

- *Base case ($r = 2$).* We first note that $a_1 < a_2$. Indeed, since i is the minimal element of S, we can infer that $a_2 \ge i$. These indices can not be equal otherwise $i \in C_{j_2}$ and we would obtain a strictly shorter path between v and j_2 by considering the path $(j_2, a_3, \dots, a_l, j_l)$ and this contradicts the minimality of l. We now prove the inclusion. We infer from the definition of weakly laminar consideration sets that either $C_{j_1} \cap [a_2] \subset C_{j_2} \cap [a_2]$ or $C_{j_2} \cap [a_2] \subset C_{j_1} \cap [a_2]$. In addition, item i is contained in C_{j_1} and $i \notin C_{j_2}$, otherwise it would contradict the minimality of the path. Since $i \notin C_{j_2}$, we can infer that $C_{j_2} \cap [a_2] \subset C_{j_1} \cap [a_2]$, which leads to the desired result.

- Inductive step $r > 2$. We begin by assuming that $a_r > a_{r-1}$. Again, by definition, either $C_{j_r} \cap [a_r] \subset C_{j_{r-1}} \cap [a_r]$ or $C_{j_{r-1}} \cap [a_r] \subset C_{j_r} \cap [a_r]$. We assume that the latter is satisfied to prove a contradiction. Since we assume that $a_r > a_{r-1}$, the latter set inclusion leads to $a_{r-1} \in C_{j_r}$. Therefore, $C_{j_{r-2}}$ and C_{j_r} both contain product a_{r-1} and (j_{r-2}, a_{r-1}, j_r) is a path of $G[S,T]$. Thus, we can obtain a path between v^* and v of strictly smaller length using the shortcut (j_{r-2}, a_{r-1}, j_r) instead of $(j_{r-2}, a_{r-1}, j_{r-1}, a_r, j_r)$. However, this would contradict the minimality of l. Thus: $C_{j_r} \cap [a_r] \subset C_{j_{r-1}} \cap [a_r]$.

In order to prove the above assumption that $a_r > a_{r-1}$, we now assume that $a_r \le a_{r-1}$ and prove that it leads to a contradiction. By the induction hypothesis, we know that $C_{j_{r-1}} \cap [a_{r-1}] \subset C_{j_{r-2}} \cap [a_{r-1}]$. Thus, if $a_r \le a_{r-1}$, it follows that $a_r \in C_{j_{r-1}} \cap [a_{r-1}]$. From the above inclusion, we obtain that $a_r \in C_{j_{r-2}}$. Therefore, there is an edge between j_{r-2} and a_r and $(j_1, a_2, j_2, \dots, j_{r-2}, a_r, j_r, \dots, j_l)$ would form a path between v^* and v of strictly smaller length, which contradicts the minimality of l. We can thus obtain that $a_r > a_{r-1}$.

So far, for any given $v \in T$, we have proven the existence of $v^* \in T$ such that $C_v \cap [i] \subset C_{v^*} \cap [i]$ and $i \in C_{v^*}$. Defining $T(i)$ as the subset of customer-types in T that consider product i, we may verify that the collection of subsets $C_j \cap [i]$ where $j \in T(i)$ is nested. As a consequence, this collection admits a maximal element, that corresponds to a customer-type $j^* \in T$. Thus, we conclude that $C_v \subseteq C_{j^*}$ for any $v \in T$.

Proof of Theorem 5 The proof is analogous to the previously considered models. We seek to upper-bound the quantity $|\Phi\langle \mathcal{S} \rangle|$. To this end, we let (S,T) be a subproblem of $\mathcal{S}$. We have

$$\begin{aligned}
\Phi(S,T) &= [\min(S)] \bigcap \left(\bigcup_{j \in T} C_j \right) \\
&= [\min(S)] \bigcap \left(\bigcup_{j \in T} \bigcup_{e \in [d]} \{ i \in [N] : x_e^{(j)} \ge t_e^{(j)} \} \right) \\
&= [\min(S)] \bigcap \left(\bigcup_{e \in [d]} \bigcup_{j \in T} \{ i \in [N] : x_e^{(i)} \ge t_e^{(j)} \} \right) \\
&= [\min(S)] \bigcap \left(\bigcup_{e \in [d]} \left\{ i \in [N] : x_e^{(i)} \ge \min_{j \in T} t_e^{(j)} \right\} \right) ,
\end{aligned}$$

where the second equality follows from Definition 4, and the third equality proceeds by changing the union order. We conclude by observing that for each $e \in [d]$, the quantity $\min_{j \in T} t_e^{(j)}$ can take at most K distinct values. Therefore, we obtain that $|\Phi\langle \mathcal{S} \rangle| \le N \cdot K^d$. □

Appendix B: Marginalization

We give here the specifics of the marginalization algorithm described in Section 5.

Informal sketch. By constructing and updating an appropriate data-structure, denoted by $\mathcal{D}(S,T,\boldsymbol{L}) \sim \mathcal{D}$, we prevent the redundant exploration of the children subproblems appearing in equation (4). Specifically, we construct recursively a directed graph $\mathcal{D}$, illustrated by Figure (5). To this end, each node inserted in $\mathcal{D}$ is labelled by a combination of a child subproblem, and the index of the last customer-type in $T(i)$ that has been processed, termed the *layer* of the node. At each step, we consider all unmarked nodes, and process their next customer-type in $T(i)$ according to the increasing index order. The dynamic program decides whether the current customer-type is allocated to product i or not. Each decision entails a graph decomposition into children subproblems according to Lemma 3. The corresponding nodes, with the respective customer-type layer, are inserted in $\mathcal{D}$ as unmarked nodes. Also, we add directed edges connecting the father node to its respective children nodes. The procedure terminates when it attains the maximal layer index.

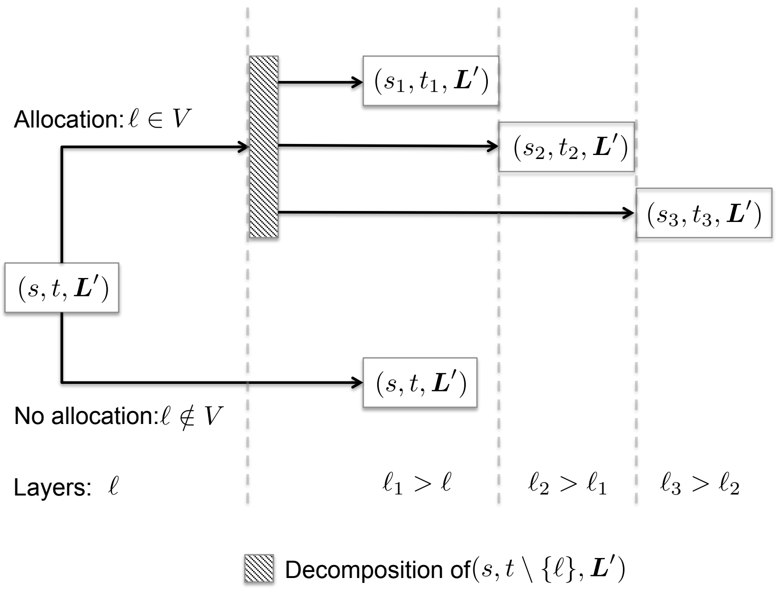

$(s_1,t_1,\boldsymbol{L}')$
Allocation: $\ell \in V$
$(s_2,t_2,\boldsymbol{L}')$
$(s_3,t_3,\boldsymbol{L}')$
$(s,t,\boldsymbol{L}')$
$(s,t,\boldsymbol{L}')$
No allocation: $\ell \notin V$
Layers: ℓ
$\ell_1 > \ell$
$\ell_2 > \ell_1$
$\ell_3 > \ell_2$
▨ Decomposition of $(s, t \setminus \{\ell\}, \boldsymbol{L}')$

Figure 5 **Recursive step of the procedure that constructs $\mathcal{D}(S,T,\boldsymbol{L})$.**

Generation of the computational tree. More formally, we assume that the customer-types $T(i)$ are reindexed in an arbitrary order $T(i) \sim [l]$ where $l = |T(i)|$. We introduce a directed graph data-structure $\mathcal{D}(S,T,\boldsymbol{L})$, initially set empty. (In the following, unless ambiguity arises, it is simply denoted $\mathcal{D}$ for ease of exposition.) Each node we add to $\mathcal{D}$ is uniquely labelled by a tuple $(j,s,t) \in [l] \times \mathcal{P}(S) \times \mathcal{P}(T)$ where $(s,t,\boldsymbol{L}')$ is a child subproblem appearing in equation (4) and generated by an allocation contained in $[j]$. The nodes are generated by an iterative procedure described below:

- *Base case.* We start with an empty graph $\mathcal{D} \leftarrow \emptyset$. The first nodes that we add correspond to the empty allocation $V = \emptyset$. Namely, for each connected components $G_{\boldsymbol{L}'}[S_u^{(\emptyset)}, T_u^{(\emptyset)}]$, we insert a node in $\mathcal{D}$ labelled $(S_u^{(\emptyset)} T_u^{(\emptyset)}, 0)$. We refer to them as the roots of $\mathcal{D}$.

- *Recursive step.* Assume that a node with label (s,t,j) has been added to $\mathcal{D}$. The next customer-type we consider, denoted j', is the minimum of $t \cap [j+1,l]$. The decision made at this stage is whether customer-type j' gets allocated to product i or not. In the latter case, a node (s,t,j') is inserted in $\mathcal{D}$ unless it already belongs to the data-structure. Also, we create a directed edge between the parent node labelled (s,t,j) and its descendant (s,t,j'). Conversely, in case j' is allocated to product i, we derive the residual graph $G_{\boldsymbol{L}'}[s,t \setminus \{j'\}]$ and compute its connected components. Each connected component $G_{\boldsymbol{L}'}[s_u,t_u]$ leads to the insertion of a new node (s_u,t_u,j') unless it already belongs to $\mathcal{D}$. Also, directed edges are added between the parent node and its descendants in $\mathcal{D}$.

The graph $\mathcal{D}$ built via this recursive procedure is a directed forest – a cycle-free directed graph. Indeed, the only edges are between father nodes and their offspring. Because the customer-type index in the node label is monotonic ($j' > j$), there cannot be any cycle. Finally, we observe that the leafs of $\mathcal{D}$ uniquely represent all subproblems generated by the allocations $V \subseteq T(i)$. Indeed, any V corresponds to a sequence of binary decisions in $[l]$. This sequence of decisions defines a collection of paths in $\mathcal{D}$ starting from the root nodes. By construction, the subproblems described by the labels of the terminating leafs are exactly the subproblems generated by V.

In terms of running time, each distinct subproblem shows up in at most l nodes of $\mathcal{D}$ (and l is smaller than K). Therefore, the total running time to generate the DP computational tree is upper bounded by $O(NK^2 \cdot |\mathcal{S}|)$.

Solving equation (4)**.** Once the DP computational tree has been drawn, the subproblems are solved backwards using the recursive formula (4). By exploiting the data-structure $\mathcal{D}(S,T,\boldsymbol{L})$, we show in this paragraph that the maximization problem (4) can be recast as a low dimensional dynamic program that can be solved efficiently. That is, at each recursive step of the master dynamic program, we solve a separate dynamic program, termed the *marginalized dynamic program.*

We consider a fixed instance $(S,T,\boldsymbol{L})$. Suppose that all subsequent subproblems have been solved as we move backwards over the computational tree. For ease of exposition, the reference to the parameters $(S,T,\boldsymbol{L})$ is omitted when introducing the marginalized dynamic program, and the notations $i, T(i), \boldsymbol{L}', l$ and $\mathcal{D}$ are consistent with the previous definitions.

By construction, we note that for each node of $\mathcal{D}$, with label $n=(s,t,j)$, the corresponding subproblem $(s,t,\boldsymbol{L}')$ has been generated by at least one allocation $V \subseteq [j]$, that we designate as $V(n)$. We define the value function $F(n)$ as the optimal expected revenue from customer-types t in the subproblem $(S,T,\boldsymbol{L})$ under the constraints that *(i)* product i is stocked and *(ii)* the allocation of this product $V \subseteq T(i)$ satisfies the constraint $V \cap [j] = V(n)$, i.e., the projection of V on $[j]$ is $V(n)$. Let j' be the next customer-type for which a decision is made when examining node n, i.e., $j' = \min([j+1,l] \cap T)$. Letting $\mathcal{N}(n)$ denote the children nodes of n if j' is allocated to product i and n' be the child node of n otherwise, we obtain:

$$F(n) = \max\left(F\left(n'\right) \ , \ \lambda_{j'} \cdot P_i + \sum_{u \in \mathcal{N}(n)} F\left(u\right) \right)$$

Indeed, if customer-type j' is allocated to product i, it generate a revenue of $\lambda_{j'} \cdot P_i$ and the residual graph decomposes into the connected subgraphs described by $\mathcal{N}(n)$. Conversely, if j' is not allocated to product i, the connected subgraph is not modified further and the revenue is that of $F(n')$. This is consistent with the constraint *(ii)* as $V(n')=V(n)$ when j' is not added to the allocation.

By applying this formula inductively from the leafs of $\mathcal{D}$, we compute $F(n_1),...,F(n_{r(\emptyset)})$ where $n_1,...,n_{r(\emptyset)}$ are the root nodes of $\mathcal{D}$. Conditional on the fact that i is stocked, we conclude that:

$$J(S,T,\boldsymbol{L}) = \sum_{u=1}^{r(\emptyset)} F(n_u)$$

Therefore, equation (4) is equivalent to:

$$J(S,T,\boldsymbol{L}) \;=\; \max\left(\sum_{u=1}^{r(-)} J(S_u^-,T_u^-,\boldsymbol{L}) \;,\; \max\left[\sum_{u=1}^{r(\emptyset)} F(n_u)\right]\right)$$

Example with the in-tree model. To flesh out our marginalized algorithm through a concrete model, we argue now that this approach allows to solve efficiently the *in-tree* model proposed by Honhon et al. (2012). Here, each product is represented by a node in a rooted tree $\mathcal{T}$. Each consideration set in $\mathcal{C}$ corresponds to a path from the root to a given node – we will denote by C_v the consideration set formed by the path from the root to node $v \in \mathcal{T}$. We further assume that such directed paths define the increasing preference order, namely, the farther from the root, the more preferred is a product, thus leading to some (non-unique) ranking function σ. The processing order is chosen as the reverse permutation $\bar{\sigma}$, that is, products are processed from the root to the descendant nodes. To argue that the marginalization is efficient, it is sufficient to show that, for any product $i \in [n]$, we can restrict attention to allocations $V \subseteq T(i)$ corresponding to subtrees of product nodes (here, by abuse of language, we mix each customer-type with his corresponding consideration set and product node in $\mathcal{T}$). To arrive at a contradiction, suppose we have an allocation $V \subseteq T(i)$ with $C_i, C_j \in V$, where j is a descendent of i, and k is on the path from i to j although $C_k \notin V$. Since product i has been allocated to customer-type associated with C_j, who prefers product k over i according to σ, it follows that product k is not contained in $S^{(V)}$. Consequently, all product nodes between j and k have been eliminated from the residual graph, and therefore the customer-type node of C_k is disconnected from any (non-trivial) connected component. Thus, we can assume without loss of generality that $C_k \in V$.

Complexity Analysis. We now derive a general upper bound on the running time.

Proof of Proposition 2 The proof of the first claim follows from our previous observations. At each node $(S,T,\boldsymbol{L})$ of the computational tree, the running time for generating the graph $\mathcal{D}(S,T,\boldsymbol{L})$ along with the running time for solving the marginalized DP is at most $O(N \cdot K^2 \cdot |\mathcal{S}|)$. Summing over all the nodes of the DP computational tree, we obtain a total running time of $O(N \cdot K^2 \cdot |\mathcal{S}|^2)$.

We now derive an upper-bound on the state space size. To this end, we construct a function Φ that maps any subproblem generated along the recursion to a subset of products as well as a product within this set. By definition, any subproblem $(S,T,\boldsymbol{L}) \in \mathcal{S}$ has been generated by a sequence of decisions whereby some products in $\{1,\dots,\min(S)-1\}$ have been stocked. We define $\mathcal{A}(S,T,\boldsymbol{L})$ as a partial assortment of products corresponding to a sequence of decisions prior to generating subproblem $(S,T,\boldsymbol{L})$. The mapping is described as follows:

$$\begin{aligned}\Phi : \mathcal{S} &\to \mathcal{P}([N]) \times [N]\\ (S,T,\boldsymbol{L}) &\mapsto (\mathcal{A}(S,T,\boldsymbol{L}) \cup S, \min(S))\end{aligned}$$

It is sufficient to show that this function is injective to obtain the desired result. Assume that two subproblems satisfy $\Phi(\mathcal{I}_1) = \Phi(\mathcal{I}_2)$ where $\mathcal{I}_1 = (S_1, T_1, \boldsymbol{L}_1)$ and $\mathcal{I}_2 = (S_2, T_2, \boldsymbol{L}')$. Then, by definition:

$$\begin{aligned} S_1 &= \Phi^{(1)}(\mathcal{I}_1) \cap [\Phi^{(2)}(\mathcal{I}_1), N] \\ &= \Phi^{(1)}(\mathcal{I}_2) \cap [\Phi^{(2)}(\mathcal{I}_2), N] \\ &= S_2\ . \end{aligned}$$

This proves that the two subproblems have the same subsets of products. By similar observations, we can claim that both $\mathcal{I}_1$ and $\mathcal{I}_2$ are generated by the same sequence of decisions, or equivalently the same assortment $\mathcal{A} \subseteq \Phi^{(1)}(\mathcal{I}_1) \cap [\Phi^{(2)}(\mathcal{I}_1) - 1]$. We also know that $\boldsymbol{L} = \boldsymbol{L}'$ because the truncation vector is determined by the previous stocking decisions. As a result, the only difference between the two subproblems could only be caused by a different sequence of allocations. Therefore, it is sufficient to prove that the set of customer-types remaining in the two connected subgraphs are exactly the same in order to obtain that $\mathcal{I}_1 = \mathcal{I}_2$. Ad absurdum, assume $j \in T_1 \setminus T_2$. Because $S_1 = S_2$, this means that customer-type j is still unsatisfied in $\mathcal{I}_1$ whereas it was allocated to a product in the sequence of decisions that generates the subproblem $\mathcal{I}_2$. Since j has been satisfied along the generation of the subproblem $\mathcal{I}_2$, there exists a product i in $\mathcal{A}$ that belongs to C_j. In addition, since $G_{\boldsymbol{L}}[S_1, T_1]$ is a connected subgraph, this means there exists an edge between node j and a product node $i' \in S_1$. Along the generation of subproblem $\mathcal{I}_1$, i has been made available to j but it was not allocated to customer-type j - as a result its consideration set has been truncated to only account for products more preferred than i: $L_j < \sigma_j(i)$. Thus customer j necessarily prefers i' over product i. On the other hand, as product i was allocated to customer-type j when generating $\mathcal{I}_2$, the product i' has been deleted because he prefers it over i. Thus $i' \notin S_2$ and since $S_1 = S_2$, we obtain a contradiction: $i' \notin S_1$.

Appendix C: Proof of Theorem 6

We construct a function Ψ that maps any subproblem generated along the recursion to a tuple that lies in a space of size $2^{2d-2}h$. By showing that Ψ is injective, we obtain the upper bound on the size of the state space.

Specifically, assuming that $(S, T, \boldsymbol{L}) \in \mathcal{S}$, we define i as the next product to be processed in S and $\mathcal{A}$ corresponds to the assortment decisions in $[i-1]$ which generate this subproblem. The image of $(S, T, \boldsymbol{L})$ by Ψ is defined as the tuple $(S_0, T_0, \boldsymbol{x}, \boldsymbol{y})$ where:

- (S_0, T_0) is the subproblem of the unique-ranking dynamic program generated by the sequence of stocking decisions $\mathcal{A} \cap [i-2d]$ over $[i-1]$ and such that $T \cap T_0 \neq \emptyset$,
- $\boldsymbol{x} = \mathcal{A} \cap [i-2d+1, i-1]$.
- and $\boldsymbol{y} = S \cap [i+1, i+2d-1]$.

To prove that this mapping is injective, we show that each subproblem $(S, T, \boldsymbol{L})$ is uniquely determined by the tuple $(S_0, T_0, \boldsymbol{x}, \boldsymbol{y})$.

We begin by remarking that all preference lists j in T do not intersect with $\mathcal{A} \cap [i-2d]$. Otherwise, we define α as an arbitrary product of the intersection of $\mathcal{A} \cap [i-2d]$ with C_j. Then, by construction, $L_j \leq \sigma_j(\alpha)$. In addition, given that $\sigma_j \in B(\sigma, d)$, any product in $[i-2d]$ is preferred over products in S (recall that the products are numbered according to the central permutation σ, meaning that σ is the identity). As a result, $\sigma_j(\alpha) < \sigma_j(\beta)$ for all product $\beta \in S$, meaning that $C_j(L_j)$ does not intersect with S which contradicts the connectivity of the subgraph $G_{\boldsymbol{L}}[S,T]$.

Uniqueness of $\boldsymbol{L}$. We now argue that $\boldsymbol{L}$ is uniquely determined by $\boldsymbol{x}$. Indeed, if $j \in T$, then by the above remark, C_j does not intersect with the projection of $\mathcal{A}$ on $[i-2d]$ and as result L_j is a deterministic function of $\boldsymbol{x}$:

$$L_j = \min\{\sigma_j(\alpha) : \alpha \in \boldsymbol{x}\}$$

Uniqueness of T. We first show that $T \subseteq T_0$. Assume that $j \in T$. Using the above remark again, we infer that j is not satisfied and eliminated by the decisions of stocking $\mathcal{A} \cap [i-2d]$ in the unique-ranking DP. As a result, T is included in the set of customer-type nodes of the residual graph obtained in the unique-ranking DP after performing the graph operations associated with the sequence of stocking decisions $\mathcal{A} \cap [i-2d]$. Thus, it is sufficient to verify that T lies in a connected component of the residual graph in order to prove that $T \subseteq T_0$. The key observation is that the residual graph obtained by the stocking decisions of $\mathcal{A}$ under the general DP is a subgraph of the residual graph generated by the decisions of stocking $\mathcal{A} \cap [i-2d]$ in the unique-ranking DP. Indeed, the graph operations performed by the unique-ranking algorithm are also performed at some step of the recursion of the general DP:

- Customer-type node deletions: by the above remark, any preference list that is discarded as a result of the stocking decisions $\mathcal{A} \cap [i-2d]$ in the unique-ranking DP is also discarded at some point of the decision sequence associated with $\mathcal{A}$ in the general DP.
- Product node deletions: in the unique-ranking case, a product node is deleted when it is processed. Given that the two algorithms follow the same processing order, any product deleted in the unique-ranking case has also been deleted in the general DP.

Therefore, because $(S,T,\boldsymbol{L})$ is connected in the residual graph of the general DP, it is also connected in the residual graph of the unique-ranking DP. Thus, $T \subseteq T_0$ and what remains to be proven is the uniqueness of $T_0 \setminus T$ conditional to $\Psi(S,T,\boldsymbol{L})$. In fact, it is immediate that $T_0 \setminus T$ corresponds to the subset of preference lists nodes that get deleted (or disconnected) due to the allocation of products $\boldsymbol{x} = \mathcal{A} \cap [i-2d+1, i-1]$. Let j be a preference list of T_0 that satisfies $\boldsymbol{x} \cap C_j \neq \emptyset$ while customer-type j still belongs to T, meaning that j has not been satisfied with any product of $\boldsymbol{x}$. Let $\alpha \in \boldsymbol{x}$ be the most preferred product of customer-type j in $\boldsymbol{x}$. Then, there necessarily exists at least one product in $S \cap C_j$ preferred over product α otherwise $C_j(L_j) \cap S = \emptyset$ and the subgraph would not be connected. In fact, because the preference rankings σ_j are contained in $B(\sigma, d)$, this product is at distance at most $2d$ from α, meaning that it belongs to $\boldsymbol{y}$. Reciprocally, if there exists a product in $\boldsymbol{y} \cap C_j$ preferred over α, then it follows that j has not been allocated to any product of $\boldsymbol{x}$. In particular, since $C_j(L_j) \cap S \neq \emptyset$, the customer-type node j is connected with S and thus $j \in T$. This shows that $T_0 \setminus T$ is uniquely determined as a function of the subsets $\boldsymbol{x}$ and $\boldsymbol{y}$, which proves the desired result.

Uniqueness of S. In general, we have:

$$S = \bigcup_{u \in T} C_j(L_j) \ .$$

It immediately follows that the uniqueness of S can be inferred from the uniqueness of T and $\boldsymbol{L}$. $\square$

Appendix D: Quasi-convex preference lists

Claim 2. *For a fixed central permutation, there exists $2^{N+1}-N-2$ quasi-convex preference lists.*

Proof Let $\Sigma(N)$ be the set of quasi-convex preference lists over N products. The preference lists are uniquely defined by their consideration set and the quasi-convex ranking function. For any fixed interval of length $\ell \in [N]$, the ranking function can be *viewed* as a permutation over ℓ elements: $[\ell] \to [\ell]$. We now construct a mapping ϕ from any subset $S \subset [2,\ell]$ to a quasi-convex permutation over the interval $[\ell]$. $\phi(S)$ is defined as follows:

$$\begin{cases} \phi(S) \text{ is decreasing over } [|S|] & \text{with } \phi(S)\langle|[|S|]\rangle = S \\ \phi(S) \text{ is increasing over } [|S|+1,N] & \text{with } \Phi(S)\langle[|S|+1,N]\rangle = [\ell]\setminus S \end{cases}$$

Indeed, the quasi-convex permutation $\phi(S)$ is uniquely defined given its monotonicity on each interval. It can be verified that this mapping is surjective (by taking S equal to the set of image values of the quasi-convex permutation on its decreasing interval excluding the minimal value 1). Finally, it is injective by observing that if $\phi(S_1)$ and $\phi(S_2)$ are equal, in particular they share the same decreasing segments and $S_1 = S_2$. Therefore, the cardinality of quasi-convex ranking functions over an interval of length ℓ is $2^{\ell-1}$. By remarking that there exists $N-\ell+1$ distinct intervals of length ℓ, we obtain:

$$\begin{aligned} |\Sigma(N)| &= \sum_{\ell=1}^{N}(N-\ell+1)\cdot 2^{\ell-1} \\ &= (N+1)\cdot\sum_{\ell=0}^{N-1}2^{\ell} - \sum_{\ell=1}^{N}\ell\cdot 2^{\ell-1} \\ &= (N+1)\cdot\left(2^N-1\right)-(N-1)\cdot 2^N - 1 \\ &= 2^{N+1}-N-2\ . \quad \square \end{aligned}$$

Appendix E: Proof of Theorem 8

The processing order can be chosen as an arbitrary permutation - but to fix ideas, we process the products in the increasing index order.

We incorporate to the model a 'dummy' preference list denoted by index 0 associated with the null utility vector $\boldsymbol{u}_0 = \vec{0}$ as well as a 'dummy' product corresponding to $\boldsymbol{x}_0 = \vec{0}$. Also, note that in what follows, by abuse of language, a product may refer to the corresponding graph node or its representation in the feature space.

Inductive hypothesis. We prove that each DP subproblem is characterized by a polytope in the feature space defined by a constant number of facets, chosen among a polynomial set of affine constraints. Specifically, we prove the following property inductively. Suppose that $(S,T,\boldsymbol{L})$ is generated along the recursion. Then, $G_{\boldsymbol{L}}[S,T]$ is the connected component of $G_{\boldsymbol{L}}[S',T']$ that contains product node $i=\min(S)$, where:

- There exists $(a,b,c,d)\in[N]^2\times[K]^2$ such that S' is defined as follows:

$$\begin{cases} \boldsymbol{z} = Rot\left(\frac{\pi}{2},\boldsymbol{x}_b-\boldsymbol{x}_a\right)\ , \\ H(a,b,c,d) = \left\{\boldsymbol{x}\in\mathbb{R}^2 : \boldsymbol{x}\cdot\boldsymbol{u}_c \le \boldsymbol{u}_c\cdot\boldsymbol{x}_a,\ \boldsymbol{x}\cdot\boldsymbol{u}_d \le \boldsymbol{u}_d\cdot\boldsymbol{x}_b,\ \boldsymbol{x}\cdot\boldsymbol{z} \ge \boldsymbol{x}_a\cdot\boldsymbol{z}\right\}\ , \\ S' = \{j\in[i,N] : x_j \in H(a,b,c,d)\setminus\partial H(a,b,c,d)\}\ . \end{cases}$$

- The set T' is formed by all customer-types whose utility vector belongs to the cone with extreme rays $(\boldsymbol{u}_c, \boldsymbol{u}_d)$ (where the rays are ordered in the anti-trigonometric order):

$$T' = \left\{ j \in [K] : \exists \lambda_1, \lambda_2 > 0 \text{ s.t. } \boldsymbol{u}^{(j)} = \lambda_1 \boldsymbol{u}_c + \lambda_2 \boldsymbol{u}_d \right\} \ .$$

- The truncation vector $\boldsymbol{L}$ is given by:

$$\forall j \in T', \ L_j = \min(\sigma_j(a), \sigma_j(b)) \ .$$

Before proving this result, observe that this property implies that there exists an injective mapping from the state space $\mathcal{S}$ to the space of 5-tuples, described by a 3-tuple of products and a pair of customer-types. As a result, we conclude that $|\mathcal{S}| = O(K^2 N^3)$.

Base case. If $(S, T, \boldsymbol{L})$ is one of the roots of the DP computational tree, no products has been examined yet and $G_{\boldsymbol{L}}[S, T]$ is a connected component of G. Then, we set $a = b = c = d = 0$ and $i = \min(S)$. The polyhedron $H(a, b, c, d)$ describes to the entire space $\mathbb{R}^2$, $S' = [N]$ and $T' = [K]$. The above property is immediately satisfied.

Recursive step. We assume that $(S, T, \boldsymbol{L})$ satisfies the above properties with respect to the tuple (a, b, c, d) and $(S', T', \boldsymbol{L})$. Let $i = \min(S)$ be the next item to be examined.

If i is not stocked in the assortment, we only need to discard product node i from the graph and compute the connected components of $G_{\boldsymbol{L}}[S \setminus \{i\}, T]$ to obtain the children subproblems. Equivalently, by the induction hypothesis, any child subproblem is a connected component of $G_{\boldsymbol{L}}[S' \setminus \{i\}, T'])$ and the above property is satisfied.

On the other hand, if the product i is allocated to a subset of customer-types $V \subseteq T(i)$. We define α, β as the indices corresponding to the extreme lines of the cone $conv\{\boldsymbol{u}_h : h \in V\}$ - such that $\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta$ are ordered in the anti-trigonometric order. Using previous notation, any subproblem generated by the allocation V is a connected component of $(S^{(V)}, T^{(V)}, \boldsymbol{L}')$. We prove that it satisfies the desired inductive property, either with respect to the parameters (a, i, c, α), or with respect to (i, b, β, d), as illustrated by Figure 6. In what follows, note that, by abuse of language, utility vectors refer indifferently to vectors in the feature space, and to the corresponding customer-types.

We begin by proving that the allocation V necessarily corresponds to the cone of utility vectors $(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$, meaning that $T^{(V)}$ is either contained in the cone $(\boldsymbol{u}_c, \boldsymbol{u}_\alpha)$ or in the cone $(\boldsymbol{u}_\beta, \boldsymbol{u}_d)$. To this end, assume that a preference list $j \in T'$ has its utility vector included in the cone $(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$, we seek to prove that $j \notin T^{(V)}$. This preference list would only pick a product whose scalar product is greater than $\boldsymbol{x}_i \cdot \boldsymbol{u}^{(j)}$. Indeed, either product i lies the consideration set C_j, thus customer-type j would only pick a product preferred over i, or it does not belong to C_j and any product it chooses has its scalar product larger than the customer's cut-off level. Given that customer-types α, β are both satisfied with product i, then, all products which satisfy $\boldsymbol{x} \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha$ or $\boldsymbol{x} \cdot \boldsymbol{u}_\beta \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta$ have been removed. By Farkas lemma, since $\boldsymbol{u}^{(j)}$ is contained in the cone $(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$, then any product x which satisfies $\boldsymbol{x} \cdot \boldsymbol{u}^{(j)} \geq \boldsymbol{x}_i \cdot \boldsymbol{u}^{(j)}$ would satisfy either $\boldsymbol{x} \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha$ or $\boldsymbol{x} \cdot \boldsymbol{u}_\beta \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta$. As a result, customer-type j does not prefer any product of $S^{(V)}$ over i, meaning that customer-type node j is disconnected from $S^{(V)}$. Hence, without loss of generality, we may assume that $j \notin T^{(V)}$.

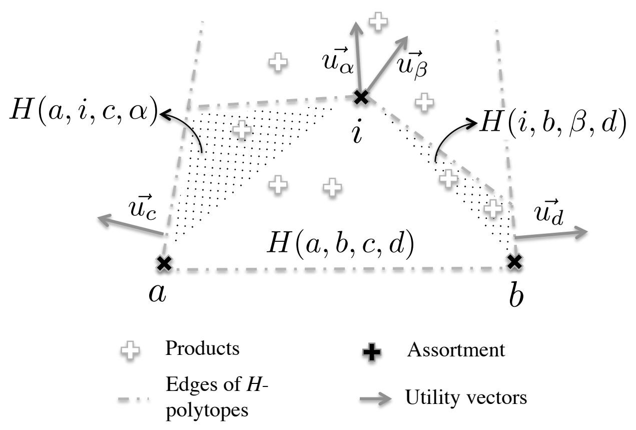


Figure 6 Recursive step: the allocation of product i to the cone $(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$ gives rise to independent subproblems, either contained in the polyhedra $H(a,i,c,\alpha)$, or $H(i,b,\beta,d)$.

We now prove that the products of $S^{(V)}$ either lie in the polyhedron $H(a,i,c,\alpha)$, or within $H(i,b,\beta,d)$. Since all products that the customer-types α and β prefer over i have been discarded, we already know that $S^{(V)}$ is contained in the set $\bar{H}$:

$$\bar{H} = H(a,b,c,d) \setminus \left(\{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{x} \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha\} \bigcup \{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{x} \cdot \boldsymbol{u}_\beta \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta\} \right)$$

Let $j \in T^{(V)}$ designate a preference list in the residual graph. Since a, b and i are stocked in the assortment, the products contained in the truncated consideration set $C_j(L'_j)$ necessarily have a scalar product with $\boldsymbol{u}^{(j)}$ greater than the following quantities: $\boldsymbol{x}_a \cdot \boldsymbol{u}^{(j)}$, $\boldsymbol{x}_i \cdot \boldsymbol{u}^{(j)}$ and $\boldsymbol{x}_b \cdot \boldsymbol{u}^{(j)}$. Equivalently, they lie in the affine half-space defined by $\boldsymbol{x} \cdot \boldsymbol{u}^{(j)} \geq y$, where:

$$y = \max\left(\{\boldsymbol{x}_a \cdot \boldsymbol{u}^{(j)}, \boldsymbol{x}_b \cdot \boldsymbol{u}^{(j)}, \boldsymbol{x}_i \cdot \boldsymbol{u}^{(j)}\} \right) \ .$$

By construction of the polyhedron $H(a,b,c,d)$, if $\boldsymbol{u}^{(j)} \in (\boldsymbol{u}_c, \boldsymbol{u}_\alpha)$, then customer j's most preferred product among $\{a,i,b\}$ is either a or i. Also, the intersection of $\bar{H}$ with the half-space $\{\boldsymbol{x} : \boldsymbol{x} \cdot \boldsymbol{u}^{(j)} \geq y\}$ is included in $H(a,i,c,\alpha)$ and it has no product in common with $H(i,b,\beta,d)$. Therefore $C_j(L'_j)$ is included in $H(a,i,c,\alpha)$ and it does not contain any product in $H(i,b,\beta,d)$. Conversely, if $\boldsymbol{u}^{(j)} \in (\boldsymbol{u}_\beta, \boldsymbol{u}_d)$, then customer j's most preferred product among $\{a,i,b\}$ is either b or i. In this case, $C_j(L'_j)$ is included in the intersection of the affine half-space $\{\boldsymbol{x} : \boldsymbol{x} \cdot \boldsymbol{u}^{(j)} \geq y\}$ with $\bar{H}$, thus it is contained in $H(i,b,\beta,d)$. Also, it does not contain any product in $H(a,i,c,\alpha)$. Combining the above two observation, since $S^{(V)}$ is equal to the union of $C_j(L'_j)$ over $j \in T^{(V)}$, we infer that any connected component of $(S^{(V)}, T^{(V)}, \boldsymbol{L}')$ has its products either included in $H(i,b,\beta,d)$ or contained in $H(a,i,c,\alpha)$.

To prove the desired inductive property, it remains to show that the allocation information is fully captured by each polyhedron. By symmetry, we may focus on $H(i,b,\beta,d)$. Observe that the customer-type deletions give rise to the cone $(\boldsymbol{u}_\beta, \boldsymbol{u}_d)$, such that each customer-type whose utility vector lies in this cone is not discarded at the moment. Similarly, the constraints of the polyhedra $H(i,b,\beta,d)$ with left-hand side $x \cdot u_\beta$ and $x \cdot u_d$ capture all the product deletions active in the polyhedron $H(i,b,\beta,d)$. Finally, since b and i are the most preferred products among all products stocked prior for any customer-type j in the cone $(\boldsymbol{u}_\beta, \boldsymbol{u}_d)$, the truncation of its consideration sets is captured by the equality $L_j = \min(\sigma_j(i), \sigma_j(b))$. $\square$

Appendix F: Capacitated Optimization

The approach that we have described extends to the *capacitated* variant of the problem. Specifically, we consider the assortment planning problem wherein at most B products can be stocked. This constraint represents storage or display space constraints, or the limited number of spots of a web page in the context of e-retail and online advertising.

The complexity performance for the different model specifications analyzed in Sections 4 and 6 carries over to the constrained setting, up to a polynomial factor. Specifically, the problem is solved by an extension of our dynamic program. We add a single state variable that encodes the remaining 'capacity' budget for each subproblem, i.e., subproblems are duplicated to account for all possible budget values $[B]$. The new computational tree is inferred by adding edges between any pair of duplicated subproblems that were previously linked by the recursive formula, as long as the budget of the child subproblem is smaller than that of the father subproblem. The recursive formula decides on how to spread the remaining capacity budget across the children subproblems. We prove that, at each step of the recursion, the optimal capacity allocation is determined by solving a shortest path problem that we explicitly describe below. For sake of clarity we will only consider the unique-ranking case wherein $(S,T,\boldsymbol{L}) \sim (S,T)$, but the reasoning is similar for the general algorithm.

State space. The state space is described by the 3-tuple (S,T,b) where b is a new variable that encodes the maximal capacity budget. In this notation, $J(S,T,b)$ designates the maximal expected revenue garnered from customer-types T with an assortment of at most b products in S. The graph and subproblem notations remain unchanged.

Recursion formula. The recursion formula should be generalized to account for all potential different budget allocations. Hence, we introduce $B(b,r)$ the set of all feasible allocations of a capacity of b products between r classes of customers: $B(b,r) = \{\boldsymbol{b} \in \mathbb{N}^r \mid \sum_{j=1}^{r} b_j = b\}$. The recursive formula between subproblems becomes:

$$J(S,T) = \max\Bigg[P_i \cdot \sum_{j\in T(i)} \lambda_j + \max_{\boldsymbol{b}\in B(b-1,r(+))} \sum_{u=1}^{r(+)} J(S_u^+, T_u^+, b_u)\ , \tag{9}$$

$$\max_{\boldsymbol{b}\in B(b,r(-))} \sum_{u=1}^{r(-)} J(S_u^-, T_u^-, b_u)\Bigg] \tag{10}$$

Resource allocation problem. We observe that finding the optimal budget allocations in each max-expression (10) and (9) boils down to solving a simple allocation problem of the form

$$\max_{\sum_{i=1}^{k} b_i \le b} \sum_{i=1}^{k} f(i,b_i)\ ,$$

where the integral non-negative decision variables b_i are coupled by a single constraint. It is well known that this problem can be efficiently solved by means of dynamic programming; see for instance Katoh and Ibaraki (1998).

Appendix G: Synthetic computational experiments

Generative models. The prices of products are sampled independently and identically from a log-normal distribution. The scale parameter is calibrated to reflect realistically the variability of prices in the Shampoo product category. The probability vector is drawn uniformly at random from the unit simplex. To generate instances of the quasi-convex model, the collection of preference lists is formed by independent and uniformly-distributed samples over the class of quasi-convex permutations. To construct instances with arbitrary consideration sets, we use a random Bernoulli generator, as explained in Section 7.1.2. The ranking function is given by the increasing price order.

Implementation of our algorithm. We use a 'plain' implementation of our algorithm which follows the two-pass approach explained in Section 3. First, we generate the computational tree using the recursive equations. Next, we compute the value function by solving a maximum flow problem. In the quasi-convex case, each subproblem is simply encoded by the latest three dynamic programming decisions, leading to an implementation in time $O(N^3K)$.

Appendix H: Estimation methods

For each model, we leverage existing estimation methods, which are described below. Recall that the data is formed by a collection of assortments $\{\mathcal{A}_1,\dots,\mathcal{A}_s\}$, with corresponding purchase probability p_{ij} of product i in the assortment $\mathcal{A}_j$, for each $i\in[N]$ and $j\in[s]$. Also, we let $\mathcal{P}$ designate the probability vector obtained by flattening the matrix $(p_{ij})_{ij}$ in column-major order.

Quasi-convex model. In order to calibrate the quasi-convex model with data, we make use of the column generation ideas developed in related literature by van Ryzin and Vulcano (2014) and Bertsimas and Mišic (2015). To this end, let $\mathcal{L}=\{\mathcal{L}_1,\dots,\mathcal{L}_m\}$ be the collection of *all* quasi-convex preference lists for a given instance (the dependency of m on N has been made explicit by Claim 2). To ease the notation, we assume that the no-purchase option is captured by an alternative in $[N]$. Hence, we introduce the observation tensor $\mathcal{O}=(\mathcal{O}_{i,j,k})\in\{0,1\}^{[s]\times[N]\times[m]}$, where $\mathcal{O}_{i,j,k}=1$ if the preference list $\mathcal{L}_j$ purchases product i in the assortment $\mathcal{A}_k$, and $\mathcal{O}_{i,j,k}=0$ otherwise. In what follows, by abuse of notation, $\mathcal{O}$ will also designate the corresponding $s\cdot N$-by-m matrix (known as the mode-3 unfolding).

In order to estimate a probability distribution $\Lambda=(\lambda_1,\dots,\lambda_m)$ over the quasi-convex preference lists, ideally we would like to solve the following convex program:

$$\begin{aligned} \max \quad & \|\mathcal{O}\cdot\Lambda-\mathcal{P}\|_1 \\ \text{s.t.} \quad & \|\Lambda\|_1\leq 1 \\ & \Lambda\geq 0\ . \end{aligned}$$

Through standard techniques, the latter problem can be recast a linear program with $O(s\cdot N)$ equality constraints. However, the dimensionality (i.e., number of variables) of the resulting linear program remains exponential. Indeed, although quasi-convex preference lists are fewer than the $(N+1)!$ potential lists, m still grows exponentially in the number of products according to Claim 2. On the other hand, the number of equality constraints is small, thus, one could resort to a column generation procedure. Since our procedure is nearly-identical to that of Bertsimas and Mišic (2015), we refer the reader to their paper for a detailed technical description. At a high level, the algorithm alternates between solving a *master* problem and a *column generation* subproblem. At step $t \in \mathbb{N}$, given an incumbent collection $\mathcal{C}_t \subseteq \mathcal{L}$ of quasi-convex preference lists, the master problem solves the ℓ_1-minimization program to find a distribution over $\mathcal{C}_t$ that best fits the data. Next, the subproblem attempts to identify a new quasi-convex list $L \in \mathcal{L} \setminus \mathcal{C}_t$ with lowest reduced cost. Although the column generation step is hard, we solve it approximately through sampling. It is worth noting that, for simplicity, we do not attempt to optimize the central permutation of the quasi-convex structure: as a heuristic, we choose the increasing (average) price order. To control for the risk of over-fitting, our stopping criterion picks the 'best' final step out of $\{100, 200, 300, 400, 500, 600\}$ through cross-validation.

MNL and MMNL models. In order to estimate the MNL and MMNL parameters from data, we use standard maximum likelihood estimation (McFadden 1973, Talluri and Van Ryzin 2006). The corresponding maximum log-likelihood problem is implemented using the optimization software Ipopt (Wächter and Biegler 2006). Contrary to the MNL model, the log-likelihood function associated with the MMNL model is non-concave, and global optimization in this setting is not guaranteed. This estimation method is the standard approach used to calibrate discrete mixtures of MNL (Bierlaire 2003, Hess et al. 2007) .

Appendix I: State space collapse

Table 5 **Relative size of the collapsed state space in comparison to naive enumeration.**

N	K	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$
20	1000	17.8%	4.8%	1.7%
20	2000	22.5%	8.0%	3.6%
100	20	–	$< 0.1\%$	$< 0.1\%$

.

References

Anupindi, Ravi, Sachin Gupta, Munirpallam A Venkataramanan. 2009. Managing variety on the retail shelf: using household scanner panel data to rationalize assortments. *Retail Supply Chain Management*. Springer, 155–182.

Aouad, Ali, Vivek Farias, Retsef Levi, Danny Segev. 2015. The approximability of assortment planning under ranking preferences. Working paper.

Belonax, JJ, Y Mittelstaedt. 1978. Evoked set size as a function of number of choice criteria and information variability. *Advances in Consumer Research* 48–51.

Ben-Akiva, Moshe E, Steven R Lerman. 1985. *Discrete choice analysis: Theory and application to travel demand*, vol. 9. MIT press.

Bertsimas, Dimitris, Velibor V Mišic. 2015. Data-driven assortment optimization. Tech. rep., Working paper, MIT Sloan School.

Bettman, James R, Mary Frances Luce, John W Payne. 1998. Constructive consumer choice processes. *Journal of Consumer Research* **25**(3) 187–217.

Bierlaire, Michel. 2003. Biogeme: a free package for the estimation of discrete choice models. *Swiss Transport Research Conference*.

Blanchet, Jose, Guillermo Gallego, Vineet Goyal. 2013. A markov chain approximation to choice modeling. *Proceedings of the fourteenth ACM conference on Electronic commerce*. ACM, 103–104.

Brandstatter, Eduard, Gerd Gigerenzer, Ralph Hertwig. 2006. The priority heuristic: Making choices without trade-offs. *Psychological Review* **113 (2)** 409–432.

Brisoux, Jacques E, Michel Laroche. 1981. Evoked set formation and composition: An empirical investigation under a routinized response behavior situation. *NA-Advances in Consumer Research* (8).

Bront, Juan José Miranda, Isabel Méndez-Díaz, Gustavo Vulcano. 2009. A column generation algorithm for choice-based network revenue management. *Operations Research* **57**(3) 769–784.

Campbell, Brian Milton. 1969. The existence of evoked set and determinants of its magnitude in brand choice behavior. Ph.D. thesis, Columbia University.

Davis, J., G. Gallego, H. Topaloglu. 2013. Assortment planning under the Multinomial Logit model with totally unimodular constraint structures. Work in Progress.

Davis, James M., Guillermo Gallego, Huseyin Topaloglu. 2014. Assortment optimization under variants of the Nested Logit model. *Operations Research* **62**(2) 250–273.

Dawes, Robyn M. 1979. The robust beauty of improper linear models in decision making. *American Psychologist* **34** 571–582.

Debreu, Gerard. 1960. Review of R. D. Luce, Individual choice behavior: A theoretical analysis **50** 186–188.

Désir, Antoine, Vineet Goyal. 2014. Near-optimal algorithms for capacity constrained assortment optimization. *Available at SSRN 2543309* .

Edmonds, Jack, Rick Giles. 1977. A min-max relation for submodular functions on graphs. *Annals of Discrete Mathematics* **1** 185–204.

Einhorn, Hillel J, Robin M Hogarth. 1975. Unit weighting schemes for decision making. *Organizational Behavior and Human Performance* **13**(2) 171–192.

Farias, Vivek, Srikanth Jagabathula, Devavrat Shah. 2013. A non-parametric approach to modeling choice with limited data. *Management Science* **59 (2)** 305–322.

Fisher, Marshall L., Ramnath Vaidyanathan. 2009. An algorithm and demand estimation procedure for retail assortment optimization with results from implementation. Working paper (Philadelphia: The Wharton School).

Gigerenzer, Gerd, Daniel G Goldstein. 1996. Reasoning the fast and frugal way: models of bounded rationality. *Psychological review* **103**(4) 650.

Gigerenzer, Gerd, Reinhard Selten. 2002. *Bounded rationality: The adaptive toolbox*. MIT press.

Gilbride, T. J., G. M Allenby. 2004. A choice model with conjunctive, disjunctive, and compensatory screening rules. *Marketing Science* **23**(3) 391–406.

Gurobi Optimization, Inc. 2015. Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

Hauser, John R. 1978. Testing the accuracy, usefulness and significance of probabilistic models: An information theoretic approach. *Operations Research* 406–421.

Hauser, John R., Min Ding, Steven P. Gaskin. 2009. Non compensatory (and compensatory) models of consideration-set decisions. *Sawtooth Software Conference Proceedings* .

Hauser, John R., Birger Wernerfelt. 1990. An evaluation cost model of consideration sets. *Journal of Consumer Research* 393–408.

Hess, Stephane, Michel Bierlaire, John Polak. 2007. A systematic comparison of continuous and discrete mixture models. *European Transport* (37).

Honhon, D., S. Jonnalagedda, X. A. Pan. 2012. Optimal algorithms for assortment selection under ranking-based consumer choice models. *Manufacturing & Service Operations Management* **14**(2) 279–289.

Howard, J. A., J. N. Sheth. 1969. *The Theory of the Buyer Behavior*. John Wiley.

Jagabathula, Srikanth, Paat Rusmevichientong. 2016. A nonparametric joint assortment and price choice model. *Management Science* (Articles in advance). doi:10.1287/mnsc.2016.2491.

Katoh, Naoki, Toshihide Ibaraki. 1998. Resource allocation problems. *Handbook of combinatorial optimization* **2** 159–260.

Kök, A Gürhan, Marshall L Fisher. 2007. Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research* **55**(6) 1001–1021.

Kök, A Gürhan, Marshall L Fisher, Ramnath Vaidyanathan. 2009. Assortment planning: Review of literature and industry practice. *Retail supply chain management*. Springer, 99–153.

Lancaster, Kelvin. 1975. Socially optimal product differentiation. *The American Economic Review* 567–585.

Lancaster, Kelvin J. 1966. A new approach to consumer theory. *The Journal of Political Economy* 132–157.

Laroche, Michel, Chankon Kim, Takayoshi Matsui. 2003. Which decision heuristics are used in consideration set formation? *Journal of Consumer Marketing* **20**(3) 192–209.

Li, Guang, Paat Rusmevichientong, Huseyin Topaloglu. 2015. The d-level Nested Logit model: Assortment and price optimization problems. *Operations Research* **63**(2) 325–342.

Mahajan, Siddharth, Garrett Van Ryzin. 2001. Stocking retail assortments under dynamic consumer substitution. *Operations Research* **49**(3) 334–351.

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

McFadden, Daniel. 1973. Conditional Logit analysis of qualitative choice behavior. *Frontiers in Econometrics* 105–142.

McFadden, Daniel, Kenneth Train. 2000. Mixed MNL models for discrete response. *Journal of applied Econometrics* 447–470.

Parkinson, T. L., M. Reilly. 1979. An information processing approach to evoked set formation. *Advances in Consumer Research* **6**(1) 227–231.

Payne, J. W., J. Bettman, R. James, M. F Luce. 1996. When time is money: Decision behavior under opportunity-cost time pressure. *Organizational behavior and human decision processes* **66**(2) 131–152.

Pentico, David W. 1974. The assortment problem with probabilistic demands. *Management Science* **21**(3) 286–290.

Posavac, Steven S., Tracy Meyer, Frank R. Kardes, James J. Kellaris. 2005. A selective hypothesis testing perspective on price-quality inference and inference-based choice. *Journal of Consumer Psychology* **15 (2)** 159–169.

Pras, Bernard, John Summers. 1975. A comparison of linear and nonlinear evaluation process models. *Journal of Marketing Research* 276–281.

Ratliff, Richard M, B Venkateshwara Rao, Chittur P Narayan, Kartik Yellepeddi. 2008. A multi-flight recapture heuristic for estimating unconstrained demand from airline bookings. *Journal of Revenue and Pricing Management* **7**(2) 153–171.

Reilly, Michael, Thomas L Parkinson. 1985. Individual and product correlates of evoked set size for consumer package goods. *Advances in Consumer Research* **12**.

Roberts, John H, James M Lattin. 1991. Development and testing of a model of consideration set composition. *Journal of Marketing Research* 429–440.

Rusmevichiengtong, Paat, Benjamin Van Roy, Peter W. Glynn. 2006. Nonparametric approach to multi-product pricing. *Operations Research* **54**(1) 82–98.

Rusmevichientong, Paat, Zuo-Jun Max Shen, David B Shmoys. 2010. Dynamic assortment optimization with a Multinomial Logit choice model and capacity constraint. *Operations Research* **58**(6) 1666–1680.

Rusmevichientong, Paat, David Shmoys, Huseyin Topaloglu. 2014. Assortment optimization under the Multinomial Logit model with random choice parameters. *Production and Operations Management* **23**(11) 2023–2039.

Rusmevichientong, Paat, Huseyin Topaloglu. 2012. Robust assortment optimization in revenue management under the Multinomial Logit choice model. *Operations Research* **60**(4) 865–882.

Samet, Hanan. 1990. Applications of spatial data structures. Addison-Wesley.

Silk, Alvin J, Glen L Urban. 1978. Pre-test-market evaluation of new packaged goods: A model and measurement methodology. *Journal of marketing Research* 171–191.

Talluri, Kalyan, Garrett van Ryzin. 2004. Revenue management under a general discrete choice model of consumer behavior. *Management Science* **50**(1) 15–33.

Talluri, Kalyan T, Garrett J Van Ryzin. 2006. *The Theory and Practice of Revenue Management*, vol. 68. Springer Science & Business Media.

Tversky, A. 1972a. Choice by elimination. *Journal of Mathematical Psychology* **9** 341–367.

Tversky, A. 1972b. Elimination by aspects : A theory of choice. *Psychological Review* **79** 281–299.

Tversky, A., S. Sattath. 1979. Preference trees. *Psychological Review* **86** 542–573.

Tversky, Amos, Daniel Kahneman. 1975. Judgment under uncertainty: Heuristics and biases. *Utility, probability, and human decision making*, vol. 185. Springer, 141–163.

Urban, Glen L. 1975. Perceptor: A model for product positioning. *Management Science* **21**(8) 858–871.

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

Vulcano, Gustavo, Garrett van Ryzin, Wassim Chaar. 2010. Choice-based revenue management: An empirical study of estimation and optimization. *Manufacturing & Service Operations Management* **12**(3) 371–392.

Wächter, Andreas, Lorenz T Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming* **106**(1) 25–57.

Zeithalm, Valarie A. 1988. Consumer perception of price, quality and value: a means-end model and synthesis of evidence. *Journal of Marketing* **52** 2–22.