Assortment Optimization Under Consider-then-Choose Choice Models

Ali Aouad
Management Science and Operations, London Business School, Regent's Park, London, United Kingdom NW14SA, aaouad@london.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 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 predictive power of our modeling approach on a combination of synthetic and real industry data sets, 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 computationally hard problems. The present paper demonstrates that a class of choice models, referred to in the literature as *consider-then-choose* models, renders assortment optimization tractable under a broad 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 data sets. 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's utility is specified through a relatively small number of parameters. This parametric approach has various benefits, including the computational tractability of the resulting choice models and their ability to incorporate contextual product and customer features. 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 common 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. These models posit structural assumptions about the customers' substitution behavior (Debreu 1960, Ben-Akiva and Lerman 1985). However, in attempting to consider further generalized models, such as a discrete mixture of MNL models (MMNL), the assortment optimization is no longer easy with the best known algorithms having a complexity that scales exponentially in the number of customer segments (Bront et al. 2009, Rusmevichientong et al. 2014, Désir and Goyal 2014).

In an attempt to construct a data-driven nonparametric approach to choice modeling, several studies in operations management (Rusmevichiengtong et al. 2006, Farias et al. 2013, van Ryzin and Vulcano 2014) have considered choice models where preferences are explicitly expressed as distributions over ranked lists. 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) have developed 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. However, 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 recently shown to be NP-hard, even to approximate by Aouad et al. (2018).

Consider-then-choose models. As explained earlier, in its utmost generality, the nonparametric assortment optimization problem is subject to strong computational limitations. As such, this study seeks to identify structural properties and behavioral assumptions that yield tractable assortment optimization formulations. Specifically, the aforementioned parametric and non-parametric models place a general condition on the customer's decision making process, where customers list *all* her options available and then pick her most desirable from that list (for example by specifying a utility function over every and each product). 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 restricted set of products considered. The history of these consider-then-choose ideas 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 in a given population. We review the evolution of this approach to modeling choice in our literature review in Sections 1.2 and 4.

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. Furthermore, our modeling approach is tested on real purchase panel data sets. We observe that the fitted consider-then-choose models achieve superior predictive accuracy in comparison with commonly-used parametric choice models.
2. This consider-then-choose structure can be leveraged to mitigate the complexity of assortment optimization problems. Indeed, a close examination of the reduction proof by Aouad et al. (2018) reveals that the primary source of complexity in assortment optimization problems stems from the structure of the customers' consideration sets. 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 classes 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 data sets. Our industry partner 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 the key ingredient of our algorithmic 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 subject to a cardinality constraint on the size of the assortment, using more general objective functions (e.g. social welfare). These results are separately described in Appendix EC.3.

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.

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
Laminar properties		$O(N^2K^2)$	4.2
Disjunction on d features		$O(N^3K^{d+2})$	4.3
Intervals	Quasi-convex permutations	$O(N^3K)$	6.1
Two-feature compensatory	Two-feature compensatory	$O(N^4K^3)$	6.2

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.

We derive polynomial running time bounds 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 analytical findings and the corresponding structural assumptions are summarized in Table 1 and further discussed in Section 2.

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 observe that the IP approach is intractable to solve large-scale instances of the quasi-convex model, contrary to the proposed dynamic program approach. Even under generic consideration set structures, our approach dominates the IP solver in several regimes of parameters.

Lastly, we demonstrate in Section 7.2 the predictive power of our modeling approach against a benchmark formed by "small" mixtures of Multinomial Logits (MMNL). The goal is to predict the purchase probabilities of products in various assortments. We conduct numerical experiments on real-world data sets provided by our industry partner, in three distinct product categories. The errors in out-of-sample predictions of the purchase probabilities are reduced under standard metrics in all data sets. In synthetic experiments, our consider-then-choose model outperforms the benchmarks under several realistic ground truth models.

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 have 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 the operations management literature. Most of the focus has been on variants of this problem under the

widespread attraction-based models such as the Multinomial Logit (MNL) model, the discrete Mixture of MNLs (MMNL), etc. Under MNL preferences, the problem was shown 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, such as robust optimization formulations (Rusmevichientong and Topaloglu 2012), and totally-unimodular constraints (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. The computationally efficiency is contingent on modeling a small number of customer segments. Given these computational barriers, recent work in assortment optimization attempts to identify new probabilisitic models leading to tractable assortment optimization problems (Blanchet et al. 2013, Davis et al. 2014, Li et al. 2015).

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., all attraction-based models can be viewed as parametrized distributions over preference lists. 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. On the positive side, Honhon et al. (2012) developed tailor-made dynamic programming ideas for several special cases, which are subsumed by our analytical results. On the other hand, Aouad et al. (2018) characterized the approximability class under generic distributions, showing that the assortment optimization problem is hard to approximate within any sub-linear factor in the number of products.

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. One key idea in this literature is that individuals apply simple decision heuristics in order to whittle down their choices to a restricted set of alternatives (Howard and Sheth 1969), and thereby, alleviate the cognitive burden in multi-alternative decision tasks (Tversky and Kahneman 1975). This idea has been justified through various empirical findings.

- *Explanatory power:* In effect, the consideration set structure conveys important information about the customer choices in a statistical sense. The seminal work of Hauser (1978) showed 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 uncertainty in customer choices 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 product categories (Belonax and Mittelstaedt 1978, Reilly and Parkinson 1985).
- *Predictive power:* The notion of consideration sets has been popularized in data applications through the observation that two-stage decision models (i.e., combining consideration set formation with random utility maximization) generally yield more accurate predictions. This observation is borne out by empirical studies in several application domains, including market share forecasting for new product launches (Urban 1975, Silk and Urban 1978), customer choice modeling (Roberts and Lattin 1991), and more recently, in the area of risky decision-making (Brandstatter et al. 2006).
- *Structural modeling:* Another stream of literature has been focused on justifying the consideration sets through rational agent models. When the consideration sets are formed endogenously, the screening stage enables to balance search efforts with potential gains; see for example the models developed by Hauser and Wernerfelt (1990), Roberts and Lattin (1991) or Payne et al. (1996). Furthermore, screening heuristics were shown to be rational under limited time and knowledge (Gigerenzer and Goldstein 1996, Gigerenzer and Selten 2002).

Motivated by this line of literature, the modeling approach we develop subsequently is centered around the notion of consideration sets. In Section 2, we review several families of consideration set models, commonly used in marketing studies, that are considered in this paper. It is worth mentioning that, prior to our work, the paper by Jagabathula and Rusmevichientong (2016) also incorporates a choice model based on consideration sets. However, the optimization problem considered therein relates more closely to combinatorial pricing.

2. Modeling Approach and Problem Formulation

In what follows, we begin by formulating the assortment optimization problem in its general form. Next, we will summarize various modeling assumptions considered in this paper, against which assortment optimization is shown to be tractable.

2.1. Assortment Optimization Problem

Throughout the paper we use the index $i \in \{1,\ldots,N\} = [N]$ to denote one of N products, each is associated with a price P_i. In addition, we use the index $j \in \{1,\ldots,K\} = [K]$ 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 $(\lambda_1,\ldots,\lambda_K)$ 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 $\mathrm{Rev}\,(j,\mathcal{A})$ 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 $\mathrm{Rev}\,(j,\mathcal{A}) = 0$ and otherwise $\mathrm{Rev}\,(j,\mathcal{A}) = P_{i(\mathcal{A},j)}$, 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 $\mathcal{A}$. In other words, the no-purchase events corresponds to cases where the assortment does not intersect with the customer's consideration set. Therefore, the objective is to find an assortment $\mathcal{A}$ that maximizes the expected revenue: $\sum_{j\in[K]} \lambda_j \cdot \mathrm{Rev}\,(j,\mathcal{A})$.

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 will provide a bird-eye 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.

2.2. Modeling Approach

In order to model $\mathcal{C}$ and Σ, we turn our attention to common assumptions used in marketing studies to describe the process by which customers whittle down the universe of products to generate their consideration set. In their literature survey, Hauser et al. (2009) explain that a consideration set can generally be viewed as the outcome of various *screening rules*. Roughly-speaking, a screening rule is a minimal requirement relative to one given product feature. To formalize this notion, suppose that each product $i \in [N]$ is represented in by a vector $\mathbf{x}^{(i)} \in \mathbb{R}^d$ in a d-dimensional feature space. With this notation at hand, a screening rule designates a cut-off level $t_e \in \mathbb{R}$ on some given feature $e \in [d]$, implying that only products $i \in [N]$ satisfying $x_e^{(i)} \geq t_e$ are considered. An important line of research in marketing aims at understanding how the formation of consideration sets can be modeled as a combination of screening rules. Generally-speaking, these approaches can be classified into *conjunctive*, *disjunctive* and *compensatory* models. In what follows, we introduce each model class in turn, and provide a roadmap of the technical results obtained in the paper.

Conjunctive models. One common assumption is that customers form their consideration sets through a *conjunction* of screening rules, i.e., a product is considered if *each one* of the specified screening rules are *all* satisfied. (In other words, features are used to eliminate products, and

customers are increasingly picky as more features are presented to them.) Mathematically speaking, we assume that for each consideration set $C \in \mathcal{C}$, there exists a vector $(t_1, \ldots, t_d)$ such that

$$C = \bigcap_{e \in [d]} \{i \in [N] : x_e^{(i)} \geq t_e\} \ .$$

It is worth noting that the conjunctive models have been supported by a number of empirical studies that benchmark the explanatory power of various consideration set models, such as the papers by Pras and Summers (1975), Brisoux and Laroche (1981), Laroche et al. (2003) to mention a few. More recently, through the structural estimation of a general two-stage choice model, Gilbride and Allenby (2004) finds that the parameters associated with conjunctive screening rules are statistically significant, contrary to other screening rules.

It is easy to verify that, for a large enough dimension d, any arbitrary collection of consideration sets $\mathcal{C}$ can be expressed through the outcomes of a conjunctive model. As such, in this paper, we will either require that the number of features is small, or we operate under additional structural assumptions. Specifically, in Section 4, we study the following settings (and other extensions):

- *Intervals (Section 4.1).* We assume that $d = 2$ and the features are inversely-related features. One canonical example is when customers have price and quality cut-offs. The consideration sets $\mathcal{C}$ arising from the combination of the two screening rules form a collection of intervals.
- *Laminar properties (Section 4.2).* Here, the combinations of screening rules are described by paths in a rooted tree, where each node represents a screening rule. This tree can be interpreted as the decision tree describing the preferences of the customer population. In this setting, the number of features d can be arbitrarily large, yet the tree structure imposes additional restrictions. If we interpret the combination of screening rules as the outcome of a random experiment, the tree structure imposes probabilistic correlations across the screening rules.

Disjunctive models. The conjunctive model has often been benchmarked against an alternative approach, known as *disjunctive rule models*. In this setting, a product is considered if *at least* one of specified screening rules is satisfied (customers are less picky when more features are presented to them.) In formal terms, we assume that for each consideration set $C \in \mathcal{C}$, there exists a a vector $(t_1, \ldots, t_d)$ such that

$$C = \bigcup_{e \in [d]} \{i \in [N] : x_e^{(i)} \geq t_e\} \ .$$

It is worth mentioning that the disjunctive models are far less popular than their conjunctive counterpart; see for example the empirical studies conducted by Pras and Summers (1975), Gilbride and Allenby (2004).

In Section 4.3, we show that assortment optimization is tractable under this model specification for a small number of features d. Specifically, we establish a parametric complexity bound that grows polynomially for a fixed d.

Compensatory models. Finally, we consider the case where the customers' decision process is *compensatory*, i.e., lower levels on certain features can be compensated by higher levels on other features. Here, the consideration sets are formed through a linear combination of screening rules. Specifically, for each consideration set $C \in \mathcal{C}$, there exists a utility vector $\mathbf{u} \in \mathbb{R}^d$ (i.e., each entry of this vector is a feature weight) along with a cut-off level $t \in \mathbb{R}$ such that

$$C = \{i \in [N] : \mathbf{u} \cdot \mathbf{x}^{(i)} \geq t\} \ .$$

This approach has received some attention in empirical studies. Interestingly, the compensatory model was shown to be robust in explaining the customers' decision process even when features are uniformly weighted, i.e., $\mathbf{u} = \mathbf{1}$; see the works of Einhorn and Hogarth (1975) and Dawes (1979).

Similar to the above models, the compensatory model can subsume any collection $\mathcal{C}$ when the feature space is high dimensional. Our main result is to establish that assortment optimization is tractable under a two-dimensional feature spaces, i.e., $d = 2$.

Ranking heterogeneity. The above discussion has been centered on the collection of consideration sets $\mathcal{C}$. We now describe the collections of rankings Σ considered in the paper. 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 setting 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. (2018) that the problem under the unique-ranking model is NP-hard to approximate within factor $O(N^{1-\epsilon})$ for any $\epsilon > 0$.

Our algorithmic approach and analysis carry over in the presence of heterogeneity in ranking decisions. Specifically, the unique-ranking assumption is relaxed in the following cases:

- Σ is formed by *similar* rankings arising from the local perturbations of a central permutation (*Appendix EC.1.5*). That is, the relative preference order between two products i, j can be swapped only when $|i - j| \leq O(1)$.
- Σ is the class of *quasi-convex* permutations (*Section 6.1*). Here, the ranking function is unimodal with respect to some numbering of the product.
- Σ is formed by two-dimensional linear utility functions (*Section 6.1*). Specifically, in the compensatory setting described above, the ranking is inferred by the utility function $u(i) = \mathbf{u} \cdot \mathbf{x}^{(i)}$ over the products $i \in C$.

We refer the reader to Section 6 for a detailed mathematical description of these settings. The technical claims established in the paper for combinations of $\mathcal{C}$ and Σ are summarized in Table 1.

Additional remarks. It is worth mentioning that, while each model is motivated by an explicit feature representation, the proposed algorithmic approach only makes use of the combinatorial properties of $\mathcal{C}$ and Σ. Thus, we will treat the product features as *latent* when estimating the corresponding choice models in Section 7.2. In addition, we emphasize that the above discussion focuses on the consideration set models studied in this paper; therefore, it is far from being exhaustive. For a broader review of the existing models, we refer the reader to the surveys by Hauser et al. (2009), Payne et al. (1996) or Bettman et al. (1998) and to the references therein.

3. Dynamic Program Under Unique-Ranking Distributions

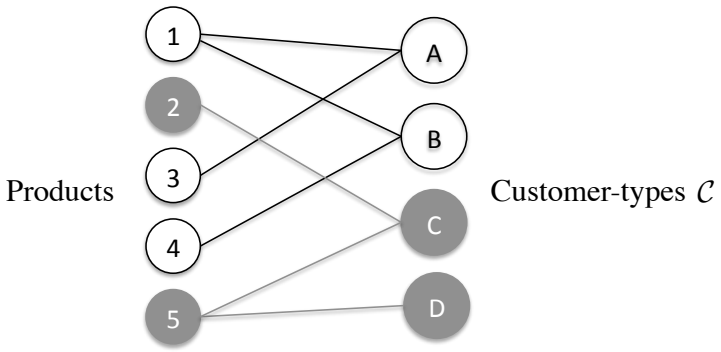
In this section, we present a dynamic programming (DP) formulation under unique-ranking distributions. As some obstacles must be surmounted to consummate our approach in the general case, the algorithm for arbitrary preference list distribution is described separately in Section 5, to ease the exposition. 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, product 1 is the most preferred one, 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 $\{1, \ldots, X\}$. For any set $S \subseteq [N]$, we use $\min(S)$ and $\max(S)$ to denote the minimal and maximal index products in S, respectively.

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 their arrival probabilities are directly induced from the original instance without normalization. Formally, the subproblem (S, T) is formulated as follows:

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

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



$$J([5], \mathcal{C}) = J(\{1,3,4\}, \{A,B\}) + J(\{2,5\}, \{C,D\})$$

Figure 1 Decomposition of the instance according to the connected components of the graph representation.

the partition of $G[S,T]$ into its connected components[1] captures a decomposition of the instance (S,T) into several subproblems that can be solved independently. This decomposition scheme, represented in Figure (1), is the 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). □

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

[1] A connected component in a undirected graph (V,E) is a set of nodes S which are all mutually connected (i.e., between every two nodes $i,j \in S$, there exists a path formed by edges of E), and there does not exists a path from a node in $V \setminus S$ to a node in S.

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 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 size of 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* $K = O(\log(N))$.

State space mapping. In order to prove Theorem 1, we introduce a fundamental characterization of the state space. Specifically, we construct an injective mapping from any state of the recursion $(S,T) \in \mathcal{S}$ to a subset of the products $\{1,\dots,N\}$[2]. As explained below, this mapping is the central piece of our analysis in the next sections, and considerably simplifies the proof techniques used to upper bound the running time under specific consideration set models.

Specifically, we define the 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:

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

In other words, each subproblem (S,T) is mapped to the set of products exactly covered by the consideration sets T in the interval $[\min(S)]$. In the following lemma, we show that Φ is injective, meaning that the size of the state space is equal to $|\Phi\langle\mathcal{S}\rangle|$. Intuitively, the mapping $\Phi(\cdot)$ provides a transformation of the state space into a simplified form, through which we can uniquely count the connected subgraphs generated by the recursion.

[2] A function f is said to be injective when $x \neq y$ imply that $f(x) \neq f(y)$.

LEMMA 2 **(Central Lemma)**. *The mapping $\Phi : \mathcal{S} \to 2^{[N]}$ is injective, and thus, $|\mathcal{S}| = |\Phi\langle\mathcal{S}\rangle|$.*

In order to establish Theorem 1, it remains to observe that for every state $(S,T) \in \mathcal{S}$, the definition of $\Phi(S,T)$ only depends on $\min(S)$ and T, for which there exists at most $N \cdot 2^K$ unique combinations. On the other hand, $\Phi\langle\mathcal{S}\rangle \subseteq \mathcal{P}([N])$, and thus 2^N is a natural upper bound on the size of the state space.

Proof of Lemma 2 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.

Since 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 contained in $[\min(S_1)-1]$, 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. The proof is deferred to Appendix EC.1.1.

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

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 $i \notin S_1$. By noting that $\min(S_2) = \min(S_1)$, 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, 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 Lemma 2, by observing that the projected sets in $\Phi\langle\mathcal{S}\rangle$ have a diameter smaller than $\mathrm{Diam}(\mathcal{C})$.

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

4. Consideration Set Structures

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* $O(N^4K)$.

Proof Given that the function Φ is injective according to Lemma 2, 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 $[N]$. It is worth noting that, in 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 $O(N^2)$. $\square$

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 $C_{j'} \subseteq C_j$.

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 $x_e^{(i)} = t$ 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 $x_{e_k}^{(i)} = t_{e_k}$. 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).

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. In addition, to illustrate the robustness of our algorithmic approach, the dynamic program is examined in Appendix EC.2.4 under a weaker notion of laminar consideration sets.

Theorem 3. *The class of laminar consideration sets is a special case of induced intervals. The corresponding running time is of $O(N^4 K)$.*

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

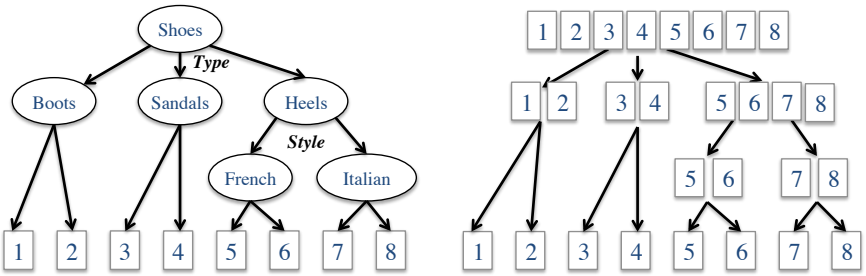


Figure 2 **An Elimination-by-Aspect screening process and the corresponding laminar tree, for the example of purchasing 'shoes', with two features: 'type' and 'style'.**

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 induction hypothesis implies that $\pi\langle \cup_{j' \in o(j)} C_{j'} \rangle$ is an interval.
3. Since this interval is concatenated to $s(j)$, observation 1 implies that $\pi\langle C_j \rangle$ is an interval. □

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 3. 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

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

Next, we show that the size of the state space $\mathcal{S}$ is polynomially bounded for a fixed parameter d.

THEOREM 4. *Under d-disjunctive consideration sets, the dynamic program has a running time of* $O(N^2 K^{d+1})$.

The proof is deferred to Appendix 4. Since the d-disjunctive model can capture any arbitrary consideration set structure $\mathcal{C}$ for a large enough d, this 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), meaning that d is relatively small.

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). 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. 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.

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.

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 $J(S, T, \boldsymbol{L}) = \sum_{u=1}^{r} J(S_u, T_u, \boldsymbol{L})$.*

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.* Since they are already satisfied with a product, the nodes corresponding to consumer-types V are discarded, and we denote by $T^{(V)} = T \setminus V$ 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:

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

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 he prefers more at some later point of the recursion. As a result, we need to truncate his preference list by updating the vector $\boldsymbol{L}$:

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

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 $P_i \cdot \sum_{j \in V} \lambda_i$. 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:

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

Correctness of the algorithm. At face value, the above recursion does not compute the exact expected revenue generated by a sequence of allocation decisions. Indeed, suppose that $T(i) \setminus V \neq \emptyset$. We do not enforce that all customer-types in $T(i) \setminus V$ are allocated with another product $i' \neq i$ at the subsequent steps of the recursion. Therefore, the above dynamic program provides us with a lower bound on the expected revenue. However, we argue that any optimal sequence of dynamic programming decisions $(i_1, V_1), \ldots, (i_T, V_T)$ necessarily necessarily satisfies $\cup_{t=1}^{T} V_t = \cup_{t=1}^{T} T(i_t)$, and thus, the recursion computes the exact optimal expected revenue. To this end, without loss of generality, we assume that the arrival rates λ_j and prices P_i are strictly positive (otherwise the corresponding customer-types and products can be deleted). Next, suppose there exists a customer-type $j \in \cup_{t=1}^{T} T(i_t) \setminus \cup_{t=1}^{T} V_t$ and let i_u be a product such that $j \in T(i_u)$. Assuming that we only change the allocation V_u by adding customer-type j, then it is not difficult to verify that all other product allocations remain feasible. However, since the arrival rates and prices are strictly positive, the total value computed by the dynamic program may only increase. This contradicts the optimality of the sequence of decisions.

The marginalized dynamic program. 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, unlike 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. Even if the state space is tightly characterized, it is still not obvious how to efficiently solve the optimization problem described by (4).

We address the difficulties raised above by proposing an efficient algorithm to generate the computational tree and solve the recursive formula (4), while avoiding an enumeration over all allocations $V \subseteq T(i)$. At a high-level, our main idea is to *marginalize* the allocation decisions. 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 of $T(i)$. Consequently, we can show that the number of descendant subproblems is polynomial in N, K and $|\mathcal{S}|$, yielding the following complexity result.

PROPOSITION 1. *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 the worst case, we have $|\mathcal{S}| \leq N \cdot 2^N$.*

To avoid lengthy technical details, the specifics of the marginalized dynamic program and the proof of the above claim are deferred to Appendix EC.1.4.

6. Consider-then-Choose Models with Ranking Heterogeneity

In this section, we study distributions over preference lists that combine consideration set heterogeneity along with ranking heterogeneity. One natural extension is to consider small "perturbations" of the unique-ranking setting. This notion is formalized in Appendix EC.1.5; we show that the computational bounds established in Section 4 carry over assuming that the rankings lie in a "small neighborhood" around a given permutation. In what follows, we explore consider-then-choose models that exhibit high levels of ranking heterogeneity.

6.1. Quasi-convex model

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

The quasi-convex model substantially increases the "degrees of freedom" of the distributions up to $O(2^N)$ - in comparison, the intervals model only has $O(N^2)$ parameters and the locational model only has $O(N^3)$ parameters (see Claim EC.3 established in Appendix EC.1.6).

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. We further assume that price and quality are reversed features. In this setting, the quasi-convex rankings include price-driven purchasing behaviors (products are ranked by decreasing prices), as well as quality-seeking purchasing behaviors (products are ranked by decreasing quality). More generally, every customer-type $j \in [K]$ may have an idiosyncratic assessment of the product quality $q_1^j, q_2^j, \ldots, q_N^j$. If quality $(q_i^j)_{i \in [N]}$ is a concave function of price, i.e., there exists a concave function $f^j : \mathbb{R}^+ \to \mathbb{R}^+$ such that $q_i^j = f^j(P_i)$ for every $i \in [N]$, then the quasi-convex model captures customers who rank products by decreasing utilities, under the utility functions $u^j(i) = q_i^j - \beta \cdot P_i$ where $\beta > 0$, and $u^j(i) = q_i^j / P_i$. In addition, the quasi-convex model subsumes the one-dimensional *locational choice model* (Lancaster 1966, 1975). In the this 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).

In the next theorem, we show that our dynamic program is efficient under the quasi-convex model for a suitable choice of the processing order.

Theorem 5. *Under the quasi-convex model with central permutation σ, the dynamic program with processing order σ has a state space of size $O(N^2)$.*

The proof is provided in Appendix EC.1.7. The main technical ideas are similar to the construction of the injective mapping in Lemma 2. Specifically, each subproblem of the recursion is uniquely characterized using only two products examined in the computational tree.

6.2. 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.

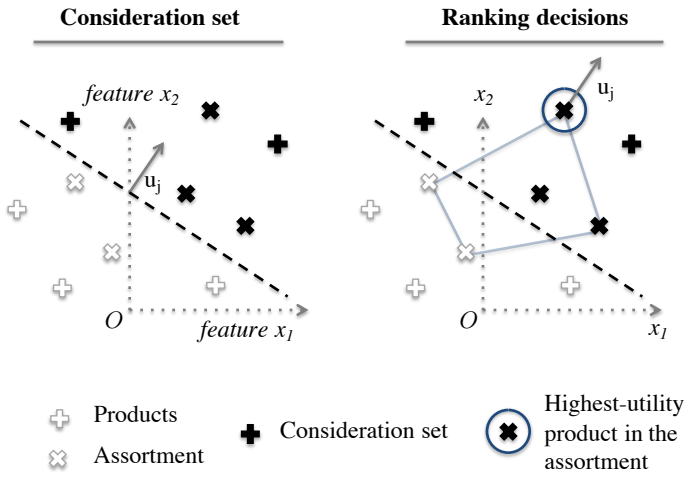


Figure 3 **Consideration sets and ranking decisions driven by linear utility maximization with two features.**

Definition 5. An distribution over preference lists belongs to the two-feature compensatory model if every 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 every 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 linear utility models, we prove that the state space is of polynomial size under the class of preference list distributions described by the two-feature compensatory model, as stated by the next theorem.

Theorem 6. *Under the two-feature compensatory model, for any arbitrary processing order, the size of the state space is of* $O(N^3K^2)$.

The proof, deferred to Appendix EC.1.8, is in the same vein as that of Theorem 5. By representing each state of the recursion as a polyhedron in the feature space, we construct an injective mapping from subproblems in $\mathcal{S}$ onto the preceding products examined in the computational tree.

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 data sets, against several parametric choice models.

7.1. Computational performance

Our dynamic programming algorithm is benchmarked against a natural integer programming (IP) formulation, implemented using a state-of-the-art commercial solve. The mathematical formulation is provided in Appendix EC.2.1, along with further details on our implementation.

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 IP formulation is implemented using the commercial IP solver GUROBI (v.6.5). 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 batches of experiments with different generative models. In the former, we generate random 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 are generated via i.i.d Bernoulli trials with a parameter $\alpha \in (0,1)$. This approach is described in greater detail in Appendix EC.2.1.

Numerical results. The running time of the algorithms on quasi-convex instances is reported in Table 2. Each entry is generated by sampling 50 random instances, unless the average running time exceeds 800 seconds, in which case we sample 20 instances. Our numerical results indicate that our algorithmic approach is very efficient and outperforms the IP 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. Another potential shortcoming of the IP approach for practitioners resides in the large variability of the running time across instances.

To keep the paper concise, the numerical results for our second batch of experiments, on randomly-generated unique-ranking instances, are reported in Appendix EC.2.2. In summary, our dynamic programming approach outperforms the IP in several settings. The IP solver scales poorly

Table 2 Runtime of our algorithm (DP) against the commercial solver (IP) under the quasi-convex 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	$> 10^3$	$< 10^{-3}$	-

with respect to the number of customer types (see Figure EC.3). 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 large consideration sets (a large Bernoulli parameter α). On the other hand, as one would expect, the dynamic program is less efficient when $N >> K$, since it enumerates over product stocking decisions. In general, the computational efficiency of our dynamic program hinges on the efficiency of the state space collapse. As shown by Table EC.1, in comparison to a "naive" recursion, the size of the state space is reduced by a factor ranging between 75% to over 99% (see Table EC.1).

7.2. Predictive performance

Practical applications of choice modeling, such as the assortment optimization problem studied here, begin with *transactional data*. The standard 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 (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 respect to the number of customer segments. Consequently, optimization is practical 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 data set 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 6.1. Surprisingly, in certain settings, the quasi-convex model provides a better fit to the data (out-of-sample) under various metrics.

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 provides a better fit by a large margin.
3. *Purchase panel data:* Using transactional data across a panel of hundreds of thousands of customers in three distinct product categories (containing tens of products) collected by an industry partner, we again run the same experiment, and evaluate predictive power on a hold-out data set. We show that the quasi-convex consider-then-choose model provides a significant improvement in predictive accuracy on the hold out set.

MMNL benchmarks. 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 $w_{i,j} \in \mathbb{R}^+$ 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:

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

The best known algorithm for the MMNL(c) assortment optimization problem runs in time $O((\log N)^{2c} \cdot N^{2c+1}/\epsilon^{4c}))$, where ϵ is the chosen level of accuracy (Désir and Goyal 2014). Hence, assortment optimization is effectively impractical for large mixtures, and as such we eventually fit MMNL models with up to $c = 3$ customer segments.

Data-driven estimation. In our subsequent empirical settings, the data observations take the form of a sequence of assortments $\mathcal{A}_1, \ldots, \mathcal{A}_\tau$ and a purchase probability matrix $(p_{it})_{i \in [N], t \in [\tau]}$, where p_{it} is the purchase probability of product i within the assortment $\mathcal{A}_t \subseteq [\tau]$. All choice models are estimated using the same training data sets, and they are evaluated on the same hold-out data sets. To this end, we leverage standard estimation methods developed in the related literature. These methods are described in detail in Appendix EC.2.3 with additional information on their implementation. In a nutshell, the MMNL choice models are fitted using maximum-likelihood estimation (MLE). The quasi-convex model is estimated by adapting the column generation ideas proposed in the anterior literature for nonparametric choice models (van Ryzin and Vulcano 2014, Bertsimas and Mišic 2015). Interestingly, we derive a new structural result showing that a critical step of

this estimation methodology is provably tractable under the quasi-convex model. Specifically, we show that the column generation step, known to be NP-hard under arbitrary distributions over preference lists, admits a polynomial time dynamic programming algorithm in our setting. This result suggests that the quasi-convex property eases the estimation process.

Experiments on synthetic data. We first explain how the synthetic data sets are generated, and then describe our numerical results. The data set 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. Each customer segment occurs with probability $\mu_1 = \cdots = \mu_5 = 1/5$. The preference weights $w_{i,j}$ are drawn independently from a log-normal distribution of scale σ, where σ is varied in the set $\{0, 1, 10, 20\}$. Here, σ controls the heterogeneity of the ranking behavior across customer segments; we will momentarily see that predictive performance is sensitive to this parameter. When $\sigma = 0$, the model is equivalent to an instance of the MNL model.
- *Intervals model:* For our second set of synthetic data, the purchase probabilities are generated using an instance of 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.

For each data set thus formed, we carried out a 5-fold cross-validation to estimate the out-of-sample prediction accuracy of the different models. This procedure is repeated over 10 randomly-generated instances of the ground truth. The prediction errors are measured by the mean squared error (MSE), expressed in normalized form as a percentage of the total variance of the data. Namely, letting $\mathcal{OS} \subseteq [100]$ designate the collection of out-of-sample assortments, we have:

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

where $(\hat{p}_{i,t})_{i,t \in [N] \times [\tau]}$ are the model estimates. The numerical results are reported in Table 3. We observe that the quasi-convex model 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 quasi-convex instances are accurately recovered by our estimation method, and the fitted models attain low out-of-sample MSEs. The prediction errors incurred by the MMNL models are significantly larger.

Interestingly, when the MMNL(5) model is posited as ground truth, the MMNL benchmarks outperform the quasi-convex model in certain settings. When the scale parameter σ that controls

Table 3 Normalized mean squared errors (MSEs) of the fitted choice models in different ground truth settings.

Ground truth	Quasi-convex	MNL	MMNL(2)	MMNL(3)
MMNL(5), $\sigma = 0$	0.056	**0.001**	0.001	0.001
MMNL(5), $\sigma = 1$	0.059	0.005	**0.002**	0.004
MMNL(5), $\sigma = 10$	**0.169**	0.207	0.204	0.197
MMNL(5), $\sigma = 20$	**0.197**	0.224	0.217	0.216
Intervals model	**0.003**	0.286	0.229	0.225

Recall that σ is the scale of the log-normal distribution generating the weights of the MMNL ground truth model.

preference heterogeneity across customer segments grows, the quasi-convex model provides more accurate out-of-sample predictions than the fitted MNL, MMNL(2) and MMNL(3) instances. For example, for $\sigma = 10$, the MSE is smaller by a factor of 14%. As such, the quasi-convex model has an excellent fit to the data when the underlying preferences of the population are highly heterogeneous.

Experiments on purchase panel data. We had access to purchase panel data describing daily transactions made by hundreds of thousands of consumers across several retailers for three product categories over 2-5 months, namely *Bath Tissue*, *Shampoo and Conditioners*, and *Dog Food and Treats*. Transactions are aggregated at the brand level. Each assortment corresponds to the combination of a retail chain vendor and a US state. Specifically, each assortment is the union of all brands with at least one transaction during the time horizon. Having specified the assortments $\mathcal{A}_1, \ldots, \mathcal{A}_\tau$, the purchase probabilities p_{it} are obtained by computing the relative market shares of products according to the observed transactions (the no-purchase option is not observable). Since the purchase data can be sparse for certain brands and locations, we restrict attention to the universe of brands with at least 400 transactions recorded during the period considered. Consequently, the *Bath Tissue*, *Shampoo and Conditioners* and *Dog Food and Treats* data sets are formed by 17, 38 and 51 products, respectively, and comprised of 246, 171 and 220 assortments, respectively.

We implement a stratified 2-fold cross-validation, to ensure that the prediction labels (purchased brands) are approximately equally represented in the training and test sets. To complement the out-of-sample MSE metric, we also measure the mean absolute error (MAE), defined as follows:

$$\text{MAE} = \sum_{t \in \mathcal{OS}} \frac{1}{|\mathcal{OS}|} \cdot \sum_{i \in \mathcal{A}_t} |\hat{p}_{it} - p_{it}| \ ,$$

where $\mathcal{OS}$ is the collection of hold-out assortments. In addition, we compute the Akaike Information Criterion (AIC). For any of fitted choice model, the AIC corresponds to the quantity $2 \cdot d - 2 \cdot \mathcal{L}$, where d is the number of parameters of the choice model, and $\mathcal{L}$ is the log-likelihood of the training data (Akaike 1998). This metric quantifies a tradeoff between the number of parameters of the model and the in-sample accuracy, and thus, this notion guards against the risk of overfitting despite using an in-sample measure of the goodness-of-fit.

Table 4 % Improvement of prediction accuracy achieved by the quasi-convex model against the benchmarks.

Data sets	MNL			MMNL(3)		
	MSE	MAE	AIC	MSE	MAE	AIC
Dog Food & Treats	24%	14%	2.3%	15%	7.8%	0.9%
Bath Tissue	20%	9.7%	1.3%	18%	6.0%	1.3%
Shampoo & Conditioners	21%	9.0%	1.4%	21%	7.0%	1.1%

Our numerical results are reported in Table 4 in the form of a percentage improvement of 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 absolute prediction errors for various additional metrics along with the running times of the estimation methods are reported in the online appendix (Tables EC.3 and EC.4). The quasi-convex model outperforms the parametric benchmark models in terms of out-of-sample prediction error in all cases. As might be expected, the accuracy gains are smaller against the fitted MMNL(3) model, but remains very significant in absolute terms. For example, on the MSE metric, the MSE is reduced by 18% in the Bath Tissue category against the MMNL(3) model. In summary, the quasi-convex model provides strong predictive power in data-rich environments.

8. Concluding Remarks

This work opens various perspectives for future research. A natural lead is to further investigate the interplay between behavioral screening rules and the computational tractability of the resulting assortment optimization problem. In particular, the analysis of conjunctive screening rules in more general settings would be relevant to practice. Moreover, our current modeling approach is based on the assumption that the distribution over preference lists is exogenous, like for the vast majority of random-utility maximization choice models. Hence, from a modeling perspective, it would be interesting to examine choice models whereby the customer preferences are formed endogenously as a function of the offered assortment.

Another important question is to investigate the identifiability of the quasi-convex model model introduced in this paper. Indeed, the existing estimation methods for nonparametric models do not come with provable recovery guarantees. Additionally, the number of products in online retailing and the computational requirements of online platforms require the development of highly-scalable algorithms. Our current algorithmic ideas need to be complemented for such large-scale applications. Lastly, when the number of choice alternatives is large, choice models are often featurized, i.e., expressed as a function of underlying product and customer features. Incorporating contextual features into nonparametric choice models is a fundamental challenge that conditions the adoption of these methods by practitioners.

Acknowledgments

We would like to thank Management Science's review team for a host of technical and editorial comments. In particular, we are grateful to an anonymous reviewer, who proposed an elegant way to derive Theorem 5, substituting a more involved proof that appeared in an early version of this paper.

References

Ailon, Nir, Moses Charikar, Alantha Newman. 2008. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)* **55**(5) 23.

Akaike, Hirotogu. 1998. Information theory and an extension of the maximum likelihood principle. *Selected papers of hirotugu akaike*. Springer, 199–213.

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. 2018. The approximability of assortment optimization under ranking preferences. *Operations Research* **66**(6) 1661–1669.

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* **63**(9) 3128–3145.

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.

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 multiproduct 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.

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.

Online Appendix

EC.1. Additional Proofs and Results

EC.1.1. Proof of Claim 1

We assume otherwise and prove a contradiction. Since 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. $\square$

EC.1.2. Weakly laminar consideration sets

To demonstrate the robustness of our algorithmic approach, we extend the analysis of Section 4.2 to a more general notion of laminar consideration sets, dubbed *weakly laminar*. Although this setting is not reducible to induced intervals (contrary to the standard laminar property, as shown by Lemma 3), we derive a polynomial running time guarantee for the corresponding dynamic programming formulation.

Definition EC.1. 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 the related literature, notably the above-mentioned *downward substitution model*, as well as the *out-tree* model proposed by Honhon et al. (2012).

Theorem EC.1. *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 EC.1. *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* $j \in T$.

The proof the above property is deferred to this end of this section. Consequently, we can upper bound $|\Phi\langle\mathcal{S}\rangle|$. Let (S,T) be a subproblem in the state space $\mathcal{S}$. By Lemma EC.1, we obtain

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

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

Proof of Lemma EC.1 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 since 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:

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

By convention, we set $a_1 := i$. We now prove by induction over r, $2 \leq r \leq l$, 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 $a_2 \geq i$. 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 $(j_2, a_3, \ldots, a_l, j_l)$ 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, product 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 $a_r \leq a_{r-1}$ 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 $a_r \leq a_{r-1}$, 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 $(j_1, a_2, j_2, \ldots, j_{r-2}, a_r, j_r, \ldots, j_l)$ 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 $v \in T$.

EC.1.3. Proof of Theorem 4

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

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

where the second equality follows from Definition 3, 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 $|\Phi\langle\mathcal{S}\rangle| \leq N \cdot K^d$. $\square$

EC.1.4. The marginalized dynamic program

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 (EC.1).

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.

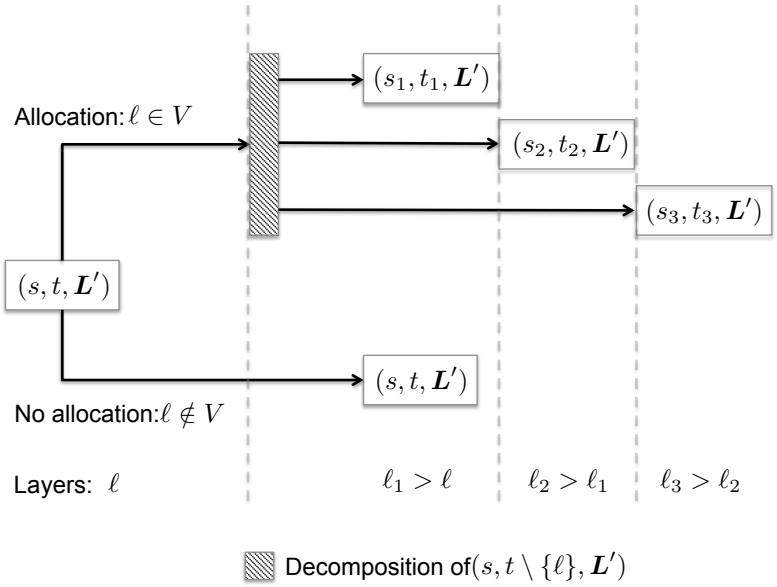


Figure EC.1 Recursive step of the procedure that constructs $\mathcal{D}(S, T, \boldsymbol{L})$.

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. Since 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:

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

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 1 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 $\{1, \ldots, \min(S) - 1\}$ 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 $\boldsymbol{L} = \boldsymbol{L}'$ since 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 $j \in T_1 \setminus T_2$. By noting $S_1 = S_2$, 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 $\mathcal{I}_2$, product i' has been deleted since it is preferred over i. Thus $i' \notin S_2$ and since $S_1 = S_2$, we obtain a contradiction: $i' \notin S_1$.

EC.1.5. 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 shows that, for a fixed parameter d, the state space complexity associated with unique-ranking $\Sigma = \{\sigma\}$ is preserved up to a polynomial factor under $\Sigma = B(\sigma, d)$. In particular, the polynomial running time guarantees established in Section 4 carry over to $\Sigma = B(\sigma, d)$. This result permits a parametric tradeoff between modeling power and tractability. In particular, the algorithm is expected to be efficient for small values of d, i.e., $d = O(\log N)$, and it is intractable for large values of d, i.e., $d >> \log N$.

THEOREM EC.2. *Let $\mathcal{S}(\mathcal{C}, \sigma)$ be the state space under a collection of consideration sets $\mathcal{C}$ and a unique ranking $\Sigma = \{\sigma\}$. 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 $2^{4d-2} \cdot |\mathcal{S}(\mathcal{C}, \sigma)|$.*

Proof We construct a function Ψ that maps any subproblem generated along the recursion to a tuple that lies in a set of size $2^{2d-2}h$. By showing that Ψ is injective, we obtain the upper bound on the size of the state space. In what follows, we assume that products are numbered according to the processing order σ.

Specifically, assuming that $(S, T, \boldsymbol{L}) \in \mathcal{S}$, we define $i = \min(S)$ 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 $[i-1]$ such that $i \in S_0$,
- $\boldsymbol{x} = \mathcal{A} \cap [i-2d+1, i-1]$.
- $\boldsymbol{y} = S \cap [i, i+2d-1]$.

We now argue that Ψ is injective. To this end, we begin by establishing two preliminary claims.

CLAIM EC.1. *For every $j \in T$, we have: $C_j \cap \mathcal{A} \cap [i-2d] = \emptyset$.*

Proof Suppose that there exists $\alpha \in \mathcal{A} \cap [i-2d] \cap C_j$. 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 $G_{\boldsymbol{L}}[S,T]$. $\square$

It immediately follows that we can express $\boldsymbol{L}$ as a function of $\boldsymbol{x}$.

Claim EC.2. *For every customer-type $j \in T$, $L_j = \min\{\sigma_j(\alpha) : \alpha \in \boldsymbol{x} \cap C_j\}$.*

The remainder of the proof seeks to establish the next lemma, which implies that Ψ is injective.

Lemma EC.2. *Let T_1 be the set of customer-types in T_0 such that $\boldsymbol{x} \cap C_j \neq \emptyset$ and $\boldsymbol{y} \cap C_j(L_j) = \emptyset$. Then, $G_{\boldsymbol{L}}[S,T]$ is the connected component of $G_{\boldsymbol{L}}[S_0 \cap (\boldsymbol{y} \cup [i+2d, N]), T_0 \setminus T_1]$ containing i.*

Proof We first establish that $G_{\boldsymbol{L}}[S,T] \subseteq G_{\boldsymbol{L}}[S_0, T_0]$. To this end, we argue that the residual graph obtained by the stocking decisions of $\mathcal{A}$ under the general dynamic program is a subgraph of the residual graph generated by the decisions of stocking $\mathcal{A} \cap [i-2d]$ under the unique-ranking dynamic program. To this end, we observe that all graph operations performed in the unique-ranking case are also performed by the general dynamic program:

- Customer-type node deletions: Observe that a customer-type node $j \in [K]$ is deleted under the unique-ranking dynamic program if $C_j \cap \mathcal{A} \cap [i-2d] \neq \emptyset$. Hence, by an argument identical to the proof of Claim EC.1, it is easy to verify that, if $C_j \cap \mathcal{A} \cap [i-2d] \neq \emptyset$, then customer-type j is also deleted under the general dynamic program by the stocking decisions $\mathcal{A}$.
- 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 and by (2), any product deleted in the unique-ranking case is also deleted under the general dynamic program.

Therefore, since $G_{\boldsymbol{L}}[S,T]$ is a connected component of the residual graph under the general dynamic program, it is also a connected subgraph of the residual graph obtained under the unique-ranking dynamic program. Since $S \cap S_0 \neq \emptyset$, it follows that $G_{\boldsymbol{L}}[S,T] \subseteq G[S_0, T_0]$, which immediately yields $G_{\boldsymbol{L}}[S,T] \subseteq G_{\boldsymbol{L}}[S_0, T_0]$.

We further refine this set inclusion. Since $S \subseteq S_0 \cap (\boldsymbol{y} \cup [i+2d, N])$ by definition of $\boldsymbol{y}$, it follows that that $G_{\boldsymbol{L}}[S,T] \subseteq G_{\boldsymbol{L}}[S_0 \cap (\boldsymbol{y} \cup [i+2d, N]), T_0]$. Finally, for every customer-types $j \in T_1$ and product $i' \in [i+2d, N]$, we have

$$\sigma_j(i') \geq i + d > L_j \ , \tag{EC.1}$$

where the last inequality holds by Claim EC.2 since $\boldsymbol{x} \cap C_j \neq \emptyset$. It follows that

$$S \cap C_j(L_j) \subseteq (\boldsymbol{y} \cup [i+2d, N]) \cap C_j(L_j) = (\boldsymbol{y} \cap C_j(L_j)) \cup ([i+2d, N] \cap C_j(L_j)) = \emptyset \ ,$$

where the last equality holds by (EC.1) and the definition of T_1. Thus, we infer that $j \notin T$. It follows that $G_{\boldsymbol{L}}[S,T] \subseteq G_{\boldsymbol{L}}[S_0 \cap (\boldsymbol{y} \cup [i+2d,N]), T_0 \setminus T_1]$.

Now, we will establish the reciprocal inclusion. Specifically, it is sufficient to show that none of the nodes in $G_{\boldsymbol{L}}[S_0 \cap (\boldsymbol{y} \cup [i+2d,N]), T_0 \setminus T_1]$ are deleted under the general dynamic program by the stocking decisions $\mathcal{A}$. By equation (2), when rankings are in $B(\sigma,d)$, it is clear that none of the products in $[i+2d,N]$ are deleted when stocking a product in $[i-1]$. Indeed, for every $i_1 \in [i-1]$, $i_2 \in [i+2d,N]$ and $j \in [K]$, $\sigma_j(i_1) \leq i+d-1 < i+2d-d \leq \sigma_j(i_2)$. It ensues that none of the product nodes $S \cup ([i+2d,N] \cap S_0) = S_0 \cap (\boldsymbol{y} \cup [i+2d,N])$ are deleted. To establish a similar claim for the customer-type nodes, we fix $j \in T_0 \setminus T_1$. The customer-type node j is deleted only if it is allocated to a product of $\mathcal{A}$. Given that $j \in T_0$, it follows that $\mathcal{A} \cap [i-2d] \cap C_j = \emptyset$, thus it is clear that customer-type j is not allocated by the stocking decisions in $\mathcal{A} \cap [i-2d]$. Now suppose ad absurdum that customer-type j was allocated to a product $i_1 \in \boldsymbol{x} \cap C_j$. It follows that there exists $i_2 \in \boldsymbol{y} \cap C_j(L_j)$ (otherwise, $\boldsymbol{y} \cap C_j(L_j) = \emptyset$, which yields the contradiction $j \in T_1$). Consequently, by equation (2), node i_2 was necessarily deleted from the graph following the allocation of product i_1 to customer-type j, which contradicts that $i_2 \in S$. $\square$

$\square$

EC.1.6. Degrees of freedom of the quasi-convex model

Claim EC.3. *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:

$$|\Sigma(N)| = \sum_{\ell=1}^{N} (N-\ell+1) \cdot 2^{\ell-1}$$

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

EC.1.7. Proof of Theorem 5

Throughout this section, we assume that products are numbered according to σ. We construct an injective mapping $\psi : \mathcal{S} \to [N] \times [N]$ from any connected subgraph generated along the recursion, i.e., belonging to the computational tree, onto pairs of products. Specifically, we have $\Psi(S, T, \boldsymbol{L}) = (a, i_{\min})$, where a is the last product stocked in the recursion before generating $(S, T, \boldsymbol{L})$, and $i_{\min} = \min(S)$ is the next product to be processed in S. Our proof proceeds from four structural claims stated below; the objective is to express the parameters of the subproblem $(S, T, \boldsymbol{L})$ as a function of $(a, i_{\min})$. To this end, we define $i_{\max} = \max(S)$, and for every $i \geq i_{\min}$, we let $T(i)$ denote the set $\{j \in [K] : C_j(L_j) \cap [i_{\min}, i] \neq \emptyset\}$.

Claim EC.4. *For every $j \in T$, if $a \in C_j$, then $L_j = \sigma_j(a)$, otherwise $L_j = N + 1$.*

Claim EC.5. $S = [i_{\min}, i_{\max}]$.

Claim EC.6. $T = T(i_{\max})$.

Claim EC.7. *$i_{\max}$ is the maximum index $i \in [i_{\min}, N]$ for which $G_{\boldsymbol{L}}[[i_{\min}, i], T(i)]$ is connected.*

By combining Claims EC.4, EC.5, EC.6 and EC.7, we infer that the subproblem $(S, T, \boldsymbol{L})$ is uniquely determined with respect to $(a, i_{\min})$. It follows that the mapping Ψ is injective. The proof of the above claims is provided in the remainder of this appendix.

Proof of Claim EC.4. Given a customer-type $j \in T$, suppose that $a \notin C_j$. Since C_j is an interval satisfying $C_j \cap [a+1, N] \neq \emptyset$, it follows that none of the products stocked before a are contained in the consideration set C_j. Therefore, according to the equation (3) governing the updates of $\boldsymbol{L}$, it follows that $L_j = N + 1$.

Now, suppose there exists $j \in T$ such that $a \in C_j$ and $L_j < \sigma_j(a)$. Then, by equation (3), we have necessarily stocked a product $i' < a$ such that $\sigma_j(i') < \sigma_j(a)$. Since the function σ_j is quasi-convex, for every $i \geq a$, it follows that $\sigma_j(i) \geq \sigma_j(a) > L_j$. Thus, $C_j(L_j) \cap S = \emptyset$, and there is no edge between $j \in T$ and any node in S in the subgraph $G_{\boldsymbol{L}}[S, T]$. This contradicts that $G_{\boldsymbol{L}}[S, T]$ is connected. Conversely, suppose there exists $j \in T$ such that $a \in C_j$ and $L_j > \sigma_j(a)$. Let $(S^a, T^a, \boldsymbol{L}^a)$ be the state corresponding to the decision of stocking product a (this state corresponds to a parent node of $(S, T, \boldsymbol{L})$ in the computational tree), and let $T(a)$ be the corresponding set of customer-types whose preference list contains a, i.e., $T(a) = \{j \in T^a : a \in C_j(L_j^a)\}$. Based on equation (3), it follows that $L_j = L_j^a$ and $j \notin T(a)$, which, by definition of $T(a)$, implies that $a \notin C_j(L_j^a)$. Hence, it follows $\sigma_j(a) > L_j^a = L_j$, which contradicts our hypothesis. $\square$

Proof of Claim EC.5. It is clear that $S \subseteq [i_{\min}, i_{\max}]$, and it remains to show that $[i_{\min}, i_{\max}] \subseteq S$. To this end, observe that, for every customer-type $j \in T$, the quasi-convex property implies that $C_j(L_j)$ is an interval. Thus, since $G_{\boldsymbol{L}}[S,T]$ is connected and $S \subseteq \cup_{j\in T} C_j(L_j)$, it is not difficult to verify that $\cup_{j\in T} C_j(L_j)$ is an interval satisfying $[i_{\min}, i_{\max}] \subseteq \cup_{j\in T} C_j(L_j)$.

Now suppose there exists $\alpha \in S \setminus [i_{\min}, i_{\max}]$, and let $j \in T$ be a customer-type such that $i \in C_j(L_j)$. Node α was necessarily deleted at an anterior step of the recursion, otherwise it would still be connected to the current subgraph via customer-type j. Thus, by equation (2), at an earlier state of the recursion $(S^\beta, T^\beta, \boldsymbol{L}^\beta)$, we have stocked a product $\beta \le a$ and allocated a customer-type $j^\beta \in [K]$ such that $\alpha \in I^\beta$, where $I^\beta = \{i \in C_{j^\beta}(L^\beta_{j^\beta}) : \sigma_{j^\beta}(i) \le \sigma_{j^\beta}(\beta)\}$. Since σ_{j^β} is quasi-convex, it follows that I^β is an interval that contains products α and β. As such, we have also deleted any remaining product contained in I^β. In particular, we have deleted product $i_{\min}$, since $\beta \le a \le i_{\min} \le \alpha$. We have obtained the desired contradiction. □

Proof of Claim EC.6. Fix a customer-type $j \in T$. Since $G_{\boldsymbol{L}}[S,T]$ is connected, it is clear that $C_j(L_j) \cap S \neq \emptyset$. By Claim EC.5, it immediately follows $C_j(L_j) \cap [i_{\min}, i_{\max}] \neq \emptyset$. On the other hand, suppose that there exists a customer-type $j \in [K]$ such that $C_j(L_j) \cap [i_{\min}, i_{\max}] \neq \emptyset$ and $j \notin T$. By Claim EC.5, this implies that $C_j(L_j) \cap S \neq \emptyset$. Hence, node j was necessarily deleted at an earlier step of the recursion, otherwise it would still be connected to the current subgraph. Thus, at an earlier step of the recursion $(S^\beta, T^\beta, \boldsymbol{L}^\beta)$, we stocked a product $\beta \le a$, which was allocated to customer-type j, and the truncation vector was updated accordingly: $L_j = \sigma_j(\beta)$. Consequently, by equation (2), we have deleted all products in $\{i \in C_j(L^\beta_j) : \sigma_j(i) \le \sigma_j(\beta)\} \subseteq C_j(L_j)$. In particular, this implies that $C_j(L_j) \cap S = \emptyset$, which yields a contradiction. We have just shown that $T = T(i_{\max})$. □

Proof of Claim EC.7. In order to establish the claim, it is sufficient to show that none of the products $i \in [i_{\min}, N]$ is deleted by the anterior stocking decisions. Suppose ad absurdum that there exists a product $i' \in [i_{\min}, N]$ that gets deleted by stocking a product $\beta \in [a]$ at an earlier state of the recursion $(S^\beta, T^\beta, \boldsymbol{L}^\beta)$. Then, by equation (2), there exists a customer-type $j \in [K]$ allocated to product β such that $\sigma_j(i') < \sigma_j(\beta)$ and $i', \beta \in C_j(L^\beta_j)$. Thus, since σ_j is quasi-convex and $i_{\min} \in [\beta, i']$, we infer that $\sigma_j(i_{\min}) < \sigma_j(\beta)$ and $i_{\min} \in C_j(L^\beta_j)$, which implies that $i_{\min}$ is also deleted. This yields a contradiction. □

EC.1.8. Proof of Theorem 6

The processing order can be chosen as an arbitrary permutation; to fix ideas, products are processed by order of increasing indices. We further assume that the model is augmented by a 'dummy' preference list, denoted by the index 0, and associated with the utility vector $\boldsymbol{u}_0 = \vec{0}$. Similarly, we introduce a 'dummy' product 0 represented by the vector features $\boldsymbol{x}_0 = \vec{0}$. In what follows, by abuse of language, a product may refer to the corresponding graph node, or its representation in the feature space.

Induction hypothesis. At a high-level, our proof shows 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 a subproblem $(S, T, \boldsymbol{L})$ is generated along the recursion. Then, there exists $(a, b, c, d) \in [N]^2 \times [K]^2$ such that $G_{\boldsymbol{L}}[S, T]$ is the connected component of $G_{\boldsymbol{L}}[S', T']$, where:

- The set of products S' is defined as follows:

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

- The set of customer-types T' is formed by all the utility vectors that lie in the cone $(\boldsymbol{u}_c, \boldsymbol{u}_d)$:

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

- The truncation vector $\boldsymbol{L}$ is defined as follows:

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

Before proving this structural result, we observe that the above-mentioned property implies that there exists an injective mapping from the DP states $(S, T, \boldsymbol{L}) \in \mathcal{S}$ onto the 5-tuples $(i, a, b, c, d) \in [N]^3 \times [K]^2$. As such, we conclude that $|\mathcal{S}| = O(N^3 \cdot K^2)$.

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, let $a = b = c = d = 0$ and $i = \min(S)$. In this case, we have $H(a, b, c, d) = \mathbb{R}^2$, $S' = [N]$ and $T' = [K]$. It is easy to verify that the induction hypothesis is satisfied.

Recursive step. Now, suppose that $(S, T, \boldsymbol{L})$ satisfies the induction hypothesis properties with respect to the parameters (a, b, c, d) and $(S', T', \boldsymbol{L})$. Next, the dynamic program examines product $i = \min(S)$.

If i is not stocked in the assortment, we only need to discard node i from the graph and compute the connected components of $G_{\boldsymbol{L}}[S \setminus \{i\}, T]$ to obtain the children subproblems. By our induction hypothesis, it follows that each child subproblem is a connected component of $G_{\boldsymbol{L}}[S' \setminus \{i\}, T'])$, and the induction hypothesis property is verified.

Now, suppose that product i is allocated to a subset of customer-types $V \subseteq T(i)$. We define $\alpha, \beta \in T$ as the indices corresponding to the extreme lines of the cone $conv\{\boldsymbol{u}_h : h \in V\}$, where $\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta$ are ordered in the anti-trigonometric order. Using our previous notation, every subproblem generated by the allocation V is a connected component of $(S^{(V)}, T^{(V)}, \boldsymbol{L}')$. In what follows, we

show that our induction hypothesis is satisfied for each child subproblem, either with respect to the parameters (a, i, c, α), or with respect to the parameters (i, b, β, d). Technically-speaking, this property is established via Claims EC.8, EC.9, EC.10, and EC.11, stated below. We refer the reader to Figure EC.2 where we illustrate the resulting decomposition of the residual subgraph, based on the polyhedra $H(a, i, c, \alpha)$ and $H(i, b, \beta, d)$.

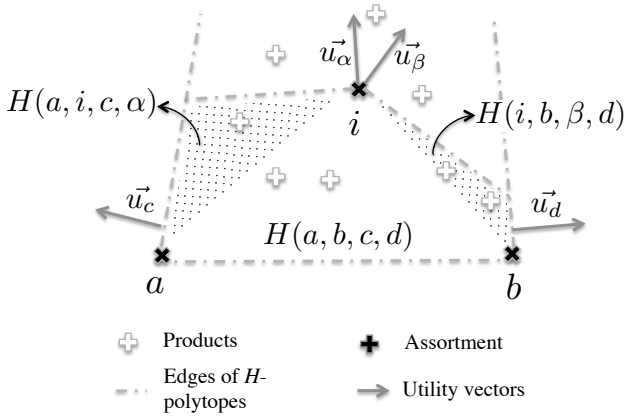


Figure EC.2 Recursive step: the allocation of product i to the cone $(\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$ gives rise to independent subproblems, either contained in the polyhedron $H(a, i, c, \alpha)$, or in the polyhedron $H(i, b, \beta, d)$.

Claim EC.8. *Without loss of generality,* $T^{(V)} = \{j \in [K] : \boldsymbol{u}_j \in (\boldsymbol{u}_c, \boldsymbol{u}_\alpha) \cup (\boldsymbol{u}_\beta, \boldsymbol{u}_d)\}$.

Proof Suppose that a customer-type $j \in T'$ has a utility vector $\boldsymbol{u}_j \in (\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$. Clearly, customer-type j may only pick a product generating a utility larger or equal to $\boldsymbol{x}_i \cdot \boldsymbol{u}^{(j)}$. Indeed, if $i \in C_j$, then customer-type j may only pick a product preferred over i. Conversely, if $i \notin C_j$, it may only choose a product generating a utility larger or equal to the customer's cut-off level. Since the customer-types α and β are both satisfied with product i, then every product i' that satisfies $\boldsymbol{x}_{i'} \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha$ or $\boldsymbol{x}_{i'} \cdot \boldsymbol{u}_\beta \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta$ has been deleted. The hypothesis $\boldsymbol{u}^{(j)} \in (\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)$ implies that every product i' for which $\boldsymbol{x}_{i'} \cdot \boldsymbol{u}^{(j)} \geq \boldsymbol{x}_i \cdot \boldsymbol{u}^{(j)}$ satisfies $\boldsymbol{x} \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha$ or $\boldsymbol{x} \cdot \boldsymbol{u}_\beta \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta$. Thus, customer-type j does not prefer any product of $S^{(V)}$ over product i, meaning that customer-type node j is disconnected from $S^{(V)}$. Consequently, without loss of generality, we may assume that $j \notin T^{(V)}$. $\square$

Claim EC.9. *$H(a, i, c, \alpha) \setminus \partial H(a, i, c, \alpha)$ and $H(i, b, \beta, d) \setminus \partial H(i, b, \beta, d)$ are disjoint.*

Proof Let $\boldsymbol{z}_1 = Rot\left(\frac{\pi}{2}, \boldsymbol{x}_i - \boldsymbol{x}_a\right)$ and $\boldsymbol{z}_2 = Rot\left(\frac{\pi}{2}, \boldsymbol{x}_b - \boldsymbol{x}_i\right)$. Since $\boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_a \cdot \boldsymbol{u}_\alpha$, $\boldsymbol{x}_a \cdot \boldsymbol{u}_c \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_c$ and $\boldsymbol{x}_i \cdot z_1 = \boldsymbol{x}_a \cdot \boldsymbol{z}_1$, then $\boldsymbol{z}_1 \in (\boldsymbol{u}_c, \boldsymbol{u}_\alpha)$. Using a similar argument, we infer that $\boldsymbol{z}_2 \in (\boldsymbol{u}_\beta, \boldsymbol{u}_d)$. By combining the latter two observations, since $\boldsymbol{u}_\beta \in (\boldsymbol{u}_\alpha, \boldsymbol{u}_d)$, we infer that $\boldsymbol{u}_\alpha \in (\boldsymbol{z}_1, \boldsymbol{z}_2)$, i.e., there exists $\lambda_1, \lambda_2 \geq 0$ such that $\boldsymbol{u}_\alpha = \lambda_1 \boldsymbol{z}_1 + \lambda_2 \boldsymbol{z}_2$. Then, by definition, if there exists a vector $\boldsymbol{x} \in (H(a, i, c, \alpha) \setminus \partial H(a, i, c, \alpha)) \cap (H(i, b, \beta, d) \setminus \partial H(i, b, \beta, d))$, then $\boldsymbol{x} \cdot \boldsymbol{z}_1 > \boldsymbol{x}_i \cdot \boldsymbol{z}_1$, $\boldsymbol{x} \cdot \boldsymbol{z}_2 > \boldsymbol{x}_i \cdot \boldsymbol{z}_2$

and $\boldsymbol{x} \cdot \boldsymbol{u}_\alpha < \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha$. However, by multiplying the first two inequalities by λ_1 and λ_2 respectively, and by summing them, we obtain

$$\boldsymbol{x} \cdot (\lambda_1 \boldsymbol{z}_1 + \lambda_2 \boldsymbol{z}_2) > \boldsymbol{x}_i \cdot (\lambda_1 \boldsymbol{z}_1 + \lambda_2 \boldsymbol{z}_2) = \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha \ .$$

This yields the desired contradiction. □

Claim EC.10. $S^{(V)} = S \cap ((H(a,i,c,\alpha) \setminus \partial H(a,i,c,\alpha)) \cup (H(i,b,\beta,d) \setminus \partial H(i,b,\beta,d)))$.

Proof By our induction hypothesis and Claim EC.8, it is sufficient to show that

$$S \cap ((H\,(a,i,c,\alpha) \setminus \partial H\,(a,i,c,\alpha)) \cup (H\,(i,b,\beta,d) \setminus \partial H\,(i,b,\beta,d))) \subseteq S^{(V)} \ .$$

Since product i is allocated to the cone $V = \{j \in [K] : \boldsymbol{u}_j \in (\boldsymbol{u}_\alpha, \boldsymbol{u}_\beta)\}$, we have $S^{(V)} = S \setminus \bar{H}$, where

$$\bar{H} = \bigcup_{j \in V} \{s \in [N] : \boldsymbol{x}_s \cdot \boldsymbol{u}_j \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_j\} \ .$$

By Farkas lemma, we have

$$\bar{H} = \{s \in [N] : \boldsymbol{x}_s \cdot \boldsymbol{u}_\alpha \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha\} \cup \{s \in [N] : \boldsymbol{x}_s \cdot \boldsymbol{u}_\beta \geq \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta\} \ ,$$

and thus, we have

$$S^{(V)} = \{s \in S : \boldsymbol{x}_s \cdot \boldsymbol{u}_\alpha < \boldsymbol{x}_i \cdot \boldsymbol{u}_\alpha\} \cap \{s \in S : \boldsymbol{x}_s \cdot \boldsymbol{u}_\beta < \boldsymbol{x}_i \cdot \boldsymbol{u}_\beta\} \ .$$

Consequently, it is not difficult to see that the linear constraints defining the vectors of $S^{(V)}$ are satisfied by $H(a,i,c,\alpha) \setminus \partial H(a,i,c,\alpha)$. Using a symmetrical reasoning, we obtain $H(i,b,\beta,d) \setminus \partial H(i,b,\beta,d) \subseteq S^{(V)}$. □

Claim EC.11. *For every $j \in T^{(V)}$, if $\boldsymbol{u}_j \in (\boldsymbol{u}_c, \boldsymbol{u}_\alpha)$, then $L'_j = \min(\sigma_j(i), \sigma_j(a))$ and $C_j(L'_j) \cap S^{(V)} \subseteq H(a,i,c,\alpha) \setminus \partial H(a,i,c,\alpha)$. If $\boldsymbol{u}_j \in (\boldsymbol{u}_\beta, \boldsymbol{u}_d)$, then $L'_j = \min(\sigma_j(i), \sigma_j(b))$ and $C_j(L'_j) \cap S^{(V)} \subseteq H(i,b,\beta,d) \setminus \partial H(i,b,\beta,d)$.*

Proof Fix a customer-type $j \in T^{(V)}$. By symmetry, the remainder of the proof focuses on the case $\boldsymbol{u}_j \in (\boldsymbol{u}_c, \boldsymbol{u}_\alpha)$. By the construction of $H(a,b,c,d)$, if $\boldsymbol{u}_j \in (\boldsymbol{u}_c, \boldsymbol{u}_\alpha)$, then customer-type j's most preferred product among $\{a,i,b\}$ is either a or i, meaning that $L'_j = \min(\sigma_j(i), \sigma_j(a))$. Next, we observe that since a and i are stocked in the assortment, the products contained in the truncated consideration set $C_j(L'_j)$ necessarily generate a utility greater or equal the quantities $\boldsymbol{x}_a \cdot \boldsymbol{u}_j$ and $\boldsymbol{x}_i \cdot \boldsymbol{u}_j$. In other words, the vectors associated with products $C_j(L'_j)$ lie in the affine half-space $\{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{x} \cdot \boldsymbol{u}_j > \bar{u}\}$, where $\bar{u} = \max\{\boldsymbol{x}_a \cdot \boldsymbol{u}_j, \boldsymbol{x}_i \cdot \boldsymbol{u}_j\}$. By Claim EC.10, the intersection of $S^{(V)}$ with the half-space $\{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{x} \cdot \boldsymbol{u}_j > \bar{u}\}$ is contained in $H(a,i,c,\alpha) \setminus \partial H(a,i,c,\alpha)$. Therefore, we infer that $C_j(L'_j) \cap S^{(V)}$ is contained in $H(a,i,c,\alpha) \setminus \partial H(a,i,c,\alpha)$. □

EC.2. Empirical Performance

EC.2.1. Additional details on the computational experiments

Computational benchmark. 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:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{N}\sum_{j=1}^{K} P_i \cdot \lambda_j \cdot x_{i,j} & & \\
\text{s.t.} \quad & x_{i,j} \le y_i & \forall (i,j) \in [N]\times[K] & \quad \text{(EC.2)}\\
& x_{i,j} + y_l \le 1 & \forall j \in [K], l \in C_j \text{ and } i \in \{x \in C_j : \sigma_j(l) < \sigma_j(i)\} & \quad \text{(EC.3)}\\
& \sum_{i \in C_j} x_{i,j} \le 1 & \forall j \in [K] & \quad \text{(EC.4)}\\
& x_{i,j}, y_i \in \{0,1\} \ , & &
\end{aligned}
$$

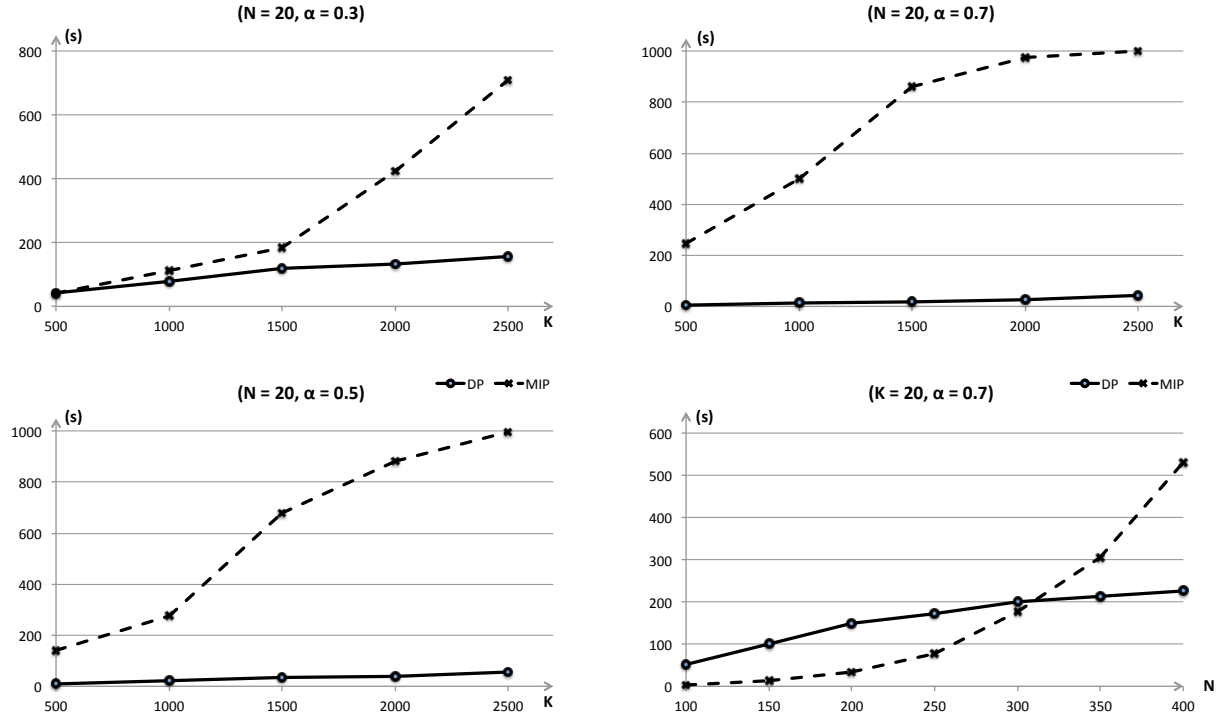
where the coupling constraints (EC.2) enforce that a customer may only pick a product made available in the assortment while the inequalities (EC.3) ensure that a given customer-type could only choose the highest rank product made available to him. Finally, the constraints (EC.4) mean that at most one product is assigned to each customer. The additional constraints (EC.4) 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.

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. 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 $O(N^3K)$.

EC.2.2. Additional tables and figures

Figure EC.3 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.

Table EC.1 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	–	$< 0.1\%$	$< 0.1\%$

.

Table EC.2 Summary statistics of the training data sets.

Data set	N	#	τ	Avg. $\lvert\mathcal{A}_t\rvert$	Max. $\lvert\mathcal{A}_t\rvert$
Dog Food & Treats	51	302557	220	19.4	39
Bath Tissue	17	262027	246	7.3	11
Shampoo & Conditioners	38	199438	171	23.7	35

designates the number of observations. Recall that τ denotes the number of distinct assortments $\mathcal{A}_t$.

Table EC.3 Running time of the estimation procedure (in seconds).

Data set	Quasi-convex	MNL	MMNL(3)
Dog Food & Treats	4380 s	172.5 s	7200 s
Bath Tissue	1210 s	6.33 s	900 s
Shampoo & Conditioners	2550 s	30.0 s	4960 s

We impose a running time limitation of 7200 s on each estimation procedure.

Table EC.4 Prediction errors achieved by the fitted choice models.

Data sets	Choice model	Accuracy metrics				
		MSE	MAE	χ^2	AIC	BIC
Dog Food & Treats	Quasi-convex	0.17	0.42	154	1363914	1370886
	MNL	0.22	0.50	194	1397034	1397429
	MMNL(3)	0.20	0.46	160	1377260	1378514
Bath Tissue	Quasi-convex	0.25	0.50	236	1001216	1005606
	MNL	0.31	0.55	301	1014380	1014770
	MMNL(3)	0.31	0.53	261	1014356	1015595
Shampoo & Conditioners	Quasi-convex	0.14	0.38	57	1012777	1019499
	MNL	0.18	0.42	72	1027823	1028204
	MMNL(3)	0.18	0.41	61	1024220	1025430

We define the χ^2 error as the quantity $\frac{(p-\hat{p})^2}{0.01+p}$ where p is observed choice probability and $\hat{p}$ is the predicted probability according to the fitted choice model. This error is summed over all the observations in the hold-out data set. Note that the quantity 0.01 is added to the denominator to avoid over-weighting the entries corresponding to very small choice probabilities. The BIC metric is defined as $2 \cdot d \cdot \ln(n) - 2 \cdot \mathcal{L}$, where d is the number of parameters of the fitted model, n is the number of observations, and $\mathcal{L}$ is the log-likelihood of the in-sample data with respect to the fitted choice model. Note that the metrics MSE, MAE and χ^2 are defined with respect to out-of-sample errors, while AIC and BIC are defined with respect to in-sample errors.

EC.2.3. Additional details on the estimation methods

In order to calibrate the choice models to training data, we leverage standard estimation methods developed in earlier literature. Specifically, the parametric choice models are estimated through standard maximum likelihood methods (McFadden 1973, Talluri and Van Ryzin 2006). We employ a column generation algorithm to construct the quasi-convex preference lists. The approach is in the same vein as the methodology proposed by van Ryzin and Vulcano (2014) or Bertsimas and Mišic (2015). However, contrary to arbitrary nonparametric choice models, we show that the column generation step can be solved in polynomial time under the quasi-convex model.

Additional notation. Recall that the training data takes the form of a collection of assortments $\{\mathcal{A}_1, \ldots, \mathcal{A}_\tau\}$, with corresponding purchase probability p_{it} of product i in the assortment $\mathcal{A}_t$, for each $i \in [N]$ and $t \in [\tau]$. Let $\mathcal{P}$ designate the probability vector obtained by flattening the matrix $(p_{it})_{i,t \in [N] \times [\tau]}$ in column-major order. When the no-purchase option is not observable, the product purchases probabilities are normalized and sum up to 1.

Estimation of MNL and MMNL. In order to estimate the MNL and MMNL parameters, we use standard maximum likelihood estimation (McFadden 1973, Talluri and Van Ryzin 2006). This estimation method is a standard approach used to calibrate discrete mixtures of MNL (Bierlaire 2003, Hess et al. 2007). The corresponding maximum log-likelihood estimation problem is implemented using the convex-programming 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 the optimization to a global optimum is not guaranteed in this setting. Thus, we use 10 random initializations of the solver, and we ultimately select the model parameters that achieve the best fit according to the log-likelihood cost.

Estimation of quasi-convex preferences. In order to calibrate the quasi-convex model with data, we employ the column generation ideas developed in the related literature (van Ryzin and Vulcano 2014, Bertsimas and Mišic 2015). To this end, let $\mathcal{L}$ be the collection of *all* quasi-convex preference lists for a given instance and let m be the number of distinct such lists (the dependency of m on N is explicitly stated by Claim EC.3 in Appendix EC.1.6). To ease the notation, we assume that the no-purchase option is captured by an alternative in $[N]$. Consequently, we introduce the observation tensor $\mathcal{O} = (\mathcal{O}_{t,i,j}) \in \{0,1\}^{[\tau]\times[N]\times[m]}$, where $\mathcal{O}_{t,i,j} = 1$ if the preference list $\mathcal{L}_j$ purchases product i in the assortment $\mathcal{A}_t$, and $\mathcal{O}_{t,i,j} = 0$ otherwise. In what follows, by abuse of notation, $\mathcal{O}$ will also designate the corresponding $\tau \cdot N$-by-m matrix (known as the *mode*-3 *unfolding*).

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

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

In the special case of the ℓ_1 norm, the latter problem can be cast a linear program with $O(t \cdot N)$ equality constraints through standard techniques. However, the number of variables of the resulting linear program remains exponential. Indeed, by Claim EC.3, the number of distinct quasi-convex preference lists m grows exponentially in the number of products. Hence, since the number of equality constraints is small, we resort to a column generation procedure. This procedure has notably been developed in the paper by Bertsimas and Mišic (2015) – we refer the reader to the latter work for a more comprehensive discussion.

The algorithm alternates between solving a *master* problem and a *column generation* subproblem. Specifically, at step $q \in \mathbb{N}$, given an incumbent (fixed) collection $\mathcal{L}_q \subseteq \mathcal{L}$ of quasi-convex preference lists, the master problem solves the ℓ_1-minimization program to find a distribution over $\mathcal{L}_q$ that best fits the data. Next, the column generation subproblem attempts to identify a new quasi-convex list $L \in \mathcal{L} \setminus \mathcal{L}_q$ with lowest reduced cost. While the master problem can be cast as a linear program, the column generation step is generally NP-hard. Indeed, this problem subsumes the widely-studied rank aggregation problem as a special case Ailon et al. (2008). Thus, prior literature has investigated heuristic procedures based on integer programming, local search ideas, and sampling-based methods.

In contrast, our next theorem shows that the column generation step can be solved in polynomial time under the quasi-convex model. Specifically, we are given the reduced cost matrix $(h_{t,i})_{t\in[\tau],i\in[N]}$. To wit, $h_{t,i}$ is the reduced cost associated with the equality constraint regarding the probability of

"purchasing product i in the assortment $\mathcal{A}_t$". For every list $L \in \mathcal{L}$, we let $I(t,i,L)$ be the binary function that indicates whether the preference list L picks product i in the assortment $\mathcal{A}_t$.

THEOREM EC.3. *The rank aggregation problem, that consists in finding the optimal quasi-convex list $L \in \mathcal{L} \setminus \mathcal{L}_q$ to minimize the reduced cost $\phi(L) = \sum_{t=1}^{\tau} \sum_{i=1}^{N} h_{t,i} \cdot I(t,i,L)$, can be solved in time $O(N^3)$.*

The proof of this theorem is provided in Appendix EC.2.4. We develop an efficient dynamic program to compute the quasi-convex preference list that minimizes the reduced cost.

Parameter tuning. For simplicity, we do not attempt to optimize the central permutation of the quasi-convex structure. The central permutation is picked uniformly at random. However, our estimation method fully exploits the flexibility of nonparametric choice models to tune all hyper-parameters based on the predictive task at hand and the metric ultimately used to evaluate the predictive accuracy. Specifically, to guard against the risk of over-fitting, the total number of preference lists is picked in the set $\{100, 200, 300, 400, 500, 600\}$ through a leave-one-out cross-validation on the training set. Similarly, the norm used in the final calibration step (ℓ_1, ℓ_2 or KL-divergence) is determined through cross-validation.

EC.2.4. Proof of Theorem EC.3

The optimization problem is formulated as a dynamic program. The state space is formed by the collection of 3-tuples (a,b,c) where $a,b,c \in [N]$. Let $c' = c + \max\{a,b\} - 1$, and, for every preference list L, let σ_L designate the corresponding ranking permutation. We define $L(a,b,c) \in \mathcal{L}$ as the optimal quasi-convex list $L \in \mathcal{L}$ (i.e., with lowest reduced cost $\phi(L)$), subject to the constraints that the consideration set is formed by the interval $[c,c']$ with $\sigma_L(c) = a$, $\sigma_L(c') = b$, and $\sigma_L\langle[c,c']\rangle = [1,b]$ if $a \geq b$, and $\sigma_L\langle[c,c']\rangle = [1,a]$ otherwise. Lastly, we introduce the value function $F(a,b,c)$, that computes the reduced cost attained by $L(a,b,c)$.

In what follows, let (a,b,c) designate a state of the recursion. To ease the exposition, suppose for now that $a - b \geq 2$. Since $L(a,b,c)$ is quasi-convex, it is not difficult to verify that the rank values associated with the interval of products $[c, c+a-b-1]$ are already determined: they form the interval $[b+1, a]$. Indeed, for every $i \in [c, c+a-b-1]$, we necessarily have $\sigma_{L(a,b,c)}(i) = a - (i-c)$, otherwise we would violate the quasi-convex property. Our next dynamic programming decision is to choose the product (within the interval $[c,c']$) whose rank value is exactly $b-1$. Due to the quasi-convex structure, there can be at most two options: either the leftmost unassigned product or the rightmost unassigned product in the consideration set. That is, we can assign $\sigma^{-1}_{L(a,b,c)}(b-1) = c+a-b$ or $\sigma^{-1}_{L(a,b,c)}(b-1) = c' - 1 = c+a-2$. Letting $c_1 = c+a-b$, $c_2 = c+a-2$, $\text{Ass}_1 \subseteq [\tau]$ be

the collection of indices $t \in [\tau]$ so that $\mathcal{A}_t \cap [c_1, c_2] = \emptyset$ while $\mathcal{A}_t \cap [c, c_1 - 1] \neq \emptyset$, and $\text{Ass}_2 \subseteq [\tau]$ be the collection of indices $t \in [\tau]$ so that $\mathcal{A}_t \cap [c, c_2 - 1] = \emptyset$ while $c_2 \in \mathcal{A}_t$, we obtain the recursion:

$$F(a,b,c) = \min \left\{ F(b-1, b, c_1) + \sum_{t \in \text{Ass}_1} h_{t, \max \mathcal{A}_t \cap [c, c_1 - 1]}, F(a, b-1, c) + \sum_{t \in \text{Ass}_2} h_{t,c_2} \right\} .$$

In the latter expression, the left-side terms of the min-expression compute the reduced cost associated with products $[c, c_1 - 1]$ when the rank value $b-1$ is assigned to the leftmost product c_1. The right-side terms of the min-expression compute the reduced cost associated with product c_2 when the rank value $b-1$ is assigned to the rightmost product $c_2 - 1$.

In the remaining cases, it can be shown that: (i) when $b - a \geq 2$, we formulate a symmetrical recursion, (ii) for the boundary cases $b = a+1$ or $a = b+1$, there exists a single quasi-convex preference list that satisfies the constraints, thus the dynamic programming value can be immediately computed. $\square$

EC.3. Other Extensions

EC.3.1. 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:

$$J(S,T) = \max\Bigg[P_i \cdot \sum_{j\in T(i)} \lambda_j + \max_{\boldsymbol{b}\in B(b-1,r(+))} \sum_{u=1}^{r(+)} J(S_u^+, T_u^+, b_u) \ , \tag{EC.5}$$

$$\max_{\boldsymbol{b}\in B(b,r(-))} \sum_{u=1}^{r(-)} J(S_u^-, T_u^-, b_u)\Bigg] \tag{EC.6}$$

Resource allocation problem. We observe that finding the optimal budget allocations in each max-expression (EC.6) and (EC.5) 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).

EC.3.2. General objective function

A close examination of the algorithm reveals that our results apply to a broader class of objective functions that we describe below. We consider *pay-off* functions $f: [N] \times [K] \to \mathbb{R}$, 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 total objective generated by assortment $\mathcal{A}$ is given by: $\sum_{j\in[K]} \lambda_j \cdot f(i(\mathcal{A},j), j)$. This generalization allows us to model social welfare maximization with customer-type-specific utility functions. Here, $f(i,j)$ is interpreted as the utility garnered by customer $j \in [k]$ when purchasing product $i \in [N]$. In addition, this general model captures the case where the arrival rate of each customer-type depends endogenously on her most preferred product in the offered assortment. For example, a customer may initially have a very restrictive consideration set, but in case none of her favorite products are offered, she relaxes some her requirements and samples a larger consideration set with some probability $p > 0$, or leaves with probability $1 - p$.