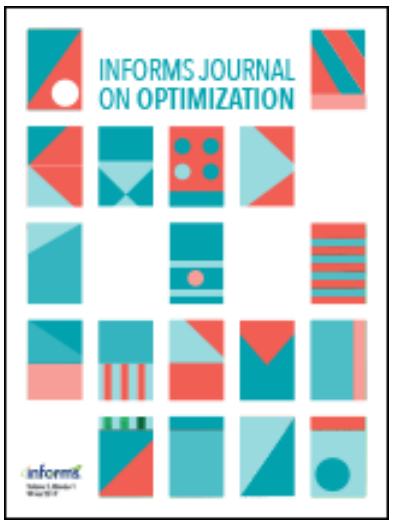


INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Partially Ranked Choice Model for Large-Scale Data-Driven Assortment Optimization

Sanjay Dominik Jena, Andrea Lodi, Hugo Palmer, Claudio Sole

To cite this article:
Sanjay Dominik Jena, Andrea Lodi, Hugo Palmer, Claudio Sole (2020) A Partially Ranked Choice Model for Large-Scale Data-Driven Assortment Optimization. INFORMS Journal on Optimization 2(4):297-319. https://doi.org/10.1287/ijoo.2019.0037

http://pubsonline.informs.org/journal/ijoo

INFORMS JOURNAL ON OPTIMIZATION
Vol. 2, No. 4, Fall 2020, pp. 297–319
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

A Partially Ranked Choice Model for Large-Scale Data-Driven Assortment Optimization

Sanjay Dominik Jena,[a,b] Andrea Lodi,[c,d] Hugo Palmer,[e] Claudio Sole[d]

[a] École des Sciences de la Gestion, Université du Québec à Montréal, Montréal, Quebec H2X 3X2, Canada; [b] Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT), Montreal, Quebec H3T 1J4, Canada; [c] Canada Excellence Research Chair in Data-Science for Real-time Decision-Making, Polytechnique Montréal, Montréal, Quebec H3T 1J4, Canada; [d] Polytechnique Montréal, Montréal, Quebec H3T 1J4, Canada; [e] BlaBlaCar, Paris 75002, France

Contact: jena.sanjay-dominik@uqam.ca, https://orcid.org/0000-0001-7650-6970 (SDJ); andrea.lodi@polymtl.ca, https://orcid.org/0000-0001-9269-633X (AL); hugo.palmer@blablacar.com, https://orcid.org/0000-0002-1836-6663 (HP); claudio.sole@polymtl.ca, https://orcid.org/0000-0002-2144-2137 (CS)

Received: September 22, 2017
Revised: January 23, 2019; December 20, 2019
Accepted: January 27, 2020
Published Online in Articles in Advance: September 22, 2020

https://doi.org/10.1287/ijoo.2019.0037

Abstract. The assortment of products carried by a store has a crucial impact on its success. However, finding the right mix of products to attract a large portion of the customers is a challenging task. Several mathematical models have been proposed to optimize assortments. Most of them are based on discrete choice models that represent the buying behavior of the customers. Among them, rank-based choice models have been acknowledged for representing well high-dimensional product substitution effects and, therefore, reflect customer preferences in a reasonably realistic manner. In this work, we extend the concept of (strictly) fully ranked choice models to models with partial ranking that additionally allow for indifference among subsets of products, that is, on which the customer does not have a strict preference. We show that partially ranked choice models are theoretically equivalent to fully ranked choice models. We then propose an embedded column-generation procedure to efficiently estimate partially ranked choice models from historical transaction and assortment data. The subproblems involved can be efficiently solved by using a growing preference tree that represents partially ranked preferences, enabling us to learn preferences and optimize assortments for thousands of products. Computational experiments on artificially generated data and a case study on real industrial retail data suggest that our proposed methods outperform existing algorithms in terms of scalability, prediction accuracy, and quality of the obtained assortments.

Funding: The work of S. D. Jena was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under [Grant 2017-05224]. H. Palmer and C. Sole were, respectively, research master and PhD student at Polytechnique Montréal in the Canada Excellence Research Chair "Data Science for Real-time Decision-Making." C. Sole has also been partially funded by Mitacs, whose contribution is gratefully acknowledged.
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoo.2019.0037.

Keywords: nonparametric discrete choice models • partially ranked preference sequences • estimation methods • column generation • assortment optimization

1. Introduction

Assortment planning denotes the process of identifying the set of products that should be offered to the customers. The planning problem is of paramount importance in operations and revenue management, since the choice of the assortment directly impacts the success of the business. However, from a managerial perspective, identifying the ideal assortment is a difficult challenge. Although offering more products to the customer may eventually increase the number of sold items (also referred to as conversion), it is well known that an assortment that is too large may jeopardize the total sales. Of course, limitations of space and capacity may naturally limit the number of products the client will be exposed to. However, one may still observe that the presence of a certain product may decrease the sales of another (which is known as product "cannibalization"). In the same way, the absence of a product may encourage the customer to substitute to another (potentially more profitable) product, illustrating the complex synergies among the products in an assortment.

The problem of finding the optimal assortment is crucial in several different domains. In particular, it is omnipresent in online advertisement, where it is necessary to decide for the limited number of advertisements that can be shown to a specific user profile in order to maximize the likelihood of conversion. In brick-and-mortar retail, assortment decisions are even more impactful, since the assortments are exposed to several different customer profiles and cannot be personalized to each customer type. Furthermore, changes in those

assortments can be longsome and costly, given that products will have to be physically removed from the store and inventory and typically be liquidated at a far lower price. Mathematical models to help find the "optimal" assortment are therefore becoming increasingly popular.

Defining a customer choice model that explains the market buying behavior sufficiently well is an essential step to optimize an assortment. Among the vast variety of choice models, rank-based choice models are rapidly gaining popularity. Rank-based choice models represent customer types via ranked preference lists on the available products. Those models hold a few important advantages. They can be easily interpreted by store managers and allow for insights regarding customer segmentation. Furthermore, they can be estimated in a purely data-driven manner without any assumptions on the market structure. In most of the application domains (such as online stores and even retail outlets), collecting transaction and assortment data has become standard, therefore making such data sufficiently available. Data-driven models that automatically make assortment recommendations based on historical data and with limited user input are likely to dominate the operational planning of the retail industry in the future. However, in practice, those models suffer from several challenges. On the computational side, rank-based choice models are hard to estimate. On the managerial side, even though preference lists (representing different customer types) provide store managers with certain insights, those lists typically contain unnecessarily large numbers of products (if not all), which limits the usefulness of those lists to understand customer segmentation.

1.1. Contributions

In this paper, we introduce a new representation for rank-based choice models that conceptually subsumes classical, fully ranked models (see, e.g., Farias et al. 2013, Bertsimas and Mišic 2016). Next to strictly ranked products, our choice model allows customers to be *indifferent* to a subset of the remaining products and therefore buy any of those products with equal probability if their strictly preferred products are unavailable. We prove that both fully and partially ranked choice models can represent the same buying behavior. However, several fully ranked preference lists are required to represent a partially ranked list. We propose an efficient method based on column generation to estimate this choice model by iteratively expanding a preference tree in which each node implicitly represents a partially ranked customer behavior. Although, in theory, finding new columns with negative reduced costs may require the solution of a mixed-integer linear programming problem (MIP), the proposed tree structure allowed us to find those columns trivially throughout all of our computational experiments. The estimation method is computationally attractive, handling instances with large number of products. It is also appealing to store managers, since the generated customer behaviors contain a significantly smaller list of ranked products that are necessary to explain the sales. An existing MIP formulation for assortment optimization is adapted to our partially ranked choice model, which allows the modeler to easily integrate side constraints such as capacities and requirements for different product categories (e.g., product subset or precedence constraints). We present extensive numerical results via simulation for choice model estimation and assortment optimization and compare with two recent benchmarks: the column-generation procedure of Bertsimas and Mišic (2016) and the k-deletion heuristic proposed by Jagabathula and Rusmevichientong (2019). Finally, we report on a case study with real-world data from a major North American fashion retail chain, provided by our industrial partner JDA Labs (2017).

1.2. Organization of the Paper

Section 2 reviews the literature that is relevant to our work. Section 3 introduces the new choice model and the procedure to efficiently identify relevant customer behaviors and the underlying distribution. Section 4 provides an MIP formulation to provide an optimized assortment based on our choice model. Section 5 reports on numerical experiments from a simulation study on synthetic data and for real-world data from the retail industry. We conclude in Section 6. Formal proofs, further theoretical developments, and additional computational results can be found in the appendix and the online supplement of this paper.

2. Relevant Literature

Most of the literature acknowledges that practically effective assortment optimization requires an appropriate choice model that represents the buying behavior of the customers when faced with an assortment. A vast variety of choice models has been applied in different domains such as transportation, marketing, and revenue management. We here focus on those that are most relevant to our work and refer the reader interested in broad overviews on assortment planning and choice models to surveys such as those of Mahajan and Van Ryzin (1999) and Kök et al. (2008). In the context of assortment optimization, two families of choice

models have found predominant popularity in the literature and have been successfully applied in the aforementioned domains.

2.1. Parametric Choice Models

The first family is that of (parametric) random utility maximization models. Among its most prominent members is the multinomial logit (MNL) choice model, which attributes a utility value to each of the available options. As in most of the choice models in revenue management, customers may also choose to abandon the buying process (e.g., if they are not willing to buy any of the available options), which is typically modeled as a dummy no-purchase option. Even though these models are analytically and computationally tractable, they have several shortfalls. In particular, the MNL assumes the independence of irrelevant alternatives (IIA) property (Arrow 1951), due to which substitution effects among products (such as cannibalization) cannot be captured. Nested logit models, pioneered by Ben-Akiva (1973) for the modeling of travel demand, capture certain cases of substitution, but are still subject to the IIA property within each nest. Further extensions, such as mixed multinomial logit (MMNL) models, overcome those shortfalls and can capture quite general customer behaviors. Unfortunately, these models are computationally challenging for practical assortment optimization, given that they do not only involve discrete variables, but are also typically nonlinear and nonconvex. Most importantly, parametric choice models generally rely on a good knowledge of the market structure and strongly depend on the application context (see, e.g., Jagabathula 2011), which makes them sensitive to issues of under- and overfitting.

2.2. Nonparametric Choice Models

The second family of choice models is nonparametric exponential models. Among them, rank-based models are quickly gaining popularity in the literature (see, e.g., Jagabathula 2011, Van Ryzin and Vulcano 2015, Vulcano and Van Ryzin 2017). Rank-based choice models assume that a customer behavior can be represented by a sorted preference list σ of the available options. The customer will then select the option that is ranked highest in her preference list and available in the assortment. Part of the model is a distribution over all possible preference sequences that specifies the probability that a random customer follows a specific buying behavior σ. Rank-based choice models have been acknowledged to offer manifold advantages. They implicitly capture high-dimensional substitution effects and, therefore, complex synergies among products. They do not make assumptions on the market structure and, therefore, do not involve the same risks of over- or underfitting, as may be the case for parametric models. Further, their distribution can, theoretically, be derived from historical data. Such a purely data-driven approach is attractive from the manager's perspective, as it requires little application-specific user input (often none at all) and is therefore easy to apply and to maintain. Identifying the relevant preference sequences may also give managers valuable insights about the customer segmentation.

Unfortunately, learning those choice models over the space of different preference sequences, which is factorially large in the number of products N, is a major challenge. Both identifying relevant customer behaviors and computing the underlying discrete *probability mass function* (*pmf*) that would explain the observed transactions are therefore computationally difficult. In this regard, Jagabathula (2011) and Farias et al. (2013) strongly advanced research by circumventing the need for searching in factorially large space, without the requirement of a preinformed market structure. These authors consider only those models that minimize the revenue for a given assortment, therefore enabling the use of the dual problem and resulting in the choice model for worst-case prediction. However, although the approach proposed by these authors may yield a good estimate of the worst-case revenue, it may not be ideal for managers, given that the assortment planning based on a worst-case choice model is likely to have a suboptimal performance on average.

Both Haensel and Koole (2011) and Vulcano and Van Ryzin (2017) focus on the case where the set of different customer types is limited and known beforehand. Instead of aggregating demand data over long periods, they divide the purchase horizon into smaller time periods. Haensel and Koole (2011) allow for multiple transactions per period. In contrast, Vulcano and Van Ryzin (2017) define at most one transaction per period. Both works explicitly handle data incompleteness that arises from the impossibility of distinguishing a period without a customer and a period with a nonpurchasing customer. The former authors estimate those customer arrivals in a preprocessing step, whereas the latter authors propose an expectation-maximization approach that integrates the estimation of the customer arrival rate. Although the assumption of knowing the customer sequences may seem overly restrictive in many settings, the proposed methods can be used in a more complex framework. In this spirit, Van Ryzin and Vulcano (2015) propose the *market discovery algorithm* that generates new customer sequences in a column-generation framework and estimate the corresponding

probability distribution with the aforementioned method (Vulcano and Van Ryzin 2017). The iterative framework assumes that a restricted master problem estimates the corresponding pmf for a subset of preference sequences, while new preference sequences are generated in a subproblem.

Bertsimas and Mišic (2016) follow a similar approach. The authors work on time-aggregated data and minimize the absolute ℓ_1 error between predicted and historical sales probabilities instead of maximizing the likelihood probability. The final choice model is then used in a novel mixed-integer programming model to provide an optimal assortment. More recently, this assortment optimization formulation has been further explored and improved (Bertsimas and Mišic 2019), to speed up the exact solution to the assortment optimization problem. Even though the proposed assortment optimization formulation scales fairly well, the computational complexity of identifying relevant customer behaviors is rather high. The entire process from choice model estimation to the optimized assortment is therefore limited to rather small numbers of products.

Our work is closely related to the aforementioned works (i.e., Van Ryzin and Vulcano 2015, Bertsimas and Mišic 2016), as it adapts the general column-generation framework. As in Bertsimas and Mišic (2016), we focus on the case of aggregated data. However, we do not make any assumptions about the loss function used to estimate the pmf other than convexity. Further, Van Ryzin and Vulcano (2015) use an integer programming heuristic to identify new reduced costs columns and emphasize that the computational burden increases significantly with the size of the problem, which is given by the number of transactions and is (in practice) directly related to the number of products. Their numerical experiments include not more than 15 products. Our paper explicitly focuses on the efficient estimation of the model when the number of products is large.

Finally, it is worth noting that Ho-Nguyen and Kilinc-Karzan (2017) provide a uniform view of the methods discussed above. Their method based on saddle point duality estimates rank-based choice models in the context of dynamic learning, when choice models are updated with new data over time. Even though the authors provide a theoretical convergence guarantee for their algorithms, they do not present any numerical experiments.

2.3. Partial Orders of Preferences

Most of the works on rank-based choice models assume that preference sequences must contain (almost) all products. However, it is known that only the first $\bar{n}+1$ ranked items are required to determine a customer choice (see, e.g., Honhon et al. 2012), when the size of the assortments does not exceed $N-\bar{n}$ items (where N is the total number of existing products and $\bar{n}$ is the number of items that are not part of the assortment). The *k-deletion* heuristic (Jagabathula and Rusmevichientong 2019) exploits this fact by enumerating all possible $O(N^{\bar{n}})$ sequences, which can only be deployed when $\bar{n}$ is small. In contrast, the number of strictly ranked products in our approach is determined dynamically, driven by the data, and can vary among customer behaviors. Further, throughout our computational experiments, our approach generally strictly ranked many fewer than $\bar{n}+1$ products.

Estimating fully (or almost fully) ranked preference sequences accurately is even more challenging when data on complete ranks are not available. Establishing partial orders among items has been of interest in both the machine learning and the operations management communities. Lebanon and Mao (2008) propose a general taxonomy to represent partial preference relationships and estimate the corresponding Mallows model. The model may consist of any sequence of single items or sets of items, where items within the same set are not preferred to each other. Although the preference information could essentially be represented by a (possibly quite large) set of pairwise preferences among items, the proposed partial rankings may allow for a much more compact representation. Jagabathula and Vulcano (2018) collect partially ranked information to construct a directed acyclic graph of preferences for each store customer. They then sample only those fully ranked sequences that are coherent with the preference graph. In this paper, we consider partially ranked sequences whose representation formally falls into the taxonomy of Lebanon and Mao (2008): a list of strictly ranked items followed by sets of items among which no strict preference is established. However, a key difference is that we enforce a uniform probability distribution among the items which are not strictly ranked, allowing us to obtain choice models that can be efficiently estimated and yield computationally tractable models to optimize the final assortment.

3. A Partially Ranked Choice Model

In this section, we will introduce a new representation for rank-based choice models that allows for strict preference on a subset of the products. This representation allows us to efficiently represent and construct relevant customer preferences using a preference tree. In Section 3.1, we will first introduce the new choice model and its representation as a tree. Then, in Section 3.2, column generation will be used to grow the decision tree and generate customer preferences.

3.0.1. Notation. We generally use boldfaced characters such as $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ to represent vectors and matrices. We use x_i to denote the ith element of vector $\boldsymbol{x}$.

3.1. The Choice Model

Rank-based choice models, such as the one introduced by Farias et al. (2013), assume that the market buying behavior is classified into different customer behaviors. Each customer behavior is represented by an ordered list that establishes a strict preference among all products. Preferred products are said to have *high ranks*, whereas less preferred products are said to have *low ranks*. The customer, following a specific behavior, buys exactly one product, which is the one that is *highest* ranked in her preference list and available in the presented assortment. Consider the set of products $\mathcal{N} = \{1, 2, \ldots, N\}$, and assume further that the customer always has the choice of selecting the no-purchase option 0, which would make her leave the store without any purchase. Following the notation used by Farias et al. (2013), we denote a customer behavior by σ and the rank of product i by $\sigma(i) \geq 0$. A fully ranked customer behavior can be defined as follows.

Definition 1 (Fully Ranked Customer Behavior)**.** A customer behavior σ with a fully ranked preference sequence may be written as a permutation of all $N + 1$ items in $\mathcal{N} \cup \{0\}$.

As an example, consider six available products. The fully ranked customer behavior $(3, 4, 1, 2, 5, 0, 6)$ indicates that the preferred product is 3, the second preferred product is 4, and so on. A customer with such a preference will buy the highest ranked product that is available in the assortment. Note that product 6 will never be bought, since it is ranked after the no-purchase option 0.

A rank-based choice model $(\boldsymbol{\sigma}, \boldsymbol{\lambda})$ is then defined as a set $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_K\}$ of customer behaviors, together with a pmf $\boldsymbol{\lambda} \in \mathbb{R}^K$ that represents the corresponding probabilities that a random customer entering the store follows the specific behavior.

Even though rank-based choice models have several desirable properties, fully ranked customer preferences are prone to some disadvantages. Their generation is computationally expensive, and from a management perspective, a fully ranked sequence allows for little insights regarding customer segmentation. This is due to the fact that most of the ranked products explain only marginal portions of the sales (see Observation 1 further below), preventing store managers from identifying those products that actually have high impact (and explain sales). Moreover, a general customer may not have a strict preference order in mind that is defined on all products. Instead, she may rather have a strict preference on a few products, but if none of those is available, she would choose any product from a subset of the available products with similar characteristics. As an example, consider a customer looking for a specific sport shoes model. If this model is not available in the exposed assortment, the customer may choose any other sport shoes model that has similar characteristics and is available in the assortment. Clearly, only a subset of the products available in the assortment complies with these characteristics.

3.1.1. Partially Ranked Customer Behaviors. The choice model proposed here assumes that customer preference lists do not necessarily have to impose a strict order on all products (which may be on the order of hundreds or thousands). Instead, a customer may have a strict preference on a subset of those products, $P(\sigma)$, for example, 3, 4, and 1. If those products are absent in the assortment, the customer may buy any similar and available product, for example, 2, 5, and 6. We may represent such a choice behavior as $\sigma = (P(\sigma), I(\sigma)) = (3, 4, 1, \{2, 5, 6\}, 0)$, where $P(\sigma) = (3, 4, 1) \subseteq \mathcal{N} \cup \{0\}$ is a strictly ranked list of preferred products and $I(\sigma) = \{2, 5, 6\} \subseteq \mathcal{N} \cup \{0\} \backslash P(\sigma)$ is a subset of *indifferent* products that will be chosen with uniform probability. There may be more than six products, but assuming that product 0 is either in $P(\sigma)$, in $I(\sigma)$, or after $I(\sigma)$, those products will never be selected.

Definition 2 (Simple Partially Ranked Customer Behavior)**.** A simple customer behavior σ contains a strictly ranked preference list $P(\sigma)$ and an indifference set $I(\sigma)$, where $P(\sigma) \subseteq \mathcal{N} \cup \{0\}$ and $I(\sigma) \subseteq \mathcal{N} \cup \{0\} \backslash P(\sigma)$ are mutually exclusive subsets of $\mathcal{N} \cup \{0\}$. Given an assortment S, a customer following behavior $\sigma = (P(\sigma), I(\sigma))$ will select the product that is ranked highest in $P(\sigma)$ and available in S. If $P(\sigma) \cap S = \emptyset$ but $I(\sigma) \cap S \neq \emptyset$, then the customer will select any of the products in $I(\sigma) \cap S$ (which may include the no-purchase option 0) with uniform probability. If, instead, $I(\sigma) \cap S = \emptyset$, the customer does not purchase any item.

Although fully ranked customer preferences require a hierarchy among all products such that $\sigma(i) < \sigma(j)$ whenever product i is preferred to j, partially ranked choice models also allow for relations of the form $\sigma(i) = \sigma(j)$, indicating that products i and j are equally preferred and therefore part of the same indifference set. In the example above, the product ranks are as follows: $\sigma(3) = 0$, $\sigma(4) = 1$, $\sigma(1) = 2$, $\sigma(2) = \sigma(5) = \sigma(6) = 3$,

and $\sigma(0) = 4$. Again, note that it is not required to list all N products in $P(\sigma)$ or $I(\sigma)$, meaning that $P(\sigma) \cup I(\sigma) \subseteq \mathcal{N} \cup \{0\}$ and the inclusion can be strict. Each product from $\mathcal{N}$ can be part of either the strictly ranked list or the indifference set, but not in both. Even though Definition 2 assumes that the no-purchase option 0 is always available, all developments made in this paper also apply to the case where 0 is unavailable, assuming that customers always purchase a product. The position of 0 also has implications on possible alternative notations. Specifically, a partially ranked behavior $\sigma = (P(\sigma), I(\sigma))$ can be equivalently written as $(P(\sigma))$ if $0 \in P(\sigma)$ and as $(P(\sigma), I(\sigma), 0)$ if $0 \notin P(\sigma) \cup I(\sigma)$.

When $P(\sigma)$ and $I(\sigma)$ do not include all products of $\mathcal{N}$, our notion of a partially ranked customer behavior becomes similar to *consideration sets* (see, e.g., Aouad et al. 2015, Jagabathula and Vulcano 2018). However, we further distinguish products in those consideration sets into strictly ranked products and indifferent products and impose a uniform probability distribution on the latter. Also note that a simple partially ranked behavior corresponds to the special case $\Xi_{1,\ldots,1,k}\pi$ within the taxonomy of Lebanon and Mao (2008): a sequence of strictly ordered items followed by k items among which no strict preference is imposed. However, the simple partially ranked behavior further assumes a uniform probability distribution among the k items.

Before elaborating on the equivalence between fully and partially ranked choice models, we note that the we may further generalize the simple partially ranked customer behavior to a more complex one, using q alternating lists of preferred products $P^\ell(\sigma)$ $(\ell = 1, \ldots, q)$ and indifference sets $I^\ell(\sigma)$. This more general behavior type cannot be fit into the taxonomy of Lebanon and Mao (2008). Given that the methodology proposed in this paper is based on simple partially ranked behaviors, we refer the reader interested in the more general behavior type to Appendix A.1.

3.1.2. Equivalence Between Representations. Even though the partially ranked choice model seems to be more general than fully ranked choice models, the underlying preference behaviors are essentially equivalent. Consider a simple partially ranked customer behavior $(3, \{2, 5, 6\}, 0)$. The same choice model can be represented by a set of fully ranked preferences with one complete (fully ranked) list for each of the permutations of the products in the indifference set: $(3, 2, 5, 6, 0)$, $(3, 2, 6, 5, 0)$, $(3, 5, 2, 6, 0)$, $(3, 5, 6, 2, 0)$, $(3, 6, 2, 5, 0)$, and $(3, 6, 5, 2, 0)$. This transformation holds for any simple partially ranked preference list.

Lemma 1. *Consider a partially ranked preference list $\sigma_p = (P(\sigma), I(\sigma))$ (see Definition 2), occurring with probability λ_p. We can generate from σ_p a set with $|I(\sigma_p)|!$ fully ranked preference lists and define the corresponding probabilities such that, given the same assortment S, the final probability that product i is bought is the same in both cases.*

A formal proof of Lemma 1 can be found in Appendix A.4. We note here that, in practice, it may not be necessary to enumerate all $|I(\sigma_p)|!$ fully ranked preference lists to represent choice probabilities that are equivalent to σ_p. We elaborate more on this topic in Appendix 1.1 in the online supplement. Based on Lemma 1, we now show that both the partially ranked choice model and the fully ranked choice model can represent equivalent buying behaviors.

Theorem 1 (Equivalence Between Fully Ranked and Partially Ranked Choice Models)**.** *A choice model $(\boldsymbol{\sigma}^C, \boldsymbol{\lambda}^C)$ based on fully ranked customer behaviors (see Definition 1) can be represented by a choice model $(\boldsymbol{\sigma}^P, \boldsymbol{\lambda}^P)$ that contains only partially ranked customer behaviors (see Definition 2). Further, any choice model $(\boldsymbol{\sigma}^P, \boldsymbol{\lambda}^P)$ that contains only partially ranked customer behaviors can be transformed into an equivalent choice model $(\boldsymbol{\sigma}^C, \boldsymbol{\lambda}^C)$ exclusively composed of fully ranked customer behaviors.*

We refer to Appendix A.5 for a formal proof of Theorem 1, which transforms a partially ranked sequence into a set of fully ranked sequences which is factorially large in the size of the indifference set. This transformation only serves to prove equivalency, and one may find tighter upper bounds for the number of required fully ranked sequences (see online supplement: Appendix 1.1).

Theorem 1 suggests that fully and partially ranked choice models can essentially reflect the same set of customer behaviors, but the latter will tend to have a sparser representation. Choice models with full ranks can reflect preference indifference on products, but it will require several fully ranked lists to represent an equivalent choice model. Further, explicitly ranking all products may be unnecessary, since low-ranked products tend to have little impact in explaining sales, that is, the proportion of the overall transaction predicted by them tends to be small. Quantifying the impact of a low-ranked product may be difficult in practice, given that assortments are typically composed by strategically chosen items. However, for the purpose of illustration, we may quantify the impact of low-ranked products on an *average assortment*, as stated below.

Observation 1 (Impact of Low-Ranked Products in Explaining Sales)**.** *The impact of low-ranked products in explaining the sales transactions can be negligibly small, both statistically and in practice. Consider a nonempty assortment S, let $r = |S|/N \in]0,1]$ be the assortment density of S, and assume that the probability that a certain product is part of S is uniform (i.e., it is equal to $|S|/N$). Then, the impact of a product decreases exponentially fast with its rank. Further, the greater the ratio r, the smaller the importance of low-ranked products.*

A verification of Observation 1 can be found in Appendix A.2. With a ratio of $r = 0.1$, the probability that none of the strictly ranked products is part of the assortment is, on average, about 3.87% for rank $k = 10$, but only about 0.05% for $k = 50$. With higher ratios, lower ranks quickly become insignificant. For example, with $r = 0.5$, the probability for $k = 10$ is as low as 0.05%. Although the above examples assume a uniform popularity in the customer priorities and in the assortments, in practice, it is likely that popular products are both highly ranked and are part of the assortment. We therefore argue that, in practice, the above stated probability tends to be a pessimistic (upper) bound on the importance of low-ranked products. In other words, we expect that the probability that low-ranked products are relevant decreases even more in practice.

We now introduce a special case of the simple partially ranked choice model, where the indifference set contains all products that are not strictly ranked, that is, $I(\sigma) = \mathcal{N} \cup \{0\} \backslash P(\sigma)$.

Definition 3 (Partially Ranked Customer Behavior with Complementary Indifference Set)**.** A partially ranked customer behavior σ with complementary indifference set is defined as a simple partially ranked behavior (see Definition 2) and imposing that $I(\sigma) = \mathcal{N} \cup \{0\} \backslash P(\sigma)$.

Although it may be possible to use the more general partially ranked choice model (see Appendix A.1) in the proposed methodological developments, we will restrict our attention from now on to the simplified case as in Definition 3. Even though using indifference sets that are composed of all nonranked products may seem unrealistic in practice, several reasons favor their use. First, such models are theoretically as sound as fully ranked choice models, since their behaviors can be transformed into fully ranked ones (see Theorem 1). Second, they implicitly define the indifference sets and, therefore, avoid the associated risks of overfitting. Third, they provide a clean data structure that can more easily be explored in an estimation method. Finally, such indifference sets also allow us to develop an intuition of their contribution on explaining the overall sales. Their *explanatory power* is analytically computed in the following for the illustrative case of an *average assortment*.

Observation 2 (Explanatory Power of the Indifference Set)**.** *Consider a consumer behavior $\sigma_k = (P(\sigma_k), I(\sigma_k))$ with estimated market probability λ_k and an assortment S selected uniformly at random with assortment density r. The expected explanatory power of σ_k, that is, its contribution to the explanation of the overall sales of product $i \notin P(\sigma_k)$ in S, amounts to $(1-r)^{|P(\sigma_k)|} \cdot \frac{\lambda_k}{|I(\sigma_k)|}$.*

A verification of Observation 2 is given in Appendix A.3. As a consequence, one can easily verify the average explanatory impact on products that are not strictly ranked in a partially ranked consumer behavior. For example, if $N = 100$, $|S| = 50$ (thus, $r = 0.5$), five products are strictly ranked (i.e., $|P(\sigma_k)| = 5$, $|I(\sigma_k)| = N - |P(\sigma_k)| = 95$), and $\lambda_k = 0.05$, the behavior σ_k explains only 0.0016% $(= (1-0.5)^5 \cdot \frac{0.05}{95})$ of the sales of any product $i \in I(\sigma_k)$. The impact of using such an indifference set on explaining the overall sales, for this example, amounts to 3.12% $(= (1-0.5)^5)$. Therefore, exposing the decision maker to such a concise list of five products is sufficient in practice, given that these five products explain 96.78% of sales caused by this consumer type.

3.2. Learning Consumer Preferences

Before we can optimize future assortments, we need to estimate the probability $\mathbb{P}(i|S)$ that a product i is sold given that a random customer is exposed to assortment $\mathcal{S}$. We may compute this probability once our choice model $(\boldsymbol{\sigma}, \boldsymbol{\lambda})$ is estimated. Given that there is a factorial number of different customer behaviors, a major challenge is to identify the set $\boldsymbol{\sigma}$ that is relevant for explaining the sales, as well as their corresponding probabilities $\boldsymbol{\lambda}$. In this section, we focus on how to efficiently learn those parameters that are consistent with the observed sales.

In line with the works of Farias et al. (2013) and Bertsimas and Mišic (2016), we assume that historical data are available in aggregated form,[1] including a total of M assortments, given by set $\mathcal{M} = \{S_1, S_2, \ldots, S_M\}$, as well as sales transaction data for each of them. Those sales are given in a vector $\boldsymbol{v} \in \mathbb{R}^{(N+1)\cdot M}$ consisting of all pairs (i,m), with $i \in \mathcal{N} \cup \{0\}$, $m \in \mathcal{M}$, which represent the customers that have chosen option i when being presented assortment S_m. Note that it is assumed that such sales data also includes the no-purchase option 0, for example, if it is accurately collected by the store or estimated by the store manager.

Our proposed estimation method fits into the general column-generation approach from Van Ryzin and Vulcano (2015). Relevant customer behaviors are generated in the subproblems, whereas the probabilities for a given set σ of customer behaviors are estimated in the master problem. In particular, the latter consists in estimating probabilities λ such that the choice model (σ, λ) is consistent with sales probabilities v. To this end, it has become quite common (see, e.g., Farias et al. 2013, Bertsimas and Mišic 2016) to embed the product choice of each customer type within a matrix A, which has one column for each customer preference sequence and one row for each product and assortment. We may compute this matrix $A \in \mathbb{R}^{((N+1)\cdot M)\times K}$ such that an entry $A^k_{(i,m)}$ is set to 1 if customer k would choose product i from assortment $S_m \cup \{0\}$. As a consequence, $\forall(k,m) : \sum_i A^k_{(i,m)} = 1$. Note that this definition of the matrix A differs from the one used by Van Ryzin and Vulcano (2015), who work on disaggregated data and therefore define one row per period (and at most one transaction per period).

The estimation of probabilities can be performed based on several criteria. Instead of estimating the choice model that results in the highest worst-case revenue (Farias et al. 2013), we follow two recent approaches that either maximize the likelihood probability (Van Ryzin and Vulcano 2015) or minimize the estimation error (Bertsimas and Mišic 2016). Although we will be exploring the two objectives in our computational experiments, we will here emphasize the model development for the latter: minimizing the ℓ_1 error between historical sales observations v and sales predictions $A\lambda = \sum_k A^k_{(i,m)}\lambda_k$ (i.e., the probability that a random customer chooses option i from assortment S_m). We therefore define the training error as

$$\epsilon^{Tr} = \sum_{(i,m)} |A\lambda - v|_{i,m}.$$

Consider a given set of potentially relevant customer behaviors $\{\sigma^1, \sigma^2, \ldots, \sigma^K\}$. One may use a simple linear program to find the corresponding probability distribution $(\lambda_1, \lambda_2, \ldots, \lambda_K)$ that results in the smallest error as follows:

$$\min_{\lambda, \epsilon^+, \epsilon^-} \quad \mathbf{1}^T\epsilon^+ + \mathbf{1}^T\epsilon^- \tag{1}$$

$$\text{s.t.} \quad A\lambda + \epsilon^+ - \epsilon^- = v, \tag{2}$$

$$\mathbf{1}^T\lambda = 1, \tag{3}$$

$$\lambda, \epsilon^+, \epsilon^- \geq 0. \tag{4}$$

Setting the Choice Matrix A. Bertsimas and Mišic (2016) propose to set the choice matrix A for fully ranked customer behaviors such that $A\lambda = v$ by using entry 1 if customer k would choose product i from $S_m \cup 0$. This notion does not hold in the context of our more general representation, which may include indifference sets. For reasons of simplicity and without loss of generality, let us consider the slightly simpler case $(P(\sigma), I(\sigma), 0)$ in which the customer behavior σ consists only of a single list $P(\sigma)$ of preferred and strictly ranked products, followed by an indifference set $I(\sigma)$. We assume that the indifference set $I(\sigma)$ (in the more general case, we have $I^1(\sigma), \ldots, I^q(\sigma)$) is externally given, for instance, by a marketing department. Alternatively, it can be learned or estimated in a previous step by identifying products with similar characteristics. We may then define the ranking entries of a customer behavior σ as follows:

$$\sigma(i) = \begin{cases} \text{rank of preference of } i & \text{if } i \in P(\sigma), \\ |P(\sigma)| & \text{if } i \in I(\sigma), \\ +\infty & \text{otherwise.} \end{cases}$$

The preferred products are ranked from 0 to $|P(\sigma)| - 1$, whereas all products in the indifference set have equivalent rank $|P(\sigma)|$. In the general case with several indifference sets, those ranks will be computed as the rank of the previous preferred product plus 1. The rank of a product that is neither in a $P(\sigma)$ nor in $I(\sigma)$ is set to $+\infty$.

Setting the choice matrix A for a partially ranked choice model has to respect $\forall(k,m) : \sum_i A^k_{i,m} = 1$, which we can achieve by a uniform distribution on the indifference set:

$$A^k_{i,m} = \begin{cases} 1 & \text{if } i \in S_m \text{ and } \forall j \in S_m \backslash \{i\} : \sigma^k(i) < \sigma^k(j), \\ \frac{1}{|I(\sigma) \cap \mathcal{G}_m|} & \text{if } i \in S_m : \text{ and } \forall j \in S_m, \sigma^k(j) = |P(\sigma)| \text{ or } \sigma^k(j) = +\infty, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

In words, customer k chooses item i from assortment S_m with a probability of 1 if i is available in m and is the most preferred among all available items. The customer chooses i with probability $1/|I(\sigma) \cap S_m|$, that is, with uniform probability among all items that are both in the indifference set and in the assortment, if none of the strictly ranked items is available. All other items are not selected: either because they are not in the assortment or because they are neither strictly ranked nor in the indifference set. Given the definitions above, once a subset σ of customer preferences (consistent with Definition 4) is identified, the corresponding choice matrix $\boldsymbol{A}$ can be efficiently computed.

Learning Consumer Behaviors Based on a Preference Tree. As noted by Bertsimas and Mišic (2016), the linear program (1)–(4) is not tractable, given the factorial number of customer behaviors here considered, causing $\boldsymbol{A}$ and $\boldsymbol{\lambda}$ to be exponentially large. The authors therefore propose a column-generation algorithm, which initializes problem (1)–(4) as the master problem with a small subset of promising columns (i.e., preference sequences). The algorithm then iteratively identifies possibly relevant columns and adds them to the master problem. Let $\boldsymbol{\alpha}$ and ν be the dual values of constraints (2) and (3) after solving problem (1)–(4). To find columns with minimal reduced cost, the authors further propose the following mixed-integer program:

$$\min_{z,a} \quad -\boldsymbol{\alpha}^T \boldsymbol{a} - \nu \tag{6}$$

$$\text{s.t.} \quad a_{i,m} \le z_{ij} \qquad \forall m \in \{1,\dots,M\},\ i,j \in S_m \cup \{0\},\ i \ne j, \tag{7}$$

$$z_{ij} + z_{ji} = 1 \qquad \forall i,j \in \{0,1,\dots,N\},\ i \ne j, \tag{8}$$

$$z_{ij} + z_{j\ell} - 1 \le z_{i\ell} \qquad \forall i,j,\ell \in \{0,1,\dots,N\}\ i \ne j,\ i \ne \ell,\ j \ne \ell, \tag{9}$$

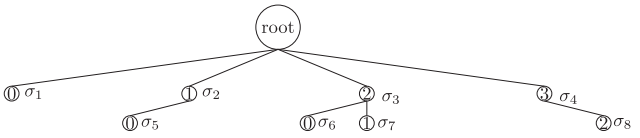
$$\boldsymbol{z} \in \{0,1\},\ \boldsymbol{a} \in \{0,1\}. \tag{10}$$

The MIP (6)–(10) minimizes the total reduced costs of the corresponding product sequence defined by the $\boldsymbol{z}$ variables. Constraints (7) ensure that $a_{i,m}$ can take a positive value only if product i is preferred to all other products in assortment S_m. The set of constraints (8) and (9) represent nonreflexivity and transitivity, respectively, in order to establish a strict order among all products. Unfortunately, system (6)–(10) is costly to solve, and one needs to find alternatives for practical purposes. The authors therefore use a local-search heuristic to find new columns with reduced costs. Both the heuristic and the exact model (6)–(10) generate fully ranked customer behaviors, which may not be necessary in practice (see Observation 1). We therefore propose to strictly rank only products with high ranks (i.e., those that are considered more preferred) and to explicitly take advantage of the structure of the proposed choice model.

The partially ranked customer behaviors with indifference subsets can be efficiently represented by a preference tree, in which explicitly listed nodes refer to strictly ranked products. Although the presented methods also apply to the general case of partially ranked customer behaviors (see Definition 4 in Appendix A.1), we will focus, without loss of generality, on the simpler case of $\sigma = (P(\sigma), I(\sigma))$ as specified in Definition 3. Here, the indifference set contains all nodes that are not strictly ranked, that is, $I(\sigma) = \mathcal{N} \backslash P(\sigma)$, and therefore does not need to be explicitly listed in the tree. Figure 1 illustrates a small example with three products and the no-purchase option 0. In this example, a total of $|K| = 8$ customer behaviors have been generated. For instance, customer behavior σ_7 refers to a customer that prioritizes product 2 if present; if not, she is willing to buy product 1. If none of those products is available, the customer will buy any available product or leave the store with equal (i.e., uniform) probability. In contrast, customer behavior σ_6 refers to a customer that will buy product 2 if available and leave the store without purchase otherwise (indicated by the no-purchase option 0).

Such a tree structure may be more intuitive for store managers who may want to understand customer segmentation, since it focuses on the products that are important to explain sales: those that are ranked early in a customer preference sequence. Further, the search for new customer behaviors in the tree structure may drastically speed up computation if one succeeds to limit the search to a significantly smaller space. This is the case if we focus on high ranks first and then gradually expand the tree by not more than one level of depth at

Figure 1. Example of Growing Preference Tree Choice Model for $N = 3$ Products



each branch and iteration. Due to the gradual expansion of the tree, we will refer to it as the *growing preference tree* (*GPT*).

We define σ_j to be a *subbehavior* of σ_i if $P(\sigma_j) = (P(\sigma_i), \ell)$, where $\ell \in I(\sigma_i)$. In words, a subbehavior σ_j inherits the strict preference list from its parent σ_i and adds to it one product (including, potentially, the no-purchase option) that is not part of σ_i's preference list. Let $\boldsymbol{\sigma}$ be the set of behaviors enumerated in the GPT. Our algorithm iteratively searches for new subbehaviors in the GPT, expands the tree, and solves problem (1)–(4) to find the corresponding probabilities $\boldsymbol{\lambda}$. When looking for new promising columns (i.e., new subbehaviors), we may restrict the search to the subbehaviors of all $\sigma \in \boldsymbol{\sigma}$. The reduced costs of each of the subbehaviors can be computed as $rc(\sigma) = -\boldsymbol{\alpha}\boldsymbol{a} - \nu$, where $\boldsymbol{\alpha}$ and ν are the dual values of constraints (2) and (3) in problem (1)–(4), and a is defined according to equality (5). The subbehaviors with the lowest reduced cost are then added to the set of customer behaviors $\boldsymbol{\sigma}$, and problem (1)–(4) is resolved. The pricing step is exemplified in Figure 2, in which the reduced cost for all subbehaviors (indicated in blue) of behaviors $\sigma \in K$ are computed, unless the last product in the preference list of σ is option 0. This would indicate that the customer would leave the store and the subbehaviors are irrelevant. The process is performed iteratively until the ℓ_1 error is sufficiently small or a defined maximum number of iterations is performed.

Note that computing the reduced cost of a fully ranked preference list has a computational complexity of $O(N \cdot M)$, even if the reduced cost of a "similar" fully ranked preference list is known. Therefore, finding new columns with negative reduced costs can become costly when using fully ranked preference lists. In contrast, using partially ranked preference lists with the GPT holds the remarkable advantage that, once the reduced cost of a partially ranked list is known, the reduced cost of any of its subbehaviors can be computed in $O(M)$ time. The reduced cost of the subbehavior (which will additionally rank item i) can be quickly adjusted by accounting for the different choices that may occur in each of the M assortments. It turns out that the reduced cost is impacted only in the case when the original preference sequence selected items from the indifference set, but the new subbehavior would select item i if $i \in S_m$. In this case, the corresponding dual value $\alpha_{i,m}$ has to be added, and one has to subtract the previous contribution of the indifference set from the reduced cost. This makes the exploration of the search space for negative reduced cost columns extremely efficient. A detailed description can be found in Appendix B.2.

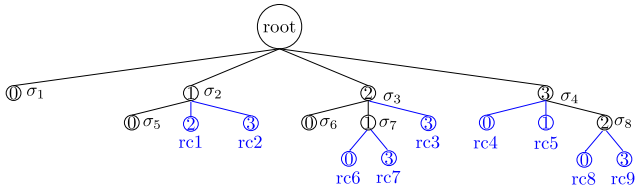
The general steps of the GPT-based column-generation method can be summarized as follows:

1. Generate initial behaviors and add them to the tree.
2. Compute reduced costs of subbehaviors of all behaviors and add those with the most negative reduced costs to the tree.
3. If no negative reduced cost column has been found, execute MIP (6)–(10) to find the most negative reduced cost column; if the cost is negative, add the behavior to the tree.
4. If none of the columns found in the previous two steps have negative reduced cost, terminate the algorithm. Otherwise, return to step 2.

A detailed description and pseudocode for the algorithm can be found in Appendix B.1. Note that, in our computational experiments, we have never encountered cases where solving MIP (6)–(10) was necessary, given that the previous steps always found sufficient subbehaviors with negative reduced cost to reach the required accuracy threshold. Contrary to directly using MIP (6)–(10) to identify the customer behavior with the lowest reduced cost, the GPT allows for controlling which type of customer behaviors to consider. For example, if indifferent sets are not at all desired by the modeler, one only needs to consider the subbehaviors that end in the no-purchase option 0 and set matrix $\boldsymbol{A}$ accordingly. The GPT would then only generate strictly ranked customer behaviors, but most likely converge much faster than when using the MIP (6)–(10) or a local search.

We close this section by noting that, even though we have illustrated the model estimation by minimizing the linear ℓ_1 loss function, the general methodology of the GPT can be used with any convex loss function.

Figure 2. (Color online) Computing Reduced Costs in the Growing Preference Tree Choice Model



We discuss those implementations in Appendix 1.6 of the online supplement by using the nonlinear loss function of the Kullback divergence (Jagabathula and Rusmevichientong 2019).

4. Assortment Optimization

In the previous sections, we have presented a new representation for rank-based choice models and an efficient methodology to identify a set $\boldsymbol{\sigma}$ of relevant customer behaviors $\sigma_k \in \boldsymbol{\sigma}$ as well as their corresponding probabilities λ^k. We now focus on how to identify optimal assortments that are coherent with the learned choice model.

Aouad et al. (2018) recently showed that assortment optimization based on a given choice model is generally NP-hard. Some authors therefore proposed to limit the search space in order to remain computationally tractable. Jagabathula (2014) proposes a local search for assortment optimization relying on a revenue prediction subroutine based on a general choice model. Honhon et al. (2012) provide efficient algorithms for several special cases of rank-based models with $O(N^2)$ customer types, where N is the number of products. Berbeglia and Joret (2020) provide tight bounds for the performance of so-called revenue-ordered assortments that respect the *regularity assumption*, therefore including the general class of random utility models.

Restricting the number of products that each customer may want to buy to smaller *consideration sets* has also found more popularity. In this spirit, Aouad et al. (2015) solve the assortment optimization problem with consideration sets via dynamic programming. Jagabathula and Rusmevichientong (2016) jointly solve the price and assortment optimization problem under a two-stage choice process where customers first consider a subset of products cheaper than a certain price threshold and then choose their favorite product among them.

First general MIP formulations that consider the entire search space have been introduced by McBride and Zufryden (1988) and Belloni et al. (2008). Initially, those models were limited to small problem instances. Nevertheless, some recent works have proposed MIP formulations that scale reasonably well (see, e.g., Farias et al. 2013; Bertsimas and Mišic 2016, 2019). We will use those works as a starting point to optimize on partially ranked choice models.

We suppose that a revenue r_i is associated with each product i. The no-purchase option 0 yields a revenue of 0. Bertsimas and Mišic (2016, 2019) propose a mixed-integer programming model that uses variables x_i that take value 1 if product i is included in the assortment, and 0 otherwise. It further uses variables y_i^k that take value 1 if product i is within the assortment and is chosen according to behavior σ_k. The optimization problem can be written as

$$\max_{x,y} \quad \sum_{k=1}^{K} \sum_{i=1}^{N} r_i \lambda^k y_i^k \tag{11}$$

$$\text{s.t.} \quad \sum_{i=0}^{N} y_i^k = 1 \qquad \forall k \in \{1, \dots, K\}, \tag{12}$$

$$y_i^k \le x_i \qquad \forall k \in \{1, \dots, K\},\ \forall i \in \{1, \dots, N\}, \tag{13}$$

$$\sum_{j:\sigma^k(j)>\sigma^k(i)} y_j^k \le 1 - x_i \qquad \forall k \in \{1, \dots, K\},\ \forall i \in \{1, \dots, N\}, \tag{14}$$

$$\sum_{j:\sigma^k(j)>\sigma^k(0)} y_j^k = 0 \qquad \forall k \in \{1, \dots, K\}, \tag{15}$$

$$\boldsymbol{x} \in \{0,1\},\ \boldsymbol{y} \ge 0.$$

The optimization problem (11)–(15) maximizes the expected revenue. Constraints (12) select exactly one product for each customer type k. Constraints (13) say that a product can be selected only if it is part of the assortment. Constraints (14) guarantee that the product in the assortment that is ranked highest also has the highest y_i^k value. Finally, constraints (15) ensure that all products ranked lower than the no-purchase option 0 are not selected.

For reasonably sized problem instances, problem (11)–(15) is computationally tractable. In particular, even though defined as continuous variables, variables $\boldsymbol{y}$ will only take binary values due to the structure of the problem. Bertsimas and Mišic (2019) explicitly showed that the formulation yields stronger linear programming relaxation bounds than the formulation initially proposed by Belloni et al. (2008). Unfortunately, the formulation requires fully ranked customer behaviors, which are not explicitly given by a choice model with partially ranked behaviors as generated by the GPT column-generation algorithm. To adapt partially

ranked behaviors to the MIP stated above and obtain an exact approach, we could transform the partially ranked choice model into a fully ranked choice model. However, the number of necessary fully ranked preference lists is factorially large (see Theorem 1), which would make the resulting MIP intractably complex. As a heuristic approximation, one may replace the indifference sets by a random sequence of those products that are not strictly ranked. Generating several random subbehaviors for each partially ranked customer's behavior, generally known as *boosting*, may be computationally tractable while improving the performance of the final assortments. However, it may be quite instance specific how many of those random sequences to generate, and the models may become too large anyhow, without mentioning the heuristic nature of the method.

We are therefore interested in finding an optimization model that directly operates on partially ordered ranks, that is, those that use a strict ranking on a subset of products and indifference on the remaining products. To directly operate on customer behaviors with indifference sets, we may add to problem (11)–(15) the following constraints, which enforce that y variables for products i and j have equal values if both products are part of the assortment and have equivalent rank in behavior k. Namely,

$$z_{ij} = x_i \cdot x_j \quad \forall i,j \in \{1,\dots,N\} : i > j, \tag{16}$$

$$|y_i^k - y_j^k| \le 1 - z_{ij} \quad \forall k \in \{1,\dots,K\};\ \forall i,j \in \{1,\dots,N\} : i > j \text{ and } \sigma^k(i) = \sigma^k(j). \tag{17}$$

The introduction of variables z_{ij} and the linearization of constraints (16) and (17) would significantly increase the model size and the difficulty of solving the problem. Fortunately, an equivalent model can be achieved by adequate transformation and substitution of the new variables.

Theorem 2. *The feasible set of the optimization model composed by* (11)–(15) *and* (16)–(17) *is equivalent to the feasible set of the optimization model composed by* (11)–(15) *and the constraints* (18)–(19)*:*

$$y_i^k - y_j^k \le 2 - x_i - x_j \quad \forall k \in \{1,\dots,K\}\ \ \forall i,j \in \{1,\dots,N\} : i > j \text{ and } \sigma^k(i) = \sigma^k(j), \tag{18}$$

$$-y_i^k + y_j^k \le 2 - x_i - x_j \quad \forall k \in \{1,\dots,K\}\ \ \forall i,j \in \{1,\dots,N\} : i > j \text{ and } \sigma^k(i) = \sigma^k(j). \tag{19}$$

We refer to Appendix A.6 for a proof of Theorem 2. Given the above equivalence, one may directly optimize the assortment based on partially ranked consumer behaviors without introducing new variables. Although the structure of problem (11)–(15) forces the continuous $\boldsymbol{y}$ variables to take binary values, adding constraints (18) and (19) breaks this structural property, allowing the variables to take any continuous value between 0 and 1. As outlined above, these constraints have an intuitive interpretation. They force y_i^k and y_j^k to take the same value if both products i and j are part of the assortment and have equal rank in customer behavior k. In this case, if none of the higher ranked products are available in the assortment, each y_i^k for the indifferent products will take value $1/|I(\sigma) \cap S|$, where S is the assortment defined by variables $\boldsymbol{x}$. Even though there are a quadratic number of constraints, those are only on the size of the indifference sets. If the indifference sets are large, one may generate those constraints on the fly, adding only those constraints that are violated in the linear programming solution in each of the branch-and-bound nodes.

5. Computational Results

We now focus on empirical experiments performed with the proposed nonparametric choice model. In Section 5.1, we will evaluate the performance of the choice model and the assortment optimization algorithms on synthetic data. In particular, we compare our approach to two existing benchmarks in Section 5.1.1, evaluating how well these models perform in terms of scalability and ability to learn the choice model. Assortment optimization on the synthetic data is performed in Section 5.1.2. Finally, in Section 5.2, we train the choice models on an industrial data set from the clothing retail sector.

We note that additional analysis and results can be found in the appendices. In particular, Appendix 1 in the online supplement focuses on additional results for the estimation method, including, among others, the characteristics of the generated choice models when modifying parameters such as accuracy thresholds, ground-truth (GT) model, or the loss function. Appendix 2 in the online supplement focuses on additional results for the assortment optimization problem.

All computational experiments have been carried out on a single Intel Xeon X5650 2.67 GHz processor, limited to 40 GB of memory. The algorithms have been coded in Python version 3.5. If not stated otherwise, all mathematical models have been solved using the MIP solver of Gurobi version 8.0.1.

5.1. Numerical Results on Synthetic Data.

Data Generation. We generate sales and assortment data according to a GT model. This data will be used to evaluate the performance of the nonparametric choice models discussed so far and the corresponding algorithms proposed in the previous sections. We choose as the GT model a MMNL model with T classes of customers. The probability distribution among the classes are drawn from the T-dimensional simplex p_t. (Therefore, $\sum_t p_t = 1$ and $p_t \geq 0\ \forall t$.) Each customer class t associates a utility $u_{t,i}$ with a product i. The overall probability that a random customer chooses a product i is given by

$$\mathbb{P}(i|S) = \sum_{t=1}^{T} p_t \frac{e^{u_{t,i}}}{e^{u_{t,0}} + \sum_{j \in S} e^{u_{t,j}}}.$$

The utilities for each customer class are generated as proposed by Bertsimas and Mišic (2016). Specifically, we generate a matrix q of the same dimension as u uniformly distributed on $[0,1]$. If not stated otherwise, four of the $N+1$ products from $\mathcal{N} \cup \{0\}$ are randomly selected for each customer class t. Those products are assumed to have high utilities for customer class t, computed as $u_{t,i} = \log(10 \cdot q_{t,i})$. The utilities for the remaining $N-4$ products are set to $u_{t,i} = \log(0.1 \cdot q_{t,i})$. The training and test sets for the choice models each consist of M assortments. In all experiments, M has been set to 20 to reflect a context where the number of historical observations is limited. Assortment densities r are set to 0.5, that is, each assortment contains $N/2$ products. Utilities $u_{t,i}$ are translated to a vector of sales probabilities $v_{i,m} = \mathbb{P}(i|S_m)$ for each product in each assortment. Sales transactions are then randomly generated according to the sales probabilities. Recall that the sales vector v is indexed by tuples (i,m). Therefore, the choice matrix $A^k_{i,m}$ has two dimensions.

Estimation of the Choice Models. The choice model is trained by iteratively generating new customer preference lists via column generation and solving the master problem which minimizes the estimation error based on the training data. Let $\boldsymbol{A}_{tr}$ and $\boldsymbol{v}_{tr}$ be the choice matrix and the sales probabilities for the M assortments of the training set. At each iteration, we compute the current training error as $\epsilon_{tr} = |\boldsymbol{A}_{tr}\boldsymbol{\lambda} - \boldsymbol{v}_{tr}|$. The test error is computed in the same way, but using choice matrix $\boldsymbol{A}_{te}$ and sales vector $\boldsymbol{v}_{te}$ based on the M assortments of the test data. Note that the scale of the training and test errors depends on the number of assortments in the training and test data. We may therefore normalize the errors, dividing by $2 \cdot M$ to obtain error values between 0 and 1. The estimation procedure then terminates once the normalized training error is smaller than or equal to threshold ϵ_0 (i.e., we stop when $\epsilon_{tr} \leq 2 \cdot M \cdot \epsilon_0$). If not otherwise stated, we set ϵ_0 to 0.01.

5.1.1. Estimation of Choice Models.

In the following, we will investigate how well the proposed choice model can be trained using our column-generation approach based on the growing preference tree (CG-GPT). We will compare our model with two existing approaches: the column-generation approach from Bertsimas and Mišic (2016), using a local search to find new fully ranked preference lists (CG-LS), and the k-deletion heuristic from Jagabathula and Rusmevichientong (2019). All approaches have been executed on the same set of 10 randomly generated instances for each $N \in \{30, 50, 100, 250, 500, 1{,}000\}$. Computing time has been limited to 12 hours.

Comparison with Benchmark CG-LS. Table 1 reports the average results over the 10 instances for two approaches and for different problem sizes N. The table reports the final and test training errors, the computing

Table 1. Learning Performance for CG-GPT and CG-LS Algorithms with $\epsilon_0 = 0.01$ (Averaged over 10 Random Instances)

	CG-GPT					CG-LS					
	Train	Test	#	Time		Train	Test	#	Time		# inst.
N	error	error	iter	(min)	K	error	error	iter	(min)	K	unsolved
30	0.37	2.88	15.9	<1	170.7	0.39	3.74	309.0	1.5	177.5	0
50	0.38	2.90	31.1	<1	310.5	0.40	4.22	619.0	8.4	337.3	0
100	0.39	2.72	46.9	<1	619.0	0.40	4.41	1,183.1	76.3	667.7	0
250	0.39	2.37	90.6	16.6	1,474.1	—	—	—	—	—	10
500	0.40	1.89	116.5	76.1	2,437.1	—	—	—	—	—	10
1,000	0.40	1.30	159.7	306.0	3,876.7	—	—	—	—	—	10
All (average)	0.39	2.34	76.8	66.6	1,481.4	0.40	4.13	703.7	28.7	394.2	30

Note. CG-GPT, Column Generation–Growing Preference Tree; CG-LS, Column Generation–Local Search; # iter, number of iterations; # inst., number of instances.

times, the number of iterations, and the final number K of generated preference lists with nonnegative probabilities λ_k. Column "# inst. unsolved" indicates the number of instances that have either hit the 40 GB memory limit or run out of time. Those instances have not been considered in the average values. The CG-GPT has successfully trained the choice model to the required threshold of $\epsilon_0 = 0.01$ (which corresponds to a training error of 0.4 when $M = 20$) for all instances within the given time and memory limits. In contrast, the CG-LS is not able to converge for instances with more than 100 products. In a direct comparison, the CG-GPT is more scalable, converges much faster, and provides better test errors than the CG-LS. In particular, one observes that the number of iterations required by the CG-GPT does not increase much as N increases. This is explained by the fact that the indifference sets have a larger explanatory power in those instances, which allows the model to have small training errors with relatively few columns. These are direct practical implications of Lemma 1 and Theorem 1. In fact, further experiments have shown that restricting the number of columns to those of highest probability has relatively little impact on CG-GPT test error, whereas CG-LS performance significantly deteriorates in this case. Detailed results and further discussions can be found in the online supplement in Appendix 1.7.

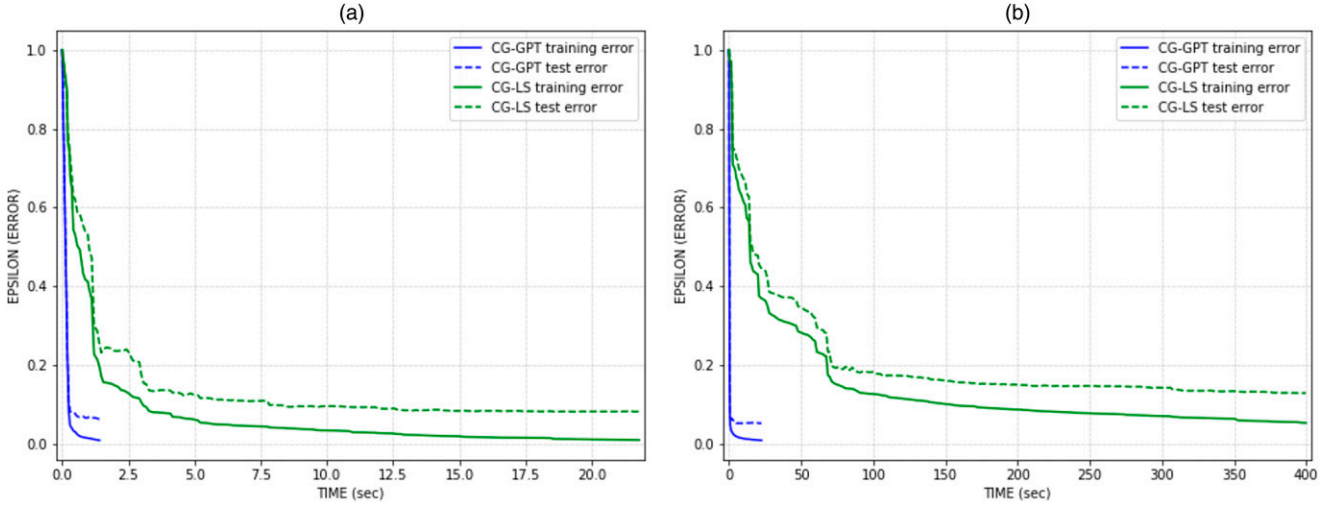
Figure 3, (a) and (b) visualize the evolution of the normalized training and test errors for two problem instances with $N = 30$ and $N = 100$, respectively. We note that CG-GPT quickly converges in both cases, achieving better test errors than CG-LS in much shorter time. Also, with 100 products, CG-LS was not able to converge within the reported time interval.

Note that, to make sure that we are comparing with a fair implementation of CG-LS, we have tried different neighborhoods explored by the local-search heuristic of CG-LS. However, those implementations did not improve its convergence and were computationally limited to $N < 250$. Detailed results are reported in the online supplement in Appendix 1.2.

Comparison with Benchmark k-Deletion Heuristic. Based on the idea that one only needs to rank k items when at most $k-1$ items are missing in any assortment, Jagabathula and Rusmevichientong (2019) propose to use the k-deletion heuristic which enumerates all preference sequences with k strictly ranked products. The computational burden of such an approach quickly increases when k becomes bigger. This technique was therefore originally proposed to deal with assortments where the number of missing products is much smaller than the total number of products. Given its computational burden, we test the k-deletion approach only for small values of k, namely, 2, 3, and 4, on assortments of size $N/2 >> k$. Since, in this case, there may be none of the ranked products in the assortment, we add the no-purchase option 0 at rank $k+1$ (if not ranked before).

Table 2 reports the computation results,[2] confirming that the k-deletion method does not scale well when increasing k. With $k = 3$, one is limited to 250 products, and with $k = 4$, one is limited to 100 products. (Note that we have omitted lines for N where none of the 10 problems were solved by k-deletion.) The method is therefore not competitive with the CG-GPT, which achieves better training and test errors with a much smaller

Figure 3. (Color online) Learning Curves (Normalized Training and Test Errors) for CG-GPT and CG-LS on Two Example Problem Instances



Problem instance with $N = 30$ Problem instance with $N = 100$

Table 2. Properties of Choice Models Generated with the k-Deletion Method, with an Assortment Density $r = 0.5$

		k-deletion				
		Train	Test	Time		# inst.
k	N	error	error	(min)	K	unsolved
2	30	6.80	9.57	<1	93.7	0
2	50	7.03	10.65	<1	179.2	0
2	100	6.14	10.47	<1	400.4	0
2	250	5.66	10.32	<1	1,155.8	0
2	500	4.46	10.33	3.9	2,530.9	0
2	1,000	3.73	10.40	12.3	5,499.8	0
2	All (average)	5.64	10.29	2.8	1,643.3	0
3	30	1.90	5.69	<1	196.7	0
3	50	1.85	6.27	<1	362.7	0
3	100	1.04	6.38	6.4	857.5	0
3	250	0.56	6.16	120.1	2,351.8	4
3	All (average)	1.34	6.12	31.8	942.2	24
4	30	0.02	3.98	2.3	278.4	0
4	50	0.03	4.36	20.2	479.4	0
4	100	0.00	5.46	493.0	980.0	9
4	All (average)	0.02	4.60	171.8	579.3	39

Note. # inst., number of instances.

number of behaviors (see, e.g., Table 1 above and Table 10 in the online supplement in Appendix 1.5). For small k, the k-deletion heuristic seems to not be flexible enough to provide a good fit of the data. For $k = 4$, the small training errors but rather large test errors may indicate overfitting (see also online supplement, Appendix 1.3). One possible cause may be the no-purchase option ranked after the first k strictly ranked products. On unseen assortments, the no-purchase probability is likely overestimated, particularly when the assortment density r is small. We here note that for CG-GPT, when using an accuracy threshold of $\epsilon_0 = 0.1$ such that it also strictly ranks not more than three products, the average test error is almost 40% smaller than the one of k-deletion with $k = 3$ (see online supplement, Appendix 1.5). This illustrates the importance of the complementary indifference set to improve predictive accuracy.

We close this section by referring the reader interested in further analysis and results of the proposed estimation procedure to Appendix 1 in the online supplement. For example, our approach is not limited to linear loss functions, but can be used to minimize any other convex loss function (since strong duality still holds to prove optimality; see, e.g., Griva et al. 2009, theorem 14.37). In the online supplement, Appendix 1.6 numerically compares the CG-GPT with different objective functions, providing empirical evidence of the robustness and flexibility of the proposed technique. An analysis of choice model characteristics such as sparsity and concision can be found in the online supplement in Appendices 1.4 and 1.5.

5.1.2. Assortment Optimization. In the previous section, we have shown that the partially ranked choice model can be accurately learned in reasonable computing times even in contexts with large numbers of products. We now explore the scalability of the mathematical models to optimize assortments once the choice model has been learned. We have implemented four different approaches. The first two approaches learn a partially ranked choice model using the CG-GPT algorithm. The first approach performs subsequent assortment optimization by adding the indifference inequalities (18) and (19) via branch-and-cut (referred to as AO-B&C). The second approach performs boosting to create several fully ranked preference lists at random and then optimizes via the classical MIP (11)–(15) based on fully ranked lists (referred to as AO-Boost). Finally, the two other approaches are used as benchmarks, namely, one based on the CG-LS, and one based on the k-deletion heuristic. Both are used with the classical MIP (11)–(15) (referred to as AO-Compl). Given that AO-Boost is an approximation of AO-B&C and has been clearly outperformed by the latter, we here focus on the direct comparison with the two other approaches. The computational comparison with AO-Boost can be found in the online supplement in Appendix 2.1.

Comparison with CG-LS Based Approach. We now investigate how well the assortment optimization approaches for CG-GPT and CG-LS scale to a large number of products. For all experiments, we have used the choice model obtained after 12 hours of training with a 40 GB memory limit, no matter whether estimation had

converged by then. The second set of experiments assumes that the choice model is composed by the 100 customer preferences that have highest λ probabilities within the choice model.[3]

Table 3 summarizes the results and reports for each problem size N the averages of the optimal GT revenues (computed as proposed by Bront et al. 2009). For each approach (i.e., CG-GPT and CG-LS), the table reports the average size K of the choice models, the average computing times to solve the optimization model, the average revenue as computed by the GT model, and the average optimality gap after 12 hours of assortment optimization.

In the first set of experiments using the entire choice models trained, K inevitably grows as the number of products N increases. It must be noted that the CG-LS would generally also have growing K. However, for large N the choice models could not be solved within the time limit (given that the local search is more time-consuming; see Table 1), which results in small choice models (and high test errors). The AO-B&C approach based on the CG-GPT consistently generates assortments with higher GT revenues. However, with larger N it quickly becomes more difficult to solve, as the number of preference sequences K becomes prohibitively large, requiring the addition of too many inequalities (18) and (19) when using indifference sets.

However, as previously discussed, partially ranked choice models tend to achieve better generalization with a sparser model than fully ranked ones. (We refer the reader again to Section 3.1 and to the online supplement in Appendix 1.7.) Furthermore, using a smaller set of preference sequences directly facilitates the solution of the assortment optimization problem. The second set of experiments, limiting the number of preference sequences to 100, yields convincing results. All models are solved to optimality within a few minutes. The CG-LS based approach tends to significantly lose prediction accuracy, resulting in assortments of lower quality (i.e., lower GT revenue). In contrast, the CG-GPT based approach maintains its predictive performance and results in assortments that have a close-to-optimal GT revenue. We conclude that limiting the number of preference sequences used in the assortment optimization seems to be a promising technique to remain computationally tractable while maintaining a choice model with high predictive performance. We refer the interested reader to Appendix 2.3 in the online supplement for the results of further computational experiments in which the number of preference sequences has been limited.

Comparison with k-Deletion Based Approach. As a last study, we investigate the performance of assortment optimization based on the choice models obtained by the k-deletion heuristic. Table 4 compares the revenues of the assortments obtained by CG-GPT with AO-B&C and the k-deletion with AO-Compl. Column "# inst unsolved" reports the number of instances that could either not be estimated by the k-deletion heuristic or not be solved in the assortment optimization model (exceeding the available memory). Lines without any results (i.e., the "# inst unsolved" equals 10) have been omitted. The results, averaged only over instances where the k-deletion heuristic was tractable, confirm that the k-deletion is not an appropriate choice in our context.

Table 3. Assortment Optimization Results for Choice Models Generated by CG-GPT and CG-LS (Averaged over 10 Random Instances) with $\epsilon_0 = 0.01$

			CG-GPT - AO B&C				CG-LS - AO-Compl			
		Opt GT		Time	GT	Opt		Time	GT	Opt
	N	revenue	K	(min)	revenue	gap %	K	(min)	revenue	gap %
$\epsilon_0 = 0.01$	30	78.73	170.7	<1	78.05	0.00	177.5	<1	76.37	0.00
max K = full	50	84.18	310.5	<1	83.84	0.00	337.3	<1	81.76	0.00
	100	88.51	619.0	3.6	88.24	0.00	667.7	<1	87.31	0.00
	250	91.61	1,474.1	83.4	91.39	0.00	1,175.0	16.4	90.65	0.00
	500	93.51	2,437.1	557.5	93.38	0.30	413.7	13.7	90.92	0.00
	1,000	95.48	3,876.7	720.2	93.00	5.04	150.8	10.2	92.75	0.00
	All (average)	88.67	1,481.4	221.9	87.98	0.89	487.0	6.9	86.63	0.00
$\epsilon_0 = 0.01$	30	78.73	96.3	<1	78.05	0.00	99.2	<1	77.09	0.00
max K = 100	50	84.18	92.2	<1	83.85	0.00	98.7	<1	80.76	0.00
	100	88.51	90.2	<1	88.30	0.00	98	<1	86.14	0.00
	250	91.61	90.4	<1	91.39	0.00	99.3	<1	88.46	0.00
	500	93.51	89.5	1.4	93.40	0.00	100	2.2	87.84	0.00
	1,000	95.48	86.8	4.2	95.42	0.00	100	5.5	91.56	0.00
	All (average)	88.67	90.9	1.1	88.40	0.00	99.2	1.4	81.33	0.00

Note. Opt, optimal; GT, Ground-Truth; CG-GPT, Column Generation–Growing Preference Tree; CG-LS, Column Generation–Local Search; AO, assortment optimization; B&C, branch-and-cut; compl, complete (i.e., fully ranked).

Table 4. Assortment Optimization with k-Deletion Compared Against Optimal Revenues and Revenues Obtained by CG-GPT on Those Instances Solved by Both Methods

				k-deletion - AO-Compl			
		Optimal	CG-GPT		Time		# inst
k	N	GT revenue	GT revenue	K	(min)	GT revenue	unsolved
2	30	78.73	78.05	93.7	<1	74.78	0
2	50	84.18	83.84	179.2	<1	80.11	0
2	100	88.51	88.24	400.4	<1	85.76	0
2	250	91.61	91.39	1,155.8	29.3	85.52	0
2	500	93.51	93.38	2,530.9	459.3	88.22	0
2	All (average)	87.31	86.98	872.0	97.8	82.88	10
3	30	78.73	78.05	196.7	<1	69.10	0
3	50	84.18	83.84	362.7	<1	79.37	0
3	100	88.51	88.24	857.5	1.8	85.08	0
3	250	93.07	92.82	2,351.8	240.5	84.18	4
3	All (average)	86.12	85.74	942.2	60.6	79.43	24
4	30	78.73	78.05	278.4	<1	74.27	0
4	50	84.18	83.84	479.4	<1	81.74	0
4	100	96.65	95.88	980.0	2.2	96.17	9
4	All (average)	86.52	85.92	579.3	<1	84.06	39

Note. CG-GPT, Column Generation–Growing Preference Tree; GT, Ground-Truth; AO, assortment optimization; compl, complete (i.e., fully ranked); # inst., number of instances.

For the choice model estimation, one can only enumerate sequences with up to four items, which is not sufficient to generalize well, resulting in assortments with lower GT revenues. As N gets larger, the choice model become too large, typically hitting the memory limit. Except for one exception ($N = 100$ with $k = 4$), the GT revenues reported by the k-deletion heuristic were consistently inferior to those of the CG-GPT based approach.

We refer to Appendix 2 in the online supplement for further experiments on the performance of the assortment optimization. In particular, Appendix 2.2 in the online supplement investigates the revenue impact of different loss functions and training accuracy thresholds. In Appendix 2.4, we tested the scalability of MIP (11)–(15) on a set of hard instances constructed following the results of Aouad et al. (2018). We conclude this section noting that the recent work of Bertsimas and Mišic (2019) also conducted numerical experiments on the original formulation (11)–(15) for fully ranked lists. The authors further propose a Benders decomposition implementation, which could also be adapted to our formulation. It has to be noted, however, that the assortment optimization MIP itself is not the only crucial ingredient of the overall approach. Learning the choice model correctly and accurately for large-scale problems is, as has been shown, computationally challenging and crucial to obtaining meaningful assortment optimization models, and for this concern, we believe that the partially ranked choice model provides an efficient option.

5.2. Case Study on Industrial Retail Data

We will now discuss an industrial case study based on real-world data from a North American clothing retailer. An anonymized data set on shoe stores has been obtained from our industrial partner JDA Labs (2017). In the following, we will discuss the data sets and preprocessing. We will then explore how well the different approaches perform when training the choice models for the industrial data set.

5.2.1. Data Description and Preprocessing. The data set includes assortment data, transaction data, and product and store characteristics from August 2014 to July 2015. Assortment information is provided for each day and each store (with information such as location, climate, price category). Sales transaction data contains product IDs, sold quantities, sales timestamps, and store ID. Products in the shoe data set have characteristics given as categorical values (class, subclass, brand, material, color) and continuous values (average price). Products in the shirt data set contain additional information (lifestyle, pattern, fit, sleeve length, fashion).

Based on this information, we define an assortment as the set of products offered in a particular store throughout one calendar week in a particular year. For each assortment, we link the corresponding sales transactions and convert those into the vector of sales probabilities v, representing the probability of selling a certain product in a given assortment. The following describes the entire process of data preparation. For more details, we refer to the master's thesis of Palmer (2017).

Store Clustering. Assuming that stores in neighborhoods with similar characteristics meet the needs of similar customer types, we need to learn separately for groups of similar stores. As a result of the frequent discussions with JDA Labs (2017), which provided the data, we clustered the stores according to four store features: location (state and city), climate (four different categorical values), price band (low, medium, and high), and percentage of sales in each subcategory. Given that our features contain both categorical and continuous values, the clustering has been performed using an extension of k-means that can handle mixed categorical and continuous data (Huang 1997).

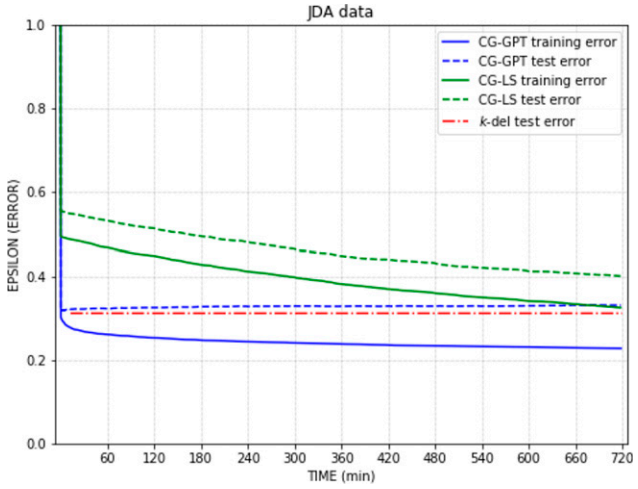
Data Preprocessing and No-Purchase Estimation. We arbitrarily selected a cluster from the shoes data set that contains 10 stores and has a fairly high number of sales. For each of these stores, we use data during 10 consecutive weeks from autumn 2014, which we consider a good trade-off to have sufficient data, while not risking that store assortments changed much due to seasonal fashion. We considered each week of store data as a proper assortment, resulting in a total of 100 assortments. To ensure that the historical data are statistically meaningful, we only considered products that have been sold at least 10 times, resulting in a final total of 299 different products.

Given that we did not have any information about how many customers left the store without a purchase, we estimated the no-purchase probabilities assuming that assortments with many sales have lower no-purchase probabilities and assortments with few sales have higher no-purchase probabilities (but always between 10% and 30%). Let $v_{0,m}$ be the no-purchase probability in assortment S_m. Let $s^{\#}_m$ be the number of sales observed in assortment S_m. Let $s^{\#}_{\text{MIN}} = \min_{m\in\mathcal{M}}\{s^{\#}_m\}$ and $s^{\#}_{\text{MAX}} = \max_{m\in\mathcal{M}}\{s^{\#}_m\}$. The no-purchase probability for a store m is computed as a linear interpolation between 0.1 and 0.3 according to the number of observed sales: $v_{0,m} = 0.1 + 0.2 \cdot \frac{s^{\#}_m}{s^{\#}_{\text{MAX}} - s^{\#}_{\text{MIN}}}$. Let $s^{\#}_{i,m}$ denote the number of sales of product i in assortment S_m. We then compute the sales probability for any other product i as $v_{i,m} = (1 - v_{0,m}) \cdot \frac{s^{\#}_{i,m}}{s^{\#}_m}$, which is the corresponding sales proportion of product i taking into consideration the probability for the no-purchase option.

5.2.2. Computational Results. ***Convergence of Training the Choice Model.*** Figure 4 plots the convergence curves for the CG-GPT and CG-LS training approaches. To warm-start the CG-LS method, we initialized it with singleton preference lists as in Van Ryzin and Vulcano (2015), that is, we generate one preference list for each product, with that product being ranked first, and the no-purchase option ranked second. The initial set of preference sequences therefore corresponds to an independent demand model. We also plot the test error achieved by the k-deletion approach once the corresponding linear program has been solved. Only the case for $k = 2$ has been reported, since higher values of k ran into memory errors.

We note that the optimal training error is not 0 (as is the case for synthetic data), because real data are noisy or even contradictory.[4] Of course, one may group items in a preprocessing step as proposed by Jagabathula and Rusmevichientong (2019), which would reduce data sparsity and improve both the training and the test errors. However, we refrain from using such techniques here for two reasons. First, we did not find any

Figure 4. (Color online) Learning Curves (Normalized Training and Test Errors) for CG-GPT and CG-LS on Industrial Shoe Data with 299 Products



grouping criteria which would be meaningful for the given context. Second, the primary interest of our study is to explore the efficiency of the methods when predicting sales probabilities for individual items.

The training error for the CG-GPT reaches about 25% in one hour and about 22% after 12 hours. The CG-LS only reaches a training error of 32% after 12 hours. Test errors are, as expected, slightly higher than the training errors. For the CG-LS, the test error starts high and monotonically decreases to about 40% after 12 hours. For the CG-GPT, the test error finds its minimum (at about 31%) surprisingly fast, after a few iterations, and then slightly starts to overfit, increasing to about 33% after 12 hours. Remarkably, our method reaches that point after a few minutes, whereas the CG-LS never reaches it within 12 hours. Note that both methods start with a similar set of initial preference sequences. However, instead of ranking the no-purchase option 0 after the first product, the CG-LS uses the complementary indifference set. The initial training and test errors therefore confirm, once again, the importance of the indifference set to explain the training data and to improve prediction accuracy on the test data. Interestingly, the k-deletion method achieves about the same level of generalization as the CG-GPT approach (with a test error of about 31%), requiring a similar amount of computing time (12 minutes for k-deletion and 6 minutes for the CG-GPT). This is a surprisingly good performance, since the k-deletion heuristic did not provide satisfactory results on synthetic data in previous experiments. We may conclude that, for this particular industrial data set (with the techniques used to cluster the assortments), a more general partially ranked choice model generalizes slightly better than a more refined one. On the other hand, capturing substitution of at least two dimensions seems to work well, since the CG-LS initialized with independent demand preference sequences (which is equivalent to a k-deletion with $k = 1$) did not yield good test errors, but could further be improved (gradually, by the CG-LS itself and by the k-deletion heuristic with $k = 2$). This and the fact that the training error generally remained quite high might be indicators that the performance of rank-based models on certain industrial data sets may still be improved in the future. Finally, also note that the same pattern has been confirmed by a five-fold cross-validation analysis. After splitting the 100 assortments into five groups of 20 assortments each, five different experiments were carried using one of these groups as the test set and the other four groups as the training set. All experiments yielded similar conclusions.

6. Conclusion

In this work, we have focused on nonparametric rank-based choice models for assortment optimization. Those choice models have several advantages. Mainly, they can be estimated in a purely data-driven manner without relying on previous knowledge about the market structure. They also tend to be less sensitive to overfitting. Our work proposes a new methodology to estimate those choice models, which scales to a large number of products. In particular, we propose to represent customer behaviors not by fully ordered preference lists of all products, but only a subset of them. This is a realistic setting, which directly exploits the fact that products with low ranks have little explanatory power and impact on buying behavior. Further, we show that any partially ranked choice model can be transformed into an equivalent fully ranked choice model and vice versa. The partial representation of the strictly ranked products enables us to efficiently train the choice model by gradually expanding a tree, in which each of the nodes represents partial lists of strictly ranked products. On this particular structure, new preference lists can be found efficiently via column generation. We finally present new inequalities to adapt an existing assortment optimization model to our partially ranked choice models. Extensive computational experiments have shown that instances with up to 1,000 products can be efficiently trained and assortments can be optimized in quite low computing times, therefore significantly increasing the capabilities of previous approaches to learn the choice model. Our numerical experiments also revealed that the indifference sets are both useful to explain sales in the training data and to improve predictive accuracy on the (unseen) test data. Given that training the partially ranked choice model by means of a growing tree and column generation has been proven to be very efficient in the case of assortment optimization, it may be a promising avenue to explore in other contexts in which discrete choice models are central.

Acknowledgments

The authors thank JDA Labs, in particular, Marie-Claude Côté, for providing the industrial data sets and for their support throughout this research, and the anonymous referees for their meaningful comments and suggestions.

Appendix

A. Theoretical Results

In this appendix, we provide theoretical developments and results, such as the definition of a more general choice model, as well as formal proofs.

A.1. General Partially Ranked Customer Behaviors. We now define a generalization of the simple partially ranked customer behavior (see Definition 2). Consider a customer behavior with q lists of preferred products $P^\ell(\sigma)$ $(\ell = 1, \ldots, q)$ and q indifference sets $I^\ell(\sigma)$ as follows.

Definition 4 (General Partially Ranked Customer Behavior)**.** A general customer behavior σ has an ordered list of several strictly ranked preference lists $P^1(\sigma), \ldots, P^q(\sigma)$ and indifference sets $I^1(\sigma), \ldots, I^q(\sigma)$, all of which are mutually exclusive subsets of $\mathcal{N} \cup \{0\}$. We may write such a general customer behavior as $\sigma = (P^1(\sigma), I^1(\sigma), P^2(\sigma), I^2(\sigma), \ldots, P^q(\sigma), I^q(\sigma))$.

As is the case for simple partially ranked behaviors, each product from $\mathcal{N}$ can be part of only one strictly ranked list or indifference set. Here, the customer would prefer to buy a preferred product within sequence $P^1(\sigma)$. If none of these products are available, the customer will choose any product with similar characteristics defined in $I^1(\sigma)$ and available in the assortment with uniform probability. If none of those products are available, further lists of preferred products and sets of indifferent products may follow. The position of no-purchase option 0 may result in equivalent notations which are in line with those for simple partially ranked behaviors mentioned above. In particular, if the no-purchase option 0 is generally available, it can be either in one of the strict preference lists or in one of the indifference sets. Otherwise, it is implicitly assumed to be at the end of the sequence indicating that no purchase is made.

To illustrate a general partially ranked behavior, recall the previous example of a simple partially ranked customer behavior $\sigma = (P(\sigma), I(\sigma)) = (3, 4, 1, \{2, 5, 6\}, 0)$ with one strictly ranked product list and one indifference set. In the case in which the customer does not find any of the indifferent products 2, 5, or 6 in the assortment, she may have further strict preferences (e.g., on a different product type), given by another strictly ranked product list, say, $(7, 10, 9)$. If those products are also not available in the assortment, the customer may be indifferent on products 8 and 11. In the absence of those products, she may want to leave the store. This more complex behavior is represented by $\sigma = (3, 4, 1, \{2, 5, 6\}, 7, 10, 9, \{8, 11\}, 0)$, having two strictly ranked products lists and two indifference sets.

Although a particular instance of the general partially ranked behavior could be fit into the taxonomy of Lebanon and Mao (2008), the generic general partially ranked behavior model itself cannot be described by its Ξ notation, which assumes that the number, size, and order of strictly ranked parts and indifference sets is fixed beforehand.

A.2. Verification of Observation 1. We may easily verify Observation 1 by noting that the probability that the product at the *kth* rank is selected by the customer equals $(1 - r)^{k-1} \cdot r$, where $(1 - r)^{k-1}$ is the probability that none of the $k - 1$ highest ranked products are selected and r is the probability that product k is subsequently selected. The observation follows, as for any $k > 2$ the probability $(1 - r)^{k-1} \cdot r$ reduces as r increases, and the decrease is exponential in k.

A.3. Verification of Observation 2. We may verify Observation 2 as follows. There are $|S| - |P(\sigma_k)|$ products in the indifference set $I(\sigma_k)$. If none of the strictly ranked products are part of the assortment, the contribution to explaining the sales of one of the products in $I(\sigma_k)$ is $\lambda_k/|I(\sigma_k)|$. Further, the probability that none of the strictly ranked products are part of the assortment is, on average, $(1 - r)^{|P(\sigma_k)|}$ (compare Observation 1). The total average contribution of σ_k to a product $i \in I(\sigma_k)$ follows as the product of the previous two terms.

A.4. Proof of Lemma 1. Let $F = |I(\sigma_p)|$. We generate $F!$ different fully ranked preference lists, each containing the items in $P(\sigma_p)$ followed by a different permutation of the elements in $I(\sigma_p)$. For each of those fully ranked lists, we define an equal probability $(\lambda_p/F!)$. Consider any given assortment S. We now show that the probability of choosing item i in the presence of an assortment S is the same for both choice models. Let $G = |S \cap I(\sigma_p)|$. If $i \notin S$, the sales probability of i is 0 in both cases. If $i \in S$ and $i \in P(\sigma_p)$, then i will be bought with equal probabilities in both cases, since both the partially and the fully ranked lists start with the same sequence of products $P(\sigma_p)$.

For the case in which $i \in S$ and $i \in I(\sigma)$ we need to show that the probabilities that i is bought given S are equal for the two cases (the partial preference list and the set of fully ranked lists). For the partially ranked list, the probability that product i is selected is defined as λ_p/G, that is, the original probability divided by the number of products that are both in the indifference set and in the assortment. Recall that each of the generated fully ranked lists has a probability of $(\lambda_p/F!)$. To compute the probability that i is selected, we multiply this probability by the number of fully ranked lists in which i ranks highest in assortment S. The latter can be obtained in two steps. First, consider only the G items that are both in the assortment and in the indifference set, and compute the number of permutations where i is ranked highest. It can be shown (e.g., via induction) that there are $(G - 1)!$ permutations in which item i is at the first rank. (Note that i has to be at the first rank to be selected, since all other $G - 1$ items are also in the assortment.) Then, for each of those $(G - 1)!$ permutations, we compute the number of possibilities of how to insert the remaining $(F - G)$ items (which are not in S) into the sublist with G items. This can be computed by dividing the number of all permutations of the F items (those in $I(\sigma_p)$) by the number of permutations of the G already ordered items (which are in the assortment), that is, $F!/G!$. The final probability that item i is selected given assortment S therefore amounts to $(\lambda_p/F!) \cdot (G - 1)! \cdot (F!/G!) = \lambda_p/G$. Q.E.D.

A.5. Proof of Theorem 1. In the following, we formally prove Theorem 1. Although Lemma 1 refers to simple partially ranked behaviors as in Definition 2, we here prove valid equivalence for both the simple (Definition 2) and the general (Definition 4) partially ranked behaviors.

To prove the first part of the theorem, consider any preference list $\sigma_c \in \boldsymbol{\sigma}^C$. We may trivially derive an equivalent σ_p from σ_c by defining a corresponding partially ranked customer behavior σ_p with the following parameters: $q = 1$, $P^1(\sigma_p) = P(\sigma_c)$,

$I^1(\sigma) = \emptyset$, and $\lambda_p = \lambda_c$. We may do this for all behaviors in $\boldsymbol{\sigma}^C$ to derive the new choice model. To derive from $(\boldsymbol{\sigma}^P, \boldsymbol{\lambda}^P)$ an equivalent choice model composed of fully ranked consumer behaviors, consider any $\sigma_p \in \boldsymbol{\sigma}^P$. It can be verified that the transformation of a simple partially ranked customer behavior with one set of strictly ranked products and one indifference set in Lemma 1 can be generalized to a general partially ranked customer behavior $(P^1(\sigma), I^1(\sigma), P^2(\sigma), I^2(\sigma), \ldots, P^q(\sigma), I^q(\sigma), 0)$ with several sets of strictly ranked products and several indifference sets. As in Lemma 1, we generate one fully ranked list for each of the permutations of the products in the indifference sets, resulting in a total of $|I^1(\sigma)|! \cdot |I^2(\sigma)|! \cdot \ldots \cdot |I^q(\sigma)|!$ fully ranked lists. Consider a product i, and define r such that product i is either in $P^r(\sigma)$ or in $I^r(\sigma)$. The equivalence of the probability that i is selected is then proven in the same way as in Lemma 1 using $P^r(\sigma)$ and $I^r(\sigma)$. Q.E.D.

A.6. Proof of Theorem 2. We now formally prove Theorem 2. Consider any pair i and j with $i > j$, as well as a customer sequence k. First note that both sets of constraints are defined for the same combinations of i, j, and k, that is, all i and j within $I(\sigma_k)$. Therefore, if $\sigma^k(i) \neq \sigma^k(j)$, neither of the two sets of constraints are defined and, therefore, do not impact the values of y_i^k or y_j^k. If $\sigma^k(i) = \sigma^k(j)$, the equivalence is easiest to show by considering the possible values of binary variables x_i and x_j in a feasible solution. If at least one of the two variables x_i or x_j has value 0, constraints (16) and (18)–(19) allow y_i^k and y_j^k to take any value between 0 and 1 (which are coherent with the other constraints in the original model). Their values would therefore be equivalent. If both variables x_i and x_j have value 1, constraints (16) with right-hand side 0 force y_i^k and y_j^k to take equal values. In the same way, constraints (18)–(19) will have right-hand side 0 and can be combined to obtain $0 \leq y_i^k - y_j^k \leq 0$. It follows that y_i^k and y_j^k have to take equal values. Given that the two sets of constraints are active for the same combinations of i, j, and k, this therefore makes them equivalent. Q.E.D.

B. Additional Details for the Estimation Algorithms

B.1. GPT-Based Column-Generation Algorithm: Pseudocode. We here give a detailed description of the GPT-based column-generation algorithm. We recall that partially ranked customer behaviors with indifference subsets are represented by a preference tree, in which explicitly listed nodes refer to strictly ranked products. In the general case (see Definition 4) with several indifferent sets that are strict subsets of $\mathcal{N}$, each indifferent set can be represented by a node that includes several products. However, selecting those sets may be context specific and require input from store managers on how to cluster those products. In the following, we will give details for the simpler case with $\sigma = (P(\sigma), I(\sigma))$ as specified in Definition 3.

Algorithm 1 (GPT-Based Column-Generation Algorithm)

Input data:

- Sales probability vector v, training set of assortments $S_1, \ldots, S_M$.
- Maximum number of column-generation iterations $iter_{MAX}^{CG}$.
- Optimality training criteria ϵ_{MIN}^{Tr}.
- Maximum number of subbehaviors δ that are added at each iteration.
- Maximum number of attempts $\overline{d}$ to find subbehaviors with negative reduced cost

Output data:

- A set $\boldsymbol{\sigma} = \{\sigma_0, \ldots, \sigma_{K-1}\}$ of K customer behaviors, where $\sigma_k = (P(\sigma_k), I(\sigma_k), 0)$.

```
begin
    Initialize set σ = {σ_0, σ_1, ..., σ_N} defined with P(σ_k) = (k) and I(σ_k) = N\{k};
    Set iter to 0;
    Solve restricted master problem (1)–(4) to obtain λ, ε^+, ε^−, and dual values α and ν;
    while (iter ≤ iter^CG_MAX) and (1^T ε^+ + 1^T ε^− > ε^Tr_MIN) do
        Set iter⟵iter + 1;
        Set attempt⟵0;
        Set NRC⟵False;
        while (NRC = False) and (attempt < d̄) do
            Set attempt⟵attempt + 1;
            for ∀σ_k ∈ σ do
                if P(σ_k)_{|P(σ_k)|} ≠ 0 (i.e., if last element in P(σ_k) is not 0) then
                    compute the reduced costs for all new subbehaviors of σ_k;
                    if ∃ subbehavior with negative reduced cost then
                        NRC⟵True;
            Add to σ up to δ of the generated subbehaviors with lowest reduced costs;
        if (NRC = False) then
            Solve MIP (6)–(10) to find σ_k with smallest reduced cost;
            if (σ_k's reduced cost is negative) then
                Add σ_k to σ;
            else
                Return σ;
        Set σ as new columns to matrix A;
        Solve restricted master problem (1)–(4) to obtain λ, ε^+, ε^−, and dual values α and ν;
    return σ
```

Algorithm 1 outlines the complete column-generation procedure to build the GPT. At each iteration, the algorithm first explores the subbehaviors of the behaviors in σ (i.e., the subbehaviors of the sequences that are already in the restricted master problem at the current iteration). Note that, even if the direct subbehaviors of σ may not have negative reduced cost (see Appendix B.2 for a detailed description on how to compute the reduced costs), behaviors further down the tree may have negative reduced cost. We therefore add to σ up to δ subbehaviors with the smallest reduced costs (even if they are not negative). If none of those subbehaviors have negative reduced cost, the algorithm iterates up to a defined number of attempts (parameter $\bar{d}$). If, after $\bar{d}$ iterations, no negative reduced cost subbehaviors have been found, MIP (6)–(10) can be employed to find the most negative reduced cost column. If this cost is positive, optimality has been proven and the algorithm terminates. Otherwise, the column is added and the procedure starts over. Note that our algorithm inherits the typical convergence guarantees from column generation, given that, in the worst case, it will identify the fully ranked column that has the lowest reduced cost.

The column-generation procedure, as outlined in Algorithm 1, explores the subbehaviors of all $\sigma_k \in \sigma$ (see code line 11). In practice, exploring all nodes may be unnecessarily time consuming. Instead, we may randomly select up to γ behaviors for which the subbehaviors should be explored. In our computational experiments, we set the maximum number of attempts $\bar{d} = 15$. We also set $\gamma = 10$ and select those behaviors randomly, weighted by their probabilities λ_k.

B.2. GPT-Based Column-Generation Algorithm: Computation of Reduced Costs. We now describe in detail how the reduced costs of new columns in the growing preference tree are computed. Let us assume that we have computed the reduced costs $rc_{\sigma_c} = rc_{P(\sigma_c)} + rc_{I(\sigma_c)}$ of behavior σ_c, where $rc_{P(\sigma_c)}$ is the part of the reduced costs coming from the strictly ranked set and $rc_{I(\sigma_c)}$ accounts for the part from the indifference set.

Let us also assume that, while having computed rc_{σ_c}, we have also generated a subset $\mathcal{M}_1 \subset \mathcal{M}$ that contains all assortments in which a strictly ranked product from $P(\sigma_c)$ was selected. For each assortment $S_m \notin \mathcal{M}_1$, we also assume that its specific contribution $rc_{I(\sigma_c)} = \sum_{m:S_m \notin \mathcal{M}_1} rc_{I(\sigma_c),m}$ has already been computed.

To compute the reduced costs of a subbehavior of σ_c that additionally ranks item $i \in I(\sigma_c)$, we have to adjust the current reduced cost rc_{σ_c} by distinguishing the following three cases for each of the M assortments:

1. If $S_m \in \mathcal{M}_1$, then adding a strictly ranked item i will not impact the choice (and therefore not the reduced cost).
2. If $S_m \notin \mathcal{M}_1$, then the indifference set was explaining sales in σ_c and contributing to its reduced cost. If $i \in S_m$, it will be selected. One therefore has to add its dual value $\alpha_{i,m}$ and subtract the entire reduced cost $rc_{I(\sigma_c),m}$ stemming from the indifference set.
3. If $S_m \notin \mathcal{M}_1$, but $i \notin S_m$, then the remaining items in the indifference set will contribute to the reduced costs. However, given that i is not in the assortment, it did not contribute to the reduced cost of σ_c and will not contribute to the reduced cost of the new subbehavior. Therefore, no changes to the reduced cost have to be made.

Given that cases 1 and 3 do not result in changes, and case 2 is performed in constant time, the total complexity to compute the reduced cost for a subbehavior is $O(M)$.

Endnotes

[1] The data are aggregated over time and customers, that is, we assume that any information about the customers (ID, features) and the exact purchase moment is either unavailable or ignored to estimate the models.

[2] Instead of using the Gurobi solver, we here used CPLEX 12.7.1 with a Python interface, since it turned out to build the model faster given the large number (sometimes millions) of columns.

[3] To be precise, after selecting the 100 preference sequences with highest probabilities λ, the restricted master problem has been resolved for this subset to estimate the new probabilities.

[4] As an example, consider two assortments $S_1 = \{1, 2, 3\}$ and $S_2 = \{1, 2, 4\}$. We may have observed only sales of product 1 in assortment S_1 and only sales of product 2 in assortment S_2. In this case, a ranking-based choice model cannot perfectly fit the sales transactions for both assortments.

References

Aouad A, Farias V, Levi R (2015) Assortment optimization under consider-then-choose choice models. Preprint, submitted June 15, http://dx.doi.org/10.2139/ssrn.2618823.

Aouad A, Farias V, Levi R, Segev D (2018) The approximability of assortment optimization under ranking preferences. *Oper. Res.* 66(6):1661–1669.

Arrow KJ (1951) *Social Choice and Individual Values* (John Wiley & Sons, Hoboken, NJ).

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

Ben-Akiva ME (1973) Structure of travel demand models. Unpublished doctoral dissertation, Massachusetts Institute of Technology, Cambridge.

Berbeglia G, Joret G (2020) Assortment optimisation under a general discrete choice model: A tight analysis of revenue-ordered assortments. *Algorithmica* 82:681–720.

Bertsimas D, Mišic V (2016) *Data-driven assortment optimization*. Technical report, Massachusetts Institute of Technology, Cambridge.

Bertsimas D, Mišic V (2019) Exact first-choice product line optimization. *Oper. Res.* 67(3):651–670.

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

Farias VF, Jagabathula S, Shah D (2013) A nonparametric approach to modeling choice with limited data. *Management Sci.* 59(2):305–322.

Griva I, Nash SG, Sofer A (2009) *Linear and Nonlinear Optimization* (SIAM, Philadelphia).
Haensel A, Koole G (2011) Estimating unconstrained demand rate functions using customer choice sets. *J. Revenue Pricing Management* 10(5):438–454.
Ho-Nguyen N, Kilinc-Karzan F (2017) Dynamic data-driven estimation of non-parametric choice models. Preprint, submitted February 19, https://arxiv.org/abs/1702.05702.
Honhon D, Jonnalagedda S, Pan XA (2012) Optimal algorithms for assortment selection under ranking-based consumer choice models. *Manufacturing Service Oper. Management* 14(2):279–289.
Huang Z (1997) Clustering large data sets with mixed numeric and categorical values. *1st Pacific-Asia Conf. Knowledge Discovery Data Mining*, (Singapore) 21–34.
Jagabathula S (2011) Nonparametric choice modeling: Applications to operations management. Unpublished doctoral dissertation, Massachusetts Institute of Technology, Cambridge.
Jagabathula S (2014) Assortment optimization under general choice. Preprint, submitted October 21, http://dx.doi.org/10.2139/ssrn.2512831.
Jagabathula S, Rusmevichientong P (2016) A nonparametric joint assortment and price choice model. *Management Sci.* 63(9):3128–3145.
Jagabathula S, Rusmevichientong P (2019) The limit of rationality in choice modeling: Formulation, computation, and implications. *Management Sci.* 65(5):2196–2215.
Jagabathula S, Vulcano G (2018) A partial-order-based model to estimate individual preferences using panel data. *Management Sci.* 64(4):1609–1628.
JDA Labs (2017) JDA Labs. Accessed September 7, 2017, https://jda.com/innovation/jda-labs.
Kök AG, Fisher ML, Vaidyanathan R (2008) Assortment planning: Review of literature and industry practice. Narendra A, Smith SA, eds. *Retail Supply Chain Management* (Springer, Boston), 99–153.
Lebanon G, Mao Y (2008) Non-parametric modeling of partially ranked data. *J. Machine Learning Res.* 9:2401–2429.
Mahajan S, Van Ryzin GJ (1999) Retail inventories and consumer choice. Sridhar T, Ganeshan R, Magazine M, eds. *Quantitative Models for Supply Chain Management* (Springer, Boston), 491–551.
McBride RD, Zufryden FS (1988) An integer programming approach to the optimal product line selection problem. *Marketing Sci.* 7(2):126–140.
Palmer H (2017). Large-scale assortment optimization. Unpublished master's thesis, Polytechnique Montréal, Montreal.
Van Ryzin G, Vulcano G (2015) A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Sci.* 61(2):281–300.
Vulcano G, Van Ryzin G (2017) Technical note - An expectation-maximization method to estimate a rank-based choice model of demand. *Oper. Res.* 65(2):396–407.