Learning to Rank an Assortment of Products

Kris J. Ferreira
Assistant Professor, Harvard Business School, Boston, MA 02163, kferreira@hbs.edu

Sunanda Parthasarathy
Wayfair, Boston, MA 02116, sparthasarathy@wayfair.com

Shreyas Sekar
Harvard Business School, Boston, MA 02163, ssekar@hbs.edu

We consider the product ranking challenge that online retailers face when their customers typically behave as "window shoppers": they form an impression of the assortment after browsing products ranked in the initial positions and then decide whether to continue browsing. We design online learning algorithms for product ranking that maximize the number of customers who engage with the site. Customers' product preferences and attention spans are correlated and unknown to the retailer; furthermore, the retailer cannot exploit similarities across products owing to the presence of subjective, stylistic elements. We develop a class of online learning-then-earning algorithms that prescribe a ranking to offer each customer, learning from preceding customers' clickstream data to offer better rankings to subsequent customers. Our algorithms balance product popularity with *diversity*: the notion of appealing to a large variety of heterogeneous customers. We prove that our learning algorithms converge to a ranking that matches the best-known approximation factors for the offline, complete information setting. Finally, we partner with Wayfair - a multi-billion dollar home goods online retailer - to estimate the impact of our algorithm in practice via simulations using actual clickstream data, and we find that our algorithm yields a significant increase (5-30%) in the number of customers that engage with the site.

Key words: Online Learning, Product Ranking, Assortment Optimization, E-Commerce

1. Introduction

Over the last decade, online retail has experienced significant growth and is becoming a larger portion of the retail industry (Economist (2017)). Online retailers[1] face many new operational opportunities to drive customer engagement and revenue growth. One such opportunity is the decision of how to rank products on their website. It is well established in the academic literature and known by retailers that more customers view products ranked in top positions compared to products ranked towards the bottom of the page where customers must scroll down to view. Furthermore, the product ranking can have a causal effect on the number of products searched (see, e.g., Ursu (2018)). Thus, the ranking of products is an important decision since this allows the

[1] We will refer to any retailer with an online presence, regardless of whether they also have brick-and-mortar stores, as an "online retailer". When the context is clear, we will refer to online retailers as simply "retailers" for brevity.

retailer to control which products are seen by most customers, impacting the customers' browsing behavior, purchase decisions, and long-term interest in the retailer.

In this paper, we study the ranking challenge in a particular setting that, to the best of our knowledge, is yet to be studied in the academic literature. We consider a retailer selling products that have a subjective style, and which are difficult to fully characterize by their attributes. The typical customer (she) does not have a good idea of the assortment or style of products offered, and may not have a particular product in mind that she is looking for. She forms an opinion about the assortment and style after browsing products in the initial ranks; we define the customer's "attention window" as the number of products she views to help form her opinion, which may differ by customer and is unknown to the retailer. Many retailers face such customers. In general, retailers who change their assortment of products frequently relative to the frequency of a customer's visit likely face such customer behavior.

We assume that each customer browses products in the initial ranks up to her attention window, and if she sees a product that she likes, she is "hooked" and continues browsing in the hopes of finding additional products that she likes. If she does not see a product she likes within her attention window, she believes she may not find anything she likes in the assortment and therefore leaves the site[2] early. Customers click on products that they like in order to learn details about the product (e.g., sizes available, dimensions, material, etc.). In contrast to other literature on product ranking, we do not make the assumption that customers purchase at most one product; instead, we are motivated by the setting where customers may like, and perhaps purchase, multiple products. The retailer observes the set of products that customers like via clicks and dynamically changes the product ranking over time to learn the ranking that maximizes the number of hooked customers, or equivalently, minimizes the number of customers who do not see anything they like and abandon the site prematurely.

Customers in our setting are analogous to "window shoppers" in brick-and-mortar retail. In a physical mall, window shoppers stroll through the mall looking at products in the retailers' display cases to learn about the assortment of products the retailer is offering. If they see a product they like, they are hooked and enter the store to browse more products; otherwise, they avoid the store altogether. The online retailer's goal of maximizing the number of hooked customers - the number of customers who engage with the site - is analogous to the brick-and-mortar retailer's goal of maximizing store traffic, a key customer engagement metric for many retailers. Although the objective of profit maximization is more commonly studied in the academic literature, many

[2] "Site" refers to the page(s) that display products in an assortment, which may include only a subset of products the retailer offers. For example, a flash sales retailer may offer several events at a time; here, "site" refers to an event's page(s).

retailers in competitive markets have the primary objective of maximizing market share and use such customer engagement metrics to do so, including the retailer we partnered with for this research, Wayfair (www.wayfair.com) - a multi-billion dollar home goods online retailer.

In order to maximize the number of hooked customers, the retailer must consider both product popularity and diversity in its ranking decisions. Popularity is an obvious consideration: by including a very popular product in a top rank, the retailer can hook many customers. Interestingly, ranking by popularity is one of the most common ranking approaches studied in the academic literature and implemented in practice. However, *diversity* (or variety) of products is also an important notion; here, we consider two products as being diverse if the set of customers that each hooks has little or no overlap. Two products that are very popular may hook the same customers. Incorporating diversity allows the retailer to hook a broader base of customers.

Our main contributions in this paper are three-fold. First, we introduce and study the product ranking challenge for retailers who often face "window shopping" consumer behavior. Our model highlights the importance of both product popularity and diversity considerations when making ranking decisions. This type of consumer shopping behavior is well recognized in the consumer psychology literature as "hedonic browsing" (see, e.g., Moe (2003)), which cites the importance of offering such consumers a diverse set of products, and we are the first to consider the ranking decision for retailers often facing such customers.

Second, we develop a class of prescriptive, dynamic ranking algorithms that balance popularity and diversity to learn the ranking that maximizes the number of hooked customers. In the (unrealistic) complete information setting - where the retailer knows the joint distribution of which products customers like along with their attention windows - we relate a special case of our model to the maximum coverage problem and extend its well-known greedy algorithm to our more general model. Then, when the retailer has no prior knowledge of product interests and attention windows, we develop algorithms that dynamically learn the greedy ranking from the complete information setting by offering different rankings over time. Our algorithms circumvent having to learn the joint distribution of products that customers like and their attention windows, and instead learn the optimal ranking directly. The algorithms that we propose vary based on their tradeoff of optimality gap vs. speed of learning and can thus be applied to retailers of varying sizes.

For our third contribution, we partner with Wayfair to develop a realistic model and implementable algorithm, and ultimately estimate its impact via simulations using actual clickstream data. Across six different product assortments, our algorithm hooks an average of 5-30% more customers than Wayfair's static, popularity-based ranking. Furthermore, our algorithm hooks 89-95% of the total number of customers hooked by an (omniscient) offline benchmark. These simulations using real clickstream data illustrate that our algorithmic contributions can make a significant impact in practice.

	Assumes customers buy at most one product	Models as optimal stopping problem (OSP) or MNL	Assumes independent utility / interest across products	Assumes *a priori* full knowledge of products' intrinsic utilities and distribution of random utilities	Descriptive (D) vs. Prescriptive (P)
Aouad and Segev (2015)	✓	MNL	✓		P
Chen and Yao (2016)	✓	OSP	✓	✓	D
Chu et al. (2017)	✓	OSP	✓	✓	P
Dzyabura and Hauser (2016)	✓	OSP			D
Gallego et al. (2016)	✓	MNL			P
Ghose et al. (2012)	✓	MNL	✓	✓	P
Ghose et al. (2014)	✓				P
Golrezaei et al. (2018)	✓	OSP	✓	✓	P
Lei et al. (2018)	✓	MNL	✓	✓	P
Kim et al. (2010)	✓	OSP	✓	✓	D
Kim et al. (2016)	✓	OSP	✓	✓	D
Koulayev (2014)	✓	OSP		✓	D
Ursu (2018)	✓	OSP	✓	✓	D

Table 1 Modeling Assumptions and Analysis Techniques in Product Ranking Literature

1.1. Related Literature

Our work is related to three different streams of literature: product ranking, learning algorithms, and submodular optimization.

Connections to Product Ranking Product ranking has been well-studied in the academic literature, although there are key differences in modeling assumptions and analysis between this stream of work and ours. We compare the major aspects of the most relevant product ranking papers in Table 1 and speak to their implications below.

First, product ranking papers assume that customers are shopping to fill a particular need and buy at most one product. Making this assumption and considering sequential, costly search, many of these papers build upon the seminal paper Weitzman (1979) and model the consumer's shopping behavior as an optimal stopping problem, trading off the cost of searching one more product vs. its expected utility. Most other papers use a multinomial logit (MNL) model. In contrast, we do not assume that customers buy at most one product, although this behavior is possible in our model; rather, we are motivated by settings where consumers are window shoppers and browse products looking for any they might like. Due to this difference, modeling our problem using the optimal stopping framework of Weitzman (1979) or an MNL model is not appropriate for our setting.

This type of shopping behavior is well recognized in consumer psychology literature as "hedonic browsing" (see, e.g., Moe (2003)). Browsing has been further described in the consumer behavior

literature as a screening activity that is independent of any purchase needs (Bloch et al. (1989)); through browsing, consumers purchase products that they would not otherwise have considered (Rowley (2002)). Research has shown that first impressions matter; for example, Dawson and Kim (2010) suggests that if a shopper is not hooked within 30 seconds of browsing the site, she will leave. Cho and Youn-Kyung (2012) suggests that shoppers form a quick judgment of the assortment after viewing a few products, which may or may not warrant further exploration. We capture this initial screening behavior and time spent on the site via the customer's attention window.

Second, most product ranking papers assume for tractability that customer interest in any two products is independent, which may be unrealistic: given that a customer likes product A, she is more likely to also like a similar product B. We do not make this independence assumption, and in fact, our learning algorithm converges to the well-studied popularity ranking when independence holds. A notable exception to this assumption is Ghose et al. (2014), which instead of explicitly modeling consumer utility, assumes a structural model (hierarchical Bayes) based on well-defined, observed attributes and estimates parameters of the model with historical data. In their structural model, they include the product ranking effect in both a linear and quadratic term. We do not assume such a functional form for the product ranking effect, and we do not require that attribute data exists or is available for search.

Third, most other papers assume for tractability that the intrinsic utilities of products, as well as the distribution of random utilities, are known to the retailer and customers before customers actually view the products, which may be unrealistic. Thus, we do not make this assumption and instead use a basic model of customer learning where the customer browses products to learn about the assortment and style of products offered. Dzyabura and Hauser (2016) also incorporate an aspect of customer learning in their model, where the customer is assumed to know attribute values of all products offered, but learns her preferences for those attributes as she searches. Although Aouad and Segev (2015) and Gallego et al. (2016) do not consider customer learning, they do assume exogenous attention windows as do we. Interestingly, even with critical differences in modeling assumptions, Gallego et al. (2016) proposes an algorithm that is similar in spirit to our offline algorithm; that said, their analysis does not extend to our model.

Finally, many of these papers are descriptive in nature, evaluating the impact of a static ranking on consumer behavior. In contrast, our work is prescriptive, in that we aim to find a ranking that maximizes a retailer's objective. Beyond differences in modeling assumptions, the other papers that are prescriptive in nature consider offline algorithms that produce a static ranking, whereas ours presents an online learning algorithm that dynamically changes rankings to learn the best rank over time. One notable exception is Lei et al. (2018), who dynamically change product rank, price, and fulfillment decisions to maximize the profitability of selling a finite inventory over a selling

season; in contrast, the dynamic nature of our algorithms stem from the necessity of *learning* the best rank given *unknown* model parameters.

Connections to Learning Algorithms Motivated by applications in information retrieval, web search, and recommendation systems, there are many recent papers on learning algorithms for information ranking. Much of this work uses clickstream data to infer optimal static rankings, so we only highlight the recent work pertaining to online learning algorithms. Online learning-to-rank can be classified under two broad categories depending on the model of user behavior (Zoghi et al. 2017). In the cascade model (Kveton et al. 2015) there is only one user type with a fixed but unknown set of click probabilities for each item and a known attention window (k), and the objective is - analogous to ours - to select k items to minimize the probability of customer abandonment. However, such a model does not account for varying and unknown attention windows that lead to the behavioral effect that fewer customers view products at lower ranks. Therefore, the cascade model essentially captures a subset selection problem since altering the order in which the k items are displayed does not affect the reward.

In the position-based model, as in sponsored search, the customer's utility or click probability for an item is the product of a position-specific and item-specific component (Lagrée et al. 2016). On the contrary, we do not assume any independence between the customer's product interest probabilities and their attention windows. Recent research has attempted to bridge the gap between the two models by proposing more general approaches (Zoghi et al. 2017). In all of these works, the optimal ranking is simply the 'popular ranking' obtained by sorting the items in descending order of click probabilities.

A parallel stream of literature has focused on the problem of designing online algorithms for the selection of diverse sets of items from a large universe (Radlinski et al. 2008, Raman et al. 2012). Although these works allow for heterogeneous customer types, all customers have the same attention window and hence, the problem is more analogous to submodular selection than ranking. Our work also bears superficial similarities to papers on combinatorial bandits such as (Chen et al. 2016) although our ranking problem cannot be captured by the model in that work since the overall reward cannot be independently decomposed into its components due to different products.

Along these lines, our work is also related to the relatively small literature on learning optimal assortments (see, e.g., Agrawal et al. (2016), Bernstein et al. (2017), and Ulu et al. (2012)). Similar to ours, papers in this area propose offering dynamic assortments over time to learn an optimal assortment. Key differences include (i) these papers either assume customers buy at most one product or that product interest is independent for each product, and (ii) customers observe all products in the assortment, i.e. the attention window is known and identical for all customers. It

is worth noting that there is a larger literature on offline, static assortment optimization without learning, some of which results in diverse assortments (see, e.g., Li et al. (2015)); the same key differences exist.

Connections to Submodular Optimization Although our model bears strong similarities to the well-studied stochastic submodular optimization problem (Asadpour and Nazerzadeh 2015), our results and techniques do not follow directly from that literature. This is because submodular maximization problems are concerned with selecting a subset of items from a known universe whereas our problem involves both selection and ranking of the selected products. In that sense, our model strictly generalizes popularly studied online submodular maximization problems such as the maximum coverage problem (Radlinski et al. 2008). Moreover, while the greedy algorithm for the maximum coverage problem provides a natural intuition on how to rank the selected products, in the online learning setting, this could lead to undesirable sample complexity. To address this concern, we draw inspiration from (Badanidiyuru and Vondrák 2014) and develop a threshold-based algorithm that is particularly well-suited for the ranking problem and show that it performs well both theoretically and in practice.

2. Model

We consider an online retailer who offers a set of n products for sale during a selling season. For convenience, the products are labeled as $[n] = \{1, 2, \ldots, n\}$. A set $\mathcal{T}$ of customers ($|\mathcal{T}| = T$) arrive sequentially and the retailer selects a ranking of products to offer to each customer. Formally, a ranking is a permutation $\pi : [n] \to [n]$ that maps each product i to a position $1 \leqslant \pi(i) \leqslant n$. We use $\pi^{-1}(j) \in [n]$ to denote the product that is ranked in the j^{th} position under ranking π. Clearly, if $\pi^{-1}(j) = i$, then $\pi(i) = j$ by definition. We will use π_t to denote the ranking of products displayed to customer t and leave off the subscript when the context is clear. The assortment of products is assumed to be fixed beforehand as are their prices, which reflects the fact that product ranking decisions are typically made after assortment and pricing decisions in practice. Furthermore, we assume that inventory is plentiful such that no products sell out before the end of the season.

Customers *browse* products sequentially and *click* on products that they like in order to learn more about the product (e.g., its size/dimensions, material, availability, etc.), and retailers observe these clicks to help inform their subsequent ranking decisions. Each customer $t \in \mathcal{T}$ is characterized by $(\mathbf{p}_t, k_t) \sim_{iid} \mathcal{D}$, where we refer to $\mathbf{p}_t = (p_{1t}, p_{2t}, ..., p_{nt})$ as her vector of *click probabilities*, and we refer to $k_t \in [n]$ as her *attention window*. Customer t browses all products within her attention window (positions 1 to k_t) in order to learn about the assortment and general style of products offered. With probability p_{it}, customer t likes product i and - conditional on browsing (seeing)

product i - she clicks on it to learn more about the product[3]. If the customer does not like any of the products she sees within her attention window, she stops browsing and exits the site; otherwise, she continues browsing[4]. The retailer has no prior knowledge of $(\mathbf{p}_t, k_t)$ or $\mathcal{D}$, and only observes customer clicks. Note that our model allows for customer heterogeneity as the click probability vector and attention window can be different across customers; in addition, customers' preferences and attention windows may not be independent.

For each product i and customer t such that $\pi(i) \leqslant k_t$, we define the Bernoulli random variable $C_{it}(\pi)$ as $C_{it}(\pi) = 1$ if the customer t - conditional upon her type $(\mathbf{p}_t, k_t)$ - clicks on product i under ranking π and $C_{it}(\pi) = 0$ if not. For notational convenience, when $\pi(i) > k_t$, we define $C_{it}(\pi) = 0$. We use $\overline{C}_{it}(\pi)$ as shorthand to denote $1 - C_{it}(\pi)\ \forall i, t$. By indexing customer t, we implicitly assume that the random variable is conditional on customer t's type, $(\mathbf{p}_t, k_t)$, and will use this shorter notation throughout the paper. Note that $\Pr(C_{it}(\pi) = 1) = p_{it}$ when $\pi(i) \leqslant k_t$, since customer t is guaranteed to have browsed product i if it is displayed within her attention window. We assume that $C_{it}(\pi) \perp\!\!\!\perp C_{jt}(\pi)\ \forall i \neq j, \pi(i) \leqslant k_t, \pi(j) \leqslant k_t$; note that this independence is conditional on customer t's type, $(\mathbf{p}_t, k_t)$, in contrast to much of the work cited in Section 1.1.

We further define the event that customer t is "hooked" as

$$H_t(\pi) = \begin{cases} 1, & \text{if } \sum_{i \in [n]} C_{it}(\pi) \geqslant 1 \\ 0, & \text{if } \sum_{i \in [n]} C_{it}(\pi) = 0 \end{cases};$$

in other words, customer t is hooked if she clicks on at least one product in the assortment under ranking π. Since $C_{it}(\pi) = 0$ when $\pi(i) > k_t$, a necessary condition for the customer to be hooked is that she clicks on at least one product within her attention window. Therefore, $H_t(\pi) = 1$ if and only if $\sum_{j \in [k_t]} C_{\pi^{-1}(j)t}(\pi) \geqslant 1$. Moreover, the probability of customer t being hooked is $1 - \prod_{j \in [k_t]} (1 - p_{\pi^{-1}(j)t})$. Note that as the number of products the customer is likely to click on within her attention window increases, the more likely she is to be hooked.

The problem faced by the retailer is to design a non-anticipatory algorithm that selects a ranking π_t for each arriving customer $t \in \mathcal{T}$ in order to maximize the total number of hooked customers over the season, or equivalently, to maximize the total number of customers who engage with the site by clicking on at least one product:

$$\max_{\pi_1, \pi_2, \ldots, \pi_T} \mathrm{E}\Big[\sum_{t \in \mathcal{T}} H_t(\pi_t)\Big]. \tag{1}$$

[3] For ease of exposition, we equate the customer "liking" a product to the customer "clicking on" a product if she sees it, although our algorithm and results remain unchanged if we instead assume that a customer clicks on a product she likes with probability $\lambda \in [0, 1]$.

[4] The customer may continue to browse all or only a subset of the remaining products.

Our objective can be interpreted as "minimizing abandonment", a common objective in service systems, where in our case, abandonment refers to a customer not finding any products she likes within her attention window and thus leaving the site early before being "served" a product she likes (if such a product exists). Although the objective of profit maximization is more commonly studied in the academic literature, many retailers in competitive markets (including our partner retailer, Wayfair) and/or retailers targeting window shoppers (e.g., flash sales sites) have the primary objective to maximize market share and use such a customer engagement metric to do so.

To summarize the sequence of events in our model, customers arrive sequentially and for each customer $t \in \mathcal{T}$,

1. The retailer selects and offers ranking π_t.

2. The customer browses all products within her attention window (positions 1 to k_t) and clicks on products that she likes.

3. If the customer does not like any of the products within her attention window, she leaves the site without browsing any additional products. Otherwise, she is hooked and continues browsing.

4. The retailer observes all customer clicks (but not $(\mathbf{p_t}, k_t)$) and can use this information when choosing rankings to offer new customers.

We will refer to the retailer's ranking problem under the consumer behavior model described above as the *online assortment ranking* problem (OnAR). We point out that "online" refers to two characteristics: first, "online" refers to the retailer's ability to observe clicks in real time as customers engage with the site. Second, "online" also refers to the online retail industry, since many online retailers face this ranking challenge and have the ability to change rankings dynamically over time. In this work, we focus on a common special class of algorithms for OnAR known as learning-then-earning algorithms - these methods focus on rapidly converging to a (near-)optimal ranking that is then fixed for the remaining customers. Such algorithms are often preferable in online retail when these sites lack accurate foreknowledge of the number of customers $|\mathcal{T}|$ who visit during the selling period.

Before presenting our class of dynamic ranking algorithms for the online assortment ranking problem in Section 4, we first study the *offline assortment ranking* problem (OffAR) in Section 3 where we make the (unrealistic) assumption that the retailer knows the distribution $\mathcal{D}$ at the start of the season. Since $\mathcal{D}$ is known ahead of time, no learning is necessary and thus the term "offline" is used, as a single, static ranking can be chosen at the beginning of the season and offered to all customers.

3. Offline Assortment Ranking

In this section, we consider the *offline* version of the assortment ranking problem (OffAR), where the retailer knows the distribution $\mathcal{D}$ from which customers' preferences and attention windows

are drawn at the beginning of the season. However, the retailer does not know the parameters $(\mathbf{p}_t, k_t)$ before the arrival of each customer $t \in \mathcal{T}$. Since $\mathcal{D}$ is known ahead of time, no learning is necessary and a single, static ranking can be chosen at the beginning of the season and offered to all customers. Although our primary interest lies in the online assortment ranking problem where the retailer must learn the customers' preferences, the methods developed in this section will provide a clear understanding of the core techniques involved in the more sophisticated learning algorithms that we will present in Section 4, and will also serve as a benchmark for which to compare our online ranking algorithms in the simulations in Section 5.

Our first result shows that even when the retailer knows $\mathcal{D}$ *a priori*, the assortment ranking problem is NP-Hard.

PROPOSITION 1. *The offline assortment ranking problem is NP-Hard.*

The proof of this result utilizes an important special case of OffAR where $p_{it} \in \{0, 1\}$ and $k_t = k \ \forall t \in \mathcal{T}, i \in [n]$. This special case admits the following intuitive interpretation: each customer $t \in \mathcal{T}$ is endowed with a set of preferred products $S_t \subseteq [n]$ and attention window $1 \leqslant k \leqslant n$ and clicks on any product $i \in S_t$ that is present within ranks $[1, k]$. The retailer's goal is to select a ranking π to maximize the expected number of customers who have at least one product from their preferred subset inside their attention window. Upon close examination, this special case of OffAR is a stochastic version of the *maximum coverage problem*, which is a well-studied problem in the combinatorial optimization literature. The deterministic maximum coverage problem is known to be NP-Hard, and given that OffAR generalizes the maximum coverage problem, it must also be NP-Hard. The formal proof of Proposition 1 involves mapping the *deterministic* maximum coverage problem to the *stochastic* offline assortment ranking problem. Proofs of all analytical results are provided in Appendix A.

Given this NP-Hardness result, it is difficult to find an optimal solution to OffAR for a reasonable problem size. Thus, a standard optimization approach is to consider approximation algorithms that come close to the optimal solution. The best approximation algorithm for the maximum coverage problem - a special case of OffAR - is a greedy algorithm that achieves a $(1 - \frac{1}{e})$ approximation factor (see Theorem 3.8 in Hochbaum (1996)). Motivated by this greedy algorithm, we formally present our algorithm for the more general offline assortment ranking problem in Algorithm 1, which we will refer to as the Greedy Algorithm for OffAR (or simply Greedy Algorithm when the context is clear).

Algorithm 1 sequentially fixes products in ranks $\{1, 2, \ldots, n\}$. Assuming that ranks 1 through $r-1$ have been fixed, in iteration r, the algorithm considers all the remaining (unranked) products for position r. The product i with the maximum marginal benefit (Δ_{ir}) - i.e., that increases the

Algorithm 1: Greedy Algorithm for OffAR

Initialize null ranking $\pi^g(i) = \emptyset$ for all $1 \leqslant i \leqslant n$;

Initialize unranked products $U = [n]$;

for $r = 1$ *to* n **do**

  for $i \in U$ **do**

    Let $\tilde{\pi}^g \leftarrow \pi^g$;

    $\tilde{\pi}^g(i) \leftarrow r$;

    $\Delta_{ir} = E_{t\sim\mathcal{D}}[H_t(\tilde{\pi}^g) - H_t(\pi^g)]$;

  end

  Let $i^*_r = \arg\max_{i\in U} \Delta_{ir}$;

  Set $\pi^g(i^*_r) \leftarrow r$;

  $U \leftarrow U \setminus i^*_r$.

end

probability of hooking an incoming customer by the largest amount - is fixed at rank r and the algorithm continues on to the next rank. The marginal benefit can also be interpreted as the probability that the customer clicks on the product at rank r but not on any products in ranks $\{1, 2, \ldots, r-1\}$. Note that Δ_{ir} depends on three key factors: (1) the probability that a customer clicks on product i, (2) the probability that a customer does not click on products in ranks $\{1, 2, \ldots, r-1\}$, and (3) whether or not rank r is within the customer's attention window. The combination of the first two factors can be interpreted as the balance of *popularity* and *diversity*. On the one hand, the retailer wants to offer popular products that are likely to be clicked on; on the other hand, the retailer wants to select diverse products to hook a more heterogeneous set of customers.

It is straightforward to see that Algorithm 1 reduces to the standard greedy algorithm for the maximum coverage problem for the special case of OffAR where $p_{it} \in \{0, 1\}$ and $k_t = k \; \forall t \in \mathcal{T}, i \in [n]$, which is known to provide a $(1 - \frac{1}{e})$-approximation factor to the optimal offline solution. The important yet subtle generalization in our algorithm can be seen in the definition of Δ_{ir}, which incorporates varying attention windows and probabilistic interests. Surprisingly, the following result shows that even for the more general offline assortment ranking problem, the Greedy Algorithm still achieves a $\frac{1}{2}$-approximation factor.

THEOREM 1. *The Greedy Algorithm for OffAR results in ranking π^g such that the probability that an incoming customer is hooked is at least one-half that of the optimal ranking π^*, i.e. Algorithm 1 achieves a $\frac{1}{2}$-approximation factor for OffAR.*

The proof involves characterizing the marginal benefit due to each product in π^* in terms of the product occupying the same rank under π^g. Informally, consider all the customer profiles (say,

Z) in the support of $\mathcal{D}$ that are first hooked by a certain product $i \in [n]$ under ranking π^* - i.e., its marginal benefit from customers who do not click on products at ranks smaller than $j = \pi^*(i)$. The central argument in the proof is that the fraction of customers hooked by product i under ranking π^* is no more than the sum of (a) the fraction of Z profiles that are hooked by the greedy algorithm π^g at ranks less than j and (b) the marginal benefit of the product at rank j under ranking π_g. Indeed, if this were not the case, one could argue that the greedy algorithm would have fixed product i at rank $\pi^*(i)$. Summing up this idea over all products gives us a factor of one-half since we may end up counting each customer profile twice. The challenge in formalizing this claim comes from the fact that the customers' clicks are governed by two sources of uncertainty - the set of preferred products and the attention window. Since the sources may be correlated, one cannot simply decompose them via typical independence arguments.

The following example shows the $\frac{1}{2}$-approximation factor in Theorem 1 is tight for Algorithm 1.

EXAMPLE 1. Consider an instance of OffAR with two products $\{1,2\}$. The distribution $\mathcal{D}$ is defined as follows:

- with probability $0.5+\epsilon$, customer t has $k_t = 2$ and $p_{1t} = 1$, $p_{2t} = 0$;
- with probability $0.5-\epsilon$, customer t has $k_t = 1$ and $p_{1t} = 0$, $p_{2t} = 1$.

The optimal ranking is $\pi^* = \{2,1\}$, which hooks all customers with probability one. On the other hand, $\pi^g = \{1,2\}$, which only hooks a customer with probability $0.5+\epsilon$. As $\epsilon \to 0$, the approximation factor for the Greedy Algorithm for OffAR approaches one half. □

Interestingly, although the $\frac{1}{2}$-approximation factor is tight for general OffAR instances, the following result shows that the Greedy Algorithm for OffAR can achieve a $(1-\frac{1}{e})$-approximation factor for a broad class of special cases to OffAR, including the maximum coverage problem.

THEOREM 2. *Consider the special case of OffAR where $\mathbf{p}_t$ and k_t are independent, referred to as OffAR with Independence. The Greedy Algorithm for OffAR achieves a $(1-\frac{1}{e})$-approximation factor for OffAR with Independence.*

The proof involves showing that for any given attention window k, the first k ranks in π^g are identical when (i) all customers are included when running the algorithm, and (ii) only customers with attention window k are included when running the algorithm. Using results from stochastic submodular optimization, we can show that this is a $(1-\frac{1}{e})$−approximation to the optimal ranking for customers with attention window k. Finally, we relate the hook probability of the optimal ranking for all customers to the optimal ranking for customers with attention window k for all $k \in [n]$.

Interestingly, for the special case of OffAR where $p_{it} \in \{0,1\}$ and allowing for dependent $\mathbf{p}_t$ and k_t, one can develop an alternative algorithm using LP-pipage rounding that also achieves a

$(1-\frac{1}{e})-$approximation factor (Asadpour (2019)). Intuitively, if $(\mathbf{p}_t, k_t)$ are correlated, there may be more opportunity to prioritize hooking different customer types due to varying k_t. Our algorithm does not account for such dependencies, whereas work by Asadpour (2019) does. Unfortunately, their algorithm does not extend well to the learning setting in OnAR. As we will soon see, this is a key benefit of Algorithm 1.

Given the prevalence of the popularity ranking - i.e., where products are ranked in decreasing order of their click probabilities - in both academia and practice, we emphasize the key distinguishing feature of our algorithm. Namely, for each rank, our Greedy Algorithm seeks to maximize the number of customers that click on the product located at that rank *and no preceding ranks*. Therefore, our approach balances both product popularity and diversity - whereas the popularity ranking only seeks to maximize the number of customers that click on that rank, regardless of whether the customer clicked on a preceding rank. The following example illustrates that the performance of the popularity ranking can be arbitrarily bad, highlighting the importance of incorporating diversity as a key criteria in the ranking algorithm.

EXAMPLE 2. Consider a distribution $\mathcal{D}$ composed of $k < \frac{n}{2}$ distinct customer types $\{z_1, z_2, \ldots, z_k\}$. An arriving customer belongs to type z_1 with probability $\frac{1}{k} + (k-1)\epsilon$ and all other types with an equal probability of $\frac{1}{k} - \epsilon$ for some arbitrarily small $\epsilon > 0$. The attention window is k for all types and the click probabilities are as defined below:

- *Type* z_1: $p_{it} = 1$ for $i = \{1, k+2, \ldots, 2k\}$ and $p_{it} = 0$ otherwise.
- *Type* z_i $(i \neq 1)$: $p_{it} = 1$ and $p_{i't} = 0$ for all $i' \neq i$.

We first argue that (one of) the optimal ranking(s) is π^* such that $\pi^*(i) = i$ for all $i \in [1, n]$. Indeed it is easy to see that every customer type has at least one product from their preferred set within their attention window (top k ranks) and therefore, the probability of getting hooked for any incoming customer is one. Furthermore, one can verify that the Greedy Algorithm would also result in a ranking π^g that hooks all customers.

Next we derive the popularity ranking, π^p. Products $i \in \{1, k+2, \ldots, 2k\}$ are preferred by customers of type z_1 and thus yield an expected click probability of $\frac{1}{k} + (k-1)\epsilon$. Each product in the set $i' = \{2, \ldots, k\}$ appeals to exactly one of the other customer types and thus its expected click probability is $\frac{1}{k} - \epsilon$. Therefore, π^p must contain all of the products in $\{1, k+2, \ldots, 2k\}$ in its first k positions, hooking only a $\frac{1}{k} + (k-1)\epsilon$ fraction of customers. As $\epsilon \to 0$ and n and k grow large, we observe that π^p can be arbitrarily worse than the optimal ranking. $\square$

4. Online Assortment Ranking

In this section, we study the more realistic *online* assortment ranking problem (OnAR), where the retailer does not know the distribution $\mathcal{D}$ from which customers' preferences and attention windows

are drawn. A naive approach for tackling this problem is to learn the complete distribution $\mathcal{D}$ by sampling the interest vectors $\mathbf{p}_t$ and attention window k_t from a sufficiently large number of customers. One could then apply (say) Algorithm 1 for OffAR to retrieve the same approximation factors as in Section 3. Unfortunately, such an approach is likely infeasible as the joint distribution $\mathcal{D}$ may have a large support, whereas the number of customers (available samples) may be relatively small. The problem is further compounded by the fact that the retailer's observations are *censored*: the retailer cannot observe the exact attention window k_t of customers nor infer the full vector $\mathbf{p}_t$.

Driven by these constraints, we develop learning-then-earning algorithms for OnAR that converge to the offline approximately optimal ranking *without learning the distribution* $\mathcal{D}$. Our learning-then-earning algorithms operate in two phases: first, in the learning phase, the algorithm offers different rankings to customers in order to learn the (approximately) best ranking. Then, in the earning phase, the algorithm offers this single, (approximately) best ranking to the remainder of customers. Although our proposed methods do sample customers to partially infer their preferences, the goal is to simply learn the (approximately) best ranking of products, not the entire distribution $\mathcal{D}$. As is typical for online learning algorithms, we measure performance in terms of two metrics: (i) an approximation factor with respect to the optimal ranking for the earning phase, and (ii) the length of the learning phase, i.e. the number of customers for which the algorithm learns before converging upon the final ranking. The first metric essentially measures how accurately the algorithm can learn, whereas the second metric measures the speed of learning.

In what follows, we present two learning algorithms for OnAR: the *Simple Learning Algorithm* and the *Threshold Acceptance Algorithm*. The former leverages multi-armed bandit (MAB) techniques to mimic the Greedy Algorithm for OffAR. These techniques come at the cost of a long learning phase, as the algorithm may require up to $\Theta(n^4)$ customers to guarantee convergence. In practice, requiring such a long learning phase is infeasible leading to the question of whether the retailer could sacrifice some performance in favor of speed. With this motivation, we develop the *Threshold Acceptance Algorithm* and prove that it achieves an approximation guarantee that is close to that of the Greedy Algorithm for OffAR but with a convergence rate that is an order of magnitude faster than that of the Simple Learning Algorithm.

4.1. Simple Learning Algorithm

We begin with a sequential learning paradigm that yields a class of algorithms for OnAR that all converge to a one-half approximation to the optimal ranking, yet differ by the length of the learning phase. In particular, the class of algorithms that we propose builds upon previous work in the area of multi-armed bandits for the problem of identifying the best arm among a finite set, i.e., the action yielding the highest expected reward. We propose a general procedure that transforms such

multi-armed bandit algorithms into algorithms that compute rankings for OnAR. We use a MAB algorithm as a black-box input and make several calls to this black-box to fix an approximately optimal product at each rank. Our reliance on black-box transformations using MAB algorithms is inspired by the large body of research in recent years that focuses on computing the best arm using the minimum number of samples (see Jamieson and Nowak (2014) for a survey); this version of the MAB problem is often referred to as MAB for Best Arm Identification, although we will simply use MAB for brevity. By using these techniques as black-box inputs to our ranking algorithm, we are able to leverage the insights generated in this literature without having to reinvent the wheel.

We first describe a few technical concepts required for stating our results. A multi-armed bandit problem comprises of n arms or actions such that upon 'pulling each arm' $i \in [n]$, the algorithm observes a reward of v_i that is drawn independently from some unknown distribution. There is a finite limit, $\tilde{T}$, on the total pulls that can be made. We focus on the objective of identifying the arm that maximizes the expected reward ($i^* = \arg\max_i E[v_i]$) while minimizing the number of arm pulls. As is common in online learning, we use the framework of (ϵ, δ)-PAC (probably approximately correct) algorithms that compute an approximately-optimal arm with high probability.

Definition 1. (Even-Dar et al. 2006) An algorithm is an (ϵ, δ)-PAC algorithm for the multi-armed bandit problem with sample complexity $\tilde{T}$ if with probability at least 1 - δ, it outputs an arm i such that $E[v_i] \geqslant \max_{i'} E[v_{i'}] - \epsilon$ and the number of times it pulls the arms is bounded by $\tilde{T}$.

Algorithm 2 formally presents our Simple Learning Algorithm for OnAR, which specifies rankings offered to customers during the learning phase. At the end of the learning phase, the ranking output by Algorithm 2 is offered to all remaining customers. The algorithm proceeds sequentially from ranks one through n and for each rank r, it makes one call to a black-box PAC algorithm (ALG) which samples at most $\tilde{T}$ customers to identify the product that achieves the maximum empirical marginal benefit - i.e., hooks the largest fraction of customers who did not click on any of the previously ranked products - when placed at rank r. The arms of ALG correspond to the unranked products and the reward on each arm is a Bernoulli random variable equal to 1 if the customer clicked on the product in rank r but not on any products in ranks $[1, r-1]$, and 0 otherwise. The Simple Learning Algorithm mimics the Greedy Algorithm for OffAR such that the product with the maximum marginal probability of hooking a customer is fixed at each rank with high probability.

The following result shows that the ranking produced by the Simple Learning Algorithm hooks at least (approximately) one-half of the customers hooked by the optimal ranking with high probability for general OnAR. Analogous to the offline setting, we define OnAR with Independence to be the special case of OnAR where p_t and k_t are independent; in this case, the one-half approximation factor can be further tightened to $1 - \frac{1}{e}$.

Algorithm 2: Simple (Black-Box) Learning Algorithm

Input: (ϵ', δ')-PAC algorithm ALG, Parameters $\epsilon, \delta > 0$;

Initialize null ranking $\tilde{\pi}(i) = \emptyset$ for all $1 \leqslant i \leqslant n$;

Initialize unranked products $U = [n]$;

for $r = 1$ *to* n **do**

For all $i \in U$, define $\tilde{\pi} \cup i$ to be a ranking identical to $\tilde{\pi}$ in positions $1, ..., r-1$ and with product i in rank r;

Run ALG on products U for $\tilde{T}$ customers, where $v_i = C_{it}(\tilde{\pi} \cup i) \prod_{j \in [r-1]} \overline{C}_{\tilde{\pi}^{-1}(j)t}(\tilde{\pi})$ for all $i \in U$;

Let $\tilde{i}_r$ be the product output by ALG;

Set $\tilde{\pi}(\tilde{i}_r) \leftarrow r$;

$U \leftarrow U \setminus \tilde{i}_r$.

end

THEOREM 3. *Given any (ϵ', δ')-PAC algorithm as a black box where $\epsilon' = \frac{2\epsilon}{n}$ and $\delta' = \frac{\delta}{n}$, the Simple Learning Algorithm returns a ranking $\tilde{\pi}$ such that with probability $(1-\delta)$:*

1. $\mathbb{E}[H_t(\tilde{\pi})] \geqslant \frac{1}{2}\mathbb{E}[H_t(\pi^*)] - \epsilon$ *for general OnAR.*
2. $\mathbb{E}[H_t(\tilde{\pi})] \geqslant (1 - \frac{1}{e})\mathbb{E}[H_t(\pi^*)] - \epsilon$ *for OnAR with Independence.*

Our final ranking $\tilde{\pi}$ is essentially an (ϵ, δ)-PAC approximation to the optimal ranking π^*. Since $\tilde{\pi}$ achieves an additive error of ϵ compared to the benchmark, the maximum permissable error for *ALG* at each individual rank r is $\frac{\epsilon}{n}$ (multiplied by a factor of two); similarly, for each rank r, the product designated at that rank via *ALG* is suboptimal with probability at most $\frac{\delta}{n}$. Given the lower bounds and the hardness results for OffAR in Section 3, it is not possible for any online learning algorithm to identify the optimal ranking using a polynomial number of queries.

The Simple Learning Algorithm makes exactly n black-box calls to the (ϵ', δ')-PAC algorithm. This allows us to bound the length of the learning phase as follows.

COROLLARY 1. *Given any (ϵ', δ')-PAC algorithm as a black box, the Simple Learning Algorithm returns a ranking $\tilde{\pi}$ after at most $n\tilde{T}$ customers.*

Due to the black-box nature of Algorithm 2, one could simply substitute any of numerous algorithms (Even-Dar et al. 2006, Jamieson and Nowak 2014, Kalyanakrishnan et al. 2012) for best-arm identification to develop a corresponding methodology for ranking. As mentioned previously, all of these algorithms converge to a ranking with the same approximation guarantee. However, the total number of customers they require may depend on different parameters and hence, the choice of the best algorithm may be instance-specific. The following corollary highlights the number of

customers used by our algorithm for two specific instantiations of the black-box (ϵ', δ')-PAC algorithm. In particular, we consider the *Exhaustive Sampling Algorithm* (Even-Dar et al. 2006), where each product is sampled for $L = \frac{4}{\epsilon^2}\log(\frac{2n}{\delta})$ times and the product yielding the largest number of hooked customers is returned; note that $\tilde{T} = nL$ in this case. We also consider the *LUCB Algorithm* (Kalyanakrishnan et al. 2012), a variant of the popular upper confidence bound (UCB) algorithm.

COROLLARY 2. *The Simple Learning Algorithm returns a ranking $\tilde{\pi}$ with approximation factors as specified in Theorem 3 with probability $(1-\delta)$ and length of learning phase as described below:*

1. $\Theta\big(\frac{n^4}{\epsilon^2}\log(\frac{n^2}{\delta})\big)$ *when the Exhaustive Sampling Algorithm is used as the black box.*
2. $\Theta\big(\frac{n^2}{\Delta_*^2}\log(\frac{n^2}{\Delta_*^2\delta})\big)$ *when the LUCB Algorithm is used as the black box, where Δ_* is a measure of the minimum gap between the rewards of the highest and second-best arms, where the minima is taken over all ranks $r \in [n]$*[5].

To summarize, the Simple Learning Algorithm balances popularity and diversity and (approximately) recovers the performance guarantees achieved in the offline setting for the earning phase. As we show in Corollary 2, the algorithm may require as many as $\Theta(n^4)$ customers in the learning phase before it converges to an approximately-optimal ranking to offer to customers in the earning phase. In practice, requiring such a long learning phase is often infeasible; thus, the next subsection is devoted to modifying the Simple Learning Algorithm to decrease the length of the learning phase at a slight loss of optimality in the earning phase.

4.2. Threshold Acceptance Algorithm

We propose a novel method that we term the *Threshold Acceptance Algorithm*, which achieves a significant speed-up in convergence time at the cost of only a small loss in performance. The approach is fully described in Algorithm 3 but its core idea is rather simple: *instead of exhaustively trying all products at a given rank in order to identify the best one, the algorithm simply selects a product that is 'good enough'*. Although this appears to be a simple tweak, the convergence time is an order of magnitude faster.

At each stage of the algorithm, we maintain a threshold τ, which is monotonically decreasing. The algorithm terminates when the threshold goes below a pre-specified value $\tau^{\min}$. Any product whose marginal benefit exceeds the threshold is fixed at the current rank and we move on to the next rank. Unlike the Simple Learning Algorithm, the Threshold Acceptance Algorithm allows us to fix products at consecutive ranks in a single pass through the products: this is the primary

[5] Technically, let $[\tilde{\pi}]_r$ denote the sub-ranking of $\tilde{\pi}$ with only the first r ranks filled and $[\tilde{\pi}]_{r-1} \cup i$ be the ranking that coincides with $\tilde{\pi}$ in its first $r-1$ positions and has product i in its r^{th} rank. For any r let U_r be the set of products that are not ranked in the first $r-1$ positions in $\tilde{\pi}$. We define $\Delta_* = \min_{r=1}^{n} \min_{i,i' \in U_r} (H_t([\tilde{\pi}]_{r-1} \cup i) - H_t([\tilde{\pi}]_{r-1} \cup j))$.

Algorithm 3: Threshold Acceptance Algorithm

Input: Parameters $\epsilon, \delta, \alpha > 0$, Initial and Final Thresholds: $\tau^{\max} \geqslant \tau^{\min} > 0$;

Initialize null ranking $\tilde{\pi}(i) = \emptyset$ for all $1 \leqslant i \leqslant n$;

Initialize unranked products $U = [n]$, $\mathtt{CurrentRank} = 1$;

Initialize $\tau = \tau^{\max}$;

while $\tau \geqslant \tau^{\min}$ *and* $\mathtt{CurrentRank} \leqslant n$ **do**

  for $i \in U$ **do**

    Define $\tilde{\pi} \cup i$ to be a ranking identical to $\tilde{\pi}$ in positions $1, ..., r-1$ and with product i in r, where $r = \mathtt{CurrentRank}$;

    Display ranking $\tilde{\pi} \cup i$ to $L = \frac{n^2}{\epsilon^2} \log(\frac{n}{\sqrt{\delta}})$ customers;

    Let Δ_{ir} be the fraction of the above customers who only click on i;

    if $\Delta_{ir} \geqslant \tau$ **then**

      Set $\tilde{\pi}(i) \leftarrow \mathtt{CurrentRank}$;

      $U \leftarrow U \setminus i$, $\mathtt{CurrentRank} \leftarrow \mathtt{CurrentRank} + 1$;

    end

  end

  $\tau \leftarrow \frac{\tau}{1+\alpha}$;

end

cause of the speed-up. Once the algorithm takes a full pass through the available products, the threshold is then lowered geometrically (i.e., from τ to $\frac{\tau}{1+\alpha}$) and thus the algorithm terminates after a logarithmic number of rounds. The choice of α allows the algorithm designer to trade off between performance and speed: small (large) α leads to more (less) conservative thresholds which result in a longer (shorter) learning phase yet better (worse) performance in the earning phase. Finally, in some cases, the ranking $\tilde{\pi}$ output by the Threshold Acceptance Algorithm may contain fewer than n products assigned to ranks. In this case, one can simply append the remaining products to $\tilde{\pi}$ (e.g., according to estimated click probabilities) without affecting the theoretical bounds.

The following result shows the algorithm's approximation factor with respect to the optimal ranking for the earning phase, as well as the required length of the learning phase.

THEOREM 4. *The Threshold Acceptance Algorithm with* $\tau^{\max} = 1, \tau^{\min} = \frac{\epsilon}{n}$ *returns a ranking* $\tilde{\pi}$ *such that with probability* $1 - \delta$

$$\mathbb{E}[H_t(\tilde{\pi})] \geqslant \frac{1}{2+\alpha} \mathbb{E}[H_t(\pi^*)] - \epsilon,$$

where π^* *is the optimal solution to OffAR. The length of the learning phase is* $O\left(\frac{n^3}{\epsilon^2} \log_{1+\alpha}(\frac{n}{\epsilon}) \log(\frac{n}{\sqrt{\delta}})\right)$.

The key idea behind the proof is two-fold: first, the product fixed by our algorithm at position r in the ranking $\tilde{\pi}$ has an empirical marginal benefit of at least τ. Due to our choice of L, its actual expected marginal benefit can be lower bounded by $\tau - \frac{\epsilon}{n}$. Second, we bound the loss due to only selecting a product that is 'good enough' by recognizing that no product can provide more marginal benefit than $(1+\alpha)\tau$, due to how the threshold is decreased at each iteration of the algorithm. The rest of the proof involves summing up the above ideas over all ranks and deriving a relationship between the marginal benefit of adding a product at rank r in $\tilde{\pi}$ vs. π^*.

Note that the length of the learning phase ($\Theta(n^4)$) for the Simple Learning Algorithm with Exhaustive Sampling is a strict and instance-agnostic limit since the algorithm makes precisely n calls to the black-box where the r^{th} such invocation contains $n - r + 1$ arms as input. For the Threshold Acceptance Algorithm, the learning phase length of $O(n^3 \log(n))$ is only a worst-case bound: in practice, the actual click probabilities tend to be closely clustered in which case the learning phase is significantly shorter. Furthermore, we note that in both algorithms, *the performance improves substantially throughout the learning phase*, since products are fixed in ranks iteratively and the initial ranks are responsible for hooking the most customers due to both (i) expiring attention windows, and (ii) customers' increased likelihood of being hooked by an earlier-ranked product.

4.3. Additional Considerations

There are a few practical considerations worth noting regarding both our algorithms and model. First, although our algorithms specify which products to rank only in positions $[1, r]$ for iteration r, in practice the retailer can simply append the remaining products (e.g., according to estimated click probabilities) without affecting the theoretical bounds. Second, the ranking returned by our algorithms would not be affected even if the retailer only observes the first product that a customer clicks on. This has key implications: for one, it implies that companies only need to track a customer's first click, rather than the entire set of products the customer clicks on. In addition, it highlights that our algorithms are robust to situations where customers change their search paths and click behavior after clicking on the first product.

Although our goal of maximizing the number of hooked customers is a very common objective in practice for companies who want to grow market share, there are other important metrics that retailers may consider, e.g., maximizing profit or total clicks. Unfortunately, our algorithms for OffAR and OnAR do not extend to these objectives because they only consider each customer's first click and do not make any assumptions on subsequent browsing or purchase behavior after clicking. We believe developing algorithms for such alternative objectives is a promising area of future work.

Finally, the performance of both our learning algorithms was stated in terms of two key metrics, namely the approximation to the optimal ranking and the speed of convergence. Presenting the performance guarantees using these two metrics is fairly typical in the online learning literature pertaining to learning-then-earning algorithms, e.g., see (Radlinski et al. 2008, Jamieson and Nowak 2014). Although notions such as *regret* are also popular, breaking down the performance in such a manner allows for practitioners to quickly gauge the tradeoffs inherently involved, e.g., the Threshold Acceptance Algorithm requires much fewer customers than the Simple Learning Algorithm to converge, but its approximation factor is slightly worse. That being said, all of our results in this section could be stated in terms of the underlying regret bound by assuming that the platform incurs unit regret during the learning phase and fixed per-round regret in the earning phase. Formally, the regret incurred by the Simple Learning Algorithm, R^{SIMPLE}, over a finite horizon of length $T > \frac{n^4}{\epsilon^2}\log(\frac{n^2}{\delta})$ with exhaustive sampling (Even-Dar et al. 2006) as its black-box could be bounded as follows:

$$E[R^{SIMPLE}] \leqslant OPT\left(\frac{n^4}{2\epsilon^2}\log(\frac{n^2}{\delta}) + \frac{1}{2}T\right) + \epsilon T,$$

with probability $(1-\delta)$. In the above bound, $OPT = \mathbb{E}[H_t(\pi^*)]$ is the value of the optimal solution although one could remove this dependence by simply using the fact that $OPT \leqslant 1$. Similarly, the regret incurred by the Threshold Acceptance Algorithm, $R^{THRESHOLD}$, over a finite horizon of length T satisfies the following bound with probability $1-\delta$:

$$E\left[R^{THRESHOLD}\right] \leqslant OPT\left((\frac{1+\alpha}{2+\alpha})\frac{n^3}{\epsilon^2}\left(1+\log_{1+\alpha}(\frac{n}{\epsilon})\right)\log(\frac{n}{\sqrt{\delta}}) + \frac{1}{2+\alpha}T\right) + \epsilon T.$$

We conclude this section by noting that it is not possible to obtain a regret bound for OnAR that is sublinear in T since OffAR is NP-Hard.

5. Simulations

In this section, we describe our empirical setting and estimate the impact of our Threshold Acceptance Algorithm as a function of a variety of model parameters.

5.1. Company and Data Description

We partnered with Wayfair - a multi-billion dollar home goods online retailer - to develop the algorithm and estimate its impact using actual clickstream data. Although a majority of Wayfair's business is driven via intentional customer search through product category pages (e.g. a customer searching for a new bed), a significant amount of sales and website traffic is through Wayfair's "events". These events are often stylistically themed and offer products from multiple product categories; typical events last about 2 weeks and offer approximately 100 products. Figure 1 shows

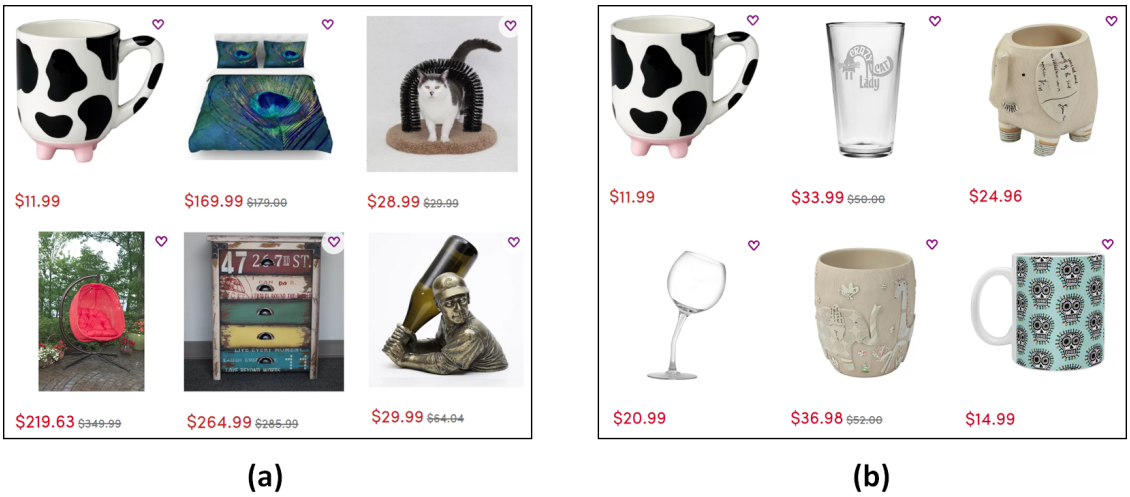

Figure 1 **Chosen from a single Wayfair event, (a) shows six diverse products with respect to style, price point, product category, color, etc. and (b) shows six more similar products across these dimensions.**

products chosen from a single Wayfair event; panel (a) highlights the diversity of products offered with respect to style, price point, product category, color, etc., whereas panel (b) shows six products that are more similar. A customer with an attention window of six products would likely have a very different perception of the full assortment of products offered in the event if she were to be shown the products in (a) vs. (b) in the first six ranks.

Wayfair provides three options for customers to sort items in events: recommended (default), customer rating, and price. For some events (particularly those that do not span multiple categories), Wayfair may offer a refinement bar to allow customers to filter the products to only those they may be interested in. Less than 5% of customers use either of these features - changing the default ranking or using filters - which supports our consumer behavior model that most customers are likely engaging in hedonic browsing as opposed to intentional search. We chose to not include customers who change the ranking or use filtering in our simulations since our algorithm is intended to replace Wayfair's default ranking and therefore would only be applied to browsing customers. Wayfair's default ranking is a static ranking (does not change over the course of the event) which ranks products in decreasing order of their expected popularity based on pre-event data.

Wayfair selected six historical, stylistically-themed events for which to evaluate our algorithm; for example, one event featured wall art and associated paraphernalia, and another featured Halloween-themed decorations. The total number of customers ($|\mathcal{T}|$) who shopped at each event ranged between 100,000 and 325,000; due to confidentiality, we disguise events as A, B, C, D, E, and F in increasing order of $|\mathcal{T}|$. For each of these events, we were provided clickstream data which includes every product each customer clicked on in the event. We represent this data as $c_{it} = 1$ if customer t

clicked on product i and 0 otherwise. We focus on only the first $n = 48$ products in the event since this is the number of products shown per page before having to click to view the second page of products. As per the data, very few customers advance to the second page; thus it is likely that the clickstream data for products not listed on the first page would be too noisy for use.

5.2. Implementation

There are two key components in implementing the simulations. First, we describe how we generate customers endowed with $(\mathbf{p}_t, k_t)$ using the clickstream data provided. Then, we describe the implementation details of the algorithm itself.

5.2.1. Customer Generation

Since $(\mathbf{p}_t, k_t)$ is unobserved, we must infer it from the censored clickstream data. To do so, we make three simplifying assumptions. First, we assume interest probabilities are independent of attention windows (OnAR with Independence). Second, we assume that if a customer is hooked, she views the remaining products on the page (i.e., she views the first 48 products in the event). Finally, we assume that $p_{it} \in \{0, 1\}\ \forall i, t$ where $p_{it} = 1$ implies that customer t will click on product i if she views it. The second assumption is motivated by the fact that customers who are interested in purchasing a product tend to carefully examine the whole assortment before making a final choice. Moreover, this is actually a very conservative assumption that handicaps our algorithm's performance. For example, the assumption would entail assigning $p_{it} = 0$ for a hooked customer t and all products i that they did not click on while in reality, the customer may not have viewed certain products. Consequently, the performance of our algorithm on the click data is only a lower bound on its potential real-world performance. In Appendix C, we present additional simulations that were conducted on a different, less-conservative set of customer behavior assumptions. In reality, customer behavior and the performance of our algorithm may fall somewhere in-between.

Next we describe our inference methodology. First we fix the distribution of attention windows $\mathcal{D}_k$ and the total number of customers who have non-zero preferences, $|\tilde{\mathcal{T}}|$ (i.e., $\sum_i p_{it} \geqslant 1$ for $t \in \tilde{\mathcal{T}}$). Given $\mathcal{D}_k$ and $|\tilde{\mathcal{T}}|$, we will infer the distribution of interest probabilities, $\mathcal{D}_p$. Subsequently we will discuss our choices of $\mathcal{D}_k$ and $|\tilde{\mathcal{T}}|$. Let $\tilde{\mathcal{T}}_c$ be the subset of customers from $\tilde{\mathcal{T}}$ who clicked on a product, i.e. $\tilde{\mathcal{T}}_c \triangleq \{t \in \tilde{\mathcal{T}} : \sum_i c_{it} \geqslant 1\}$. Note that $\tilde{\mathcal{T}}_c \subseteq \tilde{\mathcal{T}} \subseteq \mathcal{T}$ and $p_{it} = c_{it}\ \forall t \in \tilde{\mathcal{T}}_c$ and $i \in [n]$; the challenge is that we do not observe $\mathbf{p}_t$ for $t \in \tilde{\mathcal{T}} \setminus \tilde{\mathcal{T}}_c$, when $\mathbf{c}_t = 0$.

1. *Derive distribution of attention windows for customers* $t \in \tilde{\mathcal{T}}_c$*:* We define r_t^1 to be the index of the first product customer t clicks on and let i be the product located at rank r_t^1. Then, it must be true that $c_{it} = 1$ and $c_{i't} = 0$ for any product i' placed at a rank smaller than r_t^1. Since the customer clicked on the r_t^1–th ranked product, her attention window k_t must be greater than or equal to r_t^1. That is, for $r < r_t^1$, $\Pr(k_t = r | t \in \tilde{\mathcal{T}}_c, r_t^1) = 0$. Therefore, for $r \geqslant r_t^1$, $\Pr(k_t = r | t \in \tilde{\mathcal{T}}_c, r_t^1) = \frac{\Pr(k_t = r)}{\sum_{l=r_t^1}^{n} \Pr(k_t = l)}$, where $\Pr(k_t = l)$ comes from the distribution of attention windows, $\mathcal{D}_k$.

2. *Weight each observed preference vector:* For each customer $t \in \tilde{\mathcal{T}}_c$, assign weight $w_t = \frac{1}{\sum_{l=r_t^1}^{n} \Pr(k_t = l | t \in \tilde{\mathcal{T}}_c)}$ to preference vector $\mathbf{p}_t$. Intuitively, this assigns larger weights to preference vectors whose first clicks are in greater ranks, which accounts for the fact that those preference vectors are less likely to be observed in the click data due to expiring attention windows. This gives rise to a distribution $\mathcal{D}_p$ with a set of preference vectors and a weight on each vector that is proportional to the probability of a customer having the corresponding preferences.

3. *Sample preference vectors for* $|\tilde{\mathcal{T}}|$ *customers:* Sample $|\tilde{\mathcal{T}}|$ preference vectors from the empirical distribution of $\mathcal{D}_p$.

Now that we have outlined our approach of estimating $\mathcal{D}_p$ given $\mathcal{D}_k$ and $|\tilde{\mathcal{T}}|$, we discuss our choices of $\mathcal{D}_k$ and $|\tilde{\mathcal{T}}|$. For $\mathcal{D}_k$, we assumed 5% of customers viewed all n products on the page ($k_t = n$) and the remaining customers' attention windows followed a power law distribution where $\Pr(k = r) = \frac{a r^{-b}}{\sum_{l=1}^{n-1} a l^{-b}}$ $\forall r = 1, ..., n-1$. Power law distributions are well-motivated in the context of consumer behavior and product popularity on the Internet (Clauset et al. 2009). We set $|\tilde{\mathcal{T}}| = 0.8|\mathcal{T}|$ based on the parameter choices given to us by our partners at Wayfair. We conducted sensitivity analysis on these parameter choices to better understand their impact on the algorithm's performance; our analysis gleans similar results so we relegate it to Appendix B.

We tuned parameters a and b of the power law distribution such that the resultant $\mathcal{D}_k$ and $\mathcal{D}_p$ most closely matched the observed data. In particular, after generating $|\tilde{\mathcal{T}}|$ customers per the procedure outlined above, we randomly assigned each of these customers an attention window according to $\mathcal{D}_k$. We then identified which customers would have been hooked had they been presented with Wayfair's actual ranking in order to generate (hypothetical) click vectors, and we compared the total number of actual clicks vs. the total number of hypothetical clicks for each product. Figure 2 presents the actual vs. hypothetical number of clicks for each product in each of the six events, where the hypothetical clicks are averages over 100 samples in step 3. The close fit illustrates that our customer generation procedure is congruent with the observed data.

5.2.2. Algorithm Details We implemented the Threshold Acceptance Algorithm (Algorithm 3) described in Section 4.2. We chose the parameter α to be small and constant across events so that the threshold is lowered conservatively in each iteration, due to click probabilities generally being in a fairly narrow range. We varied the number of customers for which to offer each rank (L) such that $L \in [100, 1000]$, allowing us to evaluate the tradeoff between faster vs. more accurate learning. Finally, the implementation of our algorithm includes two features - described below - that were not required to develop the theory yet can make a considerable impact on performance in practice. The purpose of the first feature is to speed up the learning phase of the algorithm, and the purpose of the second feature is to hook more customers during the learning phase.

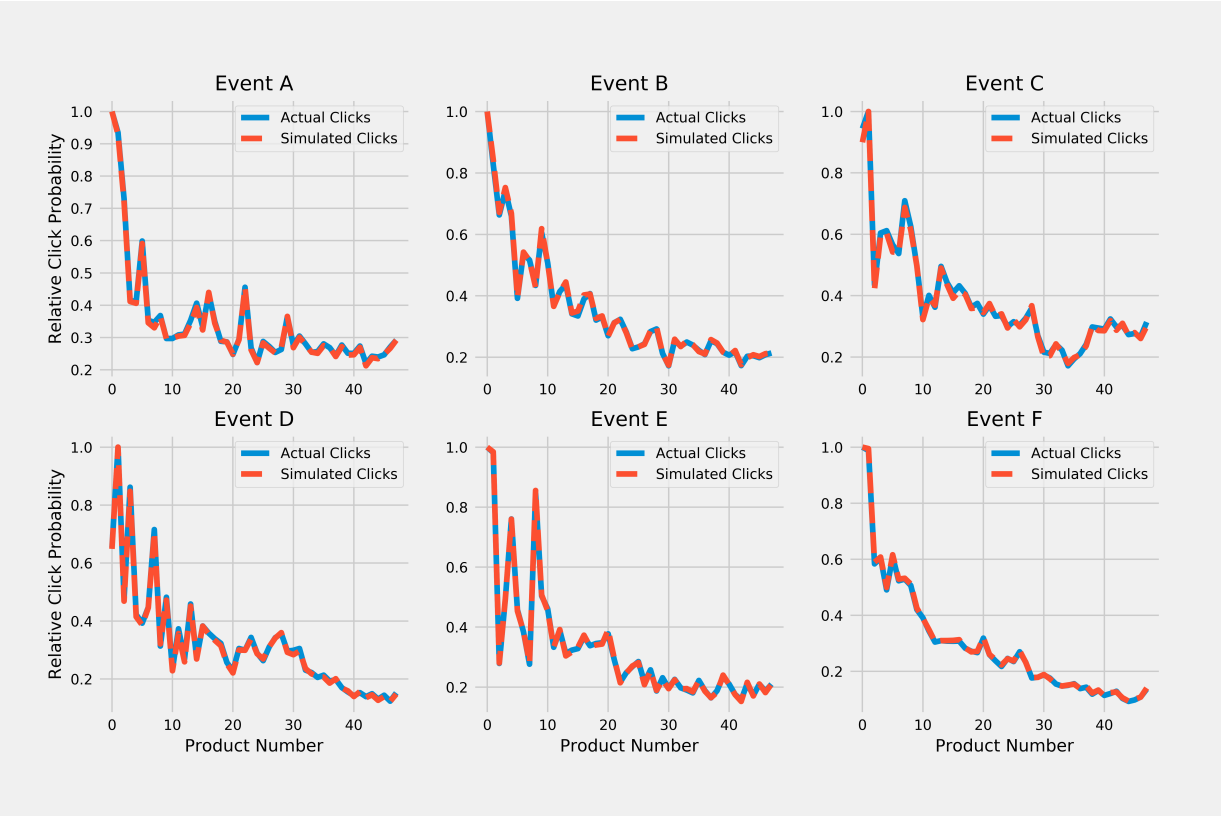

Figure 2 **Actual and simulated relative click probabilities for all six events provided by Wayfair. The click probabilities are normalized for confidentiality so that the maximum click probability of any product is one.**

First, we recognize that the expected number of new customers that can be hooked by product $\mathtt{i}$ diminishes with rank due to both expiring attention windows and because as the rank increases, the customer has a higher probability of getting hooked prior to seeing product $\mathtt{i}$. Using the notation in Algorithm 3, we can therefore use $\Delta_{\mathtt{ir}}$ calculated at rank $\mathtt{r}$ as an upper bound for $\Delta_{\mathtt{ir}'}$ at rank $\mathtt{r}'$ for any $\mathtt{r}' > \mathtt{r}$. When selecting products to test for rank $\mathtt{r}$, we choose products in descending order of their tightest upper bound which allows us to more quickly find a product that passes the threshold requirement for fixing in rank $\mathtt{r}$ and eliminates having to test clearly suboptimal products for each rank. The notion of leveraging upper bounds to speed up calculations has its origins in submodular optimization, where it is referred to as *lazy evaluations* (Leskovec et al. 2007).

Second, during the course of the algorithm, as we evaluate the marginal benefit of a product at a given rank, the remaining unranked products are placed in descending order of the upper bound described above. For products not yet tested in any rank, we include them at the end of the ranking in increasing order of their Wayfair rank. Note that this implies we use Wayfair's ranking as an input into our algorithm, but *not* their estimated popularity metrics; this is reasonable as it is the current methodology used for ranking and is how we would implement the algorithm in practice.

Furthermore, note that using Wayfair's ranking as an input does not necessarily imply that our algorithm would perform better in simulations, as our ranking must spend a considerable amount of time learning; put differently, if Wayfair's current ranking was indeed optimal, our algorithm would produce worse performance metrics as it would have to explore suboptimal rankings.

5.3. Results

For each event and each set of model parameters, we conducted the following simulation 100 times. First, we generated $|\mathcal{T}|$ customers such that each customer's attention window was drawn from $\mathcal{D}_k$. With probability $q = \frac{|\mathcal{T}_c|}{|\mathcal{T}|}$, $\mathbf{p}_t$ is drawn from $\mathcal{D}_p$; otherwise, $\mathbf{p}_t = \mathbf{0}$. We applied our algorithm sequentially to customers $t = 1, ..., |\mathcal{T}|$ and calculated the percent of customers that our algorithm hooked. Unless otherwise specified, we used a sample size of $L = 500$ customers for each ranking offered during the learning phase; we present sensitivity analysis on this modeling choice at the end of this section. We also identified the percent of customers that Wayfair's static ranking hooked, which serves as a practical, popularity-based benchmark for which to compare our algorithm. By our choice of $\mathcal{D}_p$ and $\mathcal{D}_k$, this quantity is consistently very close to the actual percent of customers Wayfair hooked in the event: In 99.5% of our simulations, the percent of customers hooked by Wayfair's ranking was within [0.98, 1.02] times the actual percent of customers hooked.

Figure 3 shows the simulation results for the performance of our algorithm compared to Wayfair's static ranking (π^{WF}). In particular, it displays the percent increase in the number of customers hooked by our algorithm vs. Wayfair's ranking[6]: (total hooked customers by our algorithm - total hooked customers by π^{WF}) / (total hooked customers by π^{WF}). Across all six events, our algorithm hooks an average of 5-30% more customers than Wayfair's ranking. There is variability both across simulations and across events. Variability across simulations stems from a relatively low number of customers L who are offered each ranking compared to the number of possible preference vectors in $\mathcal{D}_p$; thus, our algorithm may converge to a (slightly) different ranking in each simulation. Interestingly, even with this noise in the learning process, the worst-case improvement across all simulations was still considerable for five of the six events (9-22%). In addition, our algorithm led to a positive increase in the percent of hooked customers for every simulation; notably, this is not trivial - if Wayfair's ranking was optimal, our algorithm would perform worse due to exploring suboptimal rankings before converging.

Variability across events stems from the degree of optimality of Wayfair's static ranking. Intuitively, when π^{WF} is near optimal, there is not much opportunity for our algorithm to improve upon the ranking and vice versa. To explore this further, we implemented the Greedy Algorithm for OffAR for each of the simulations and calculated the percent of customers it would have hooked

[6] Due to confidentiality, we report only relative metrics rather than disclosing the actual percent of hooked customers.

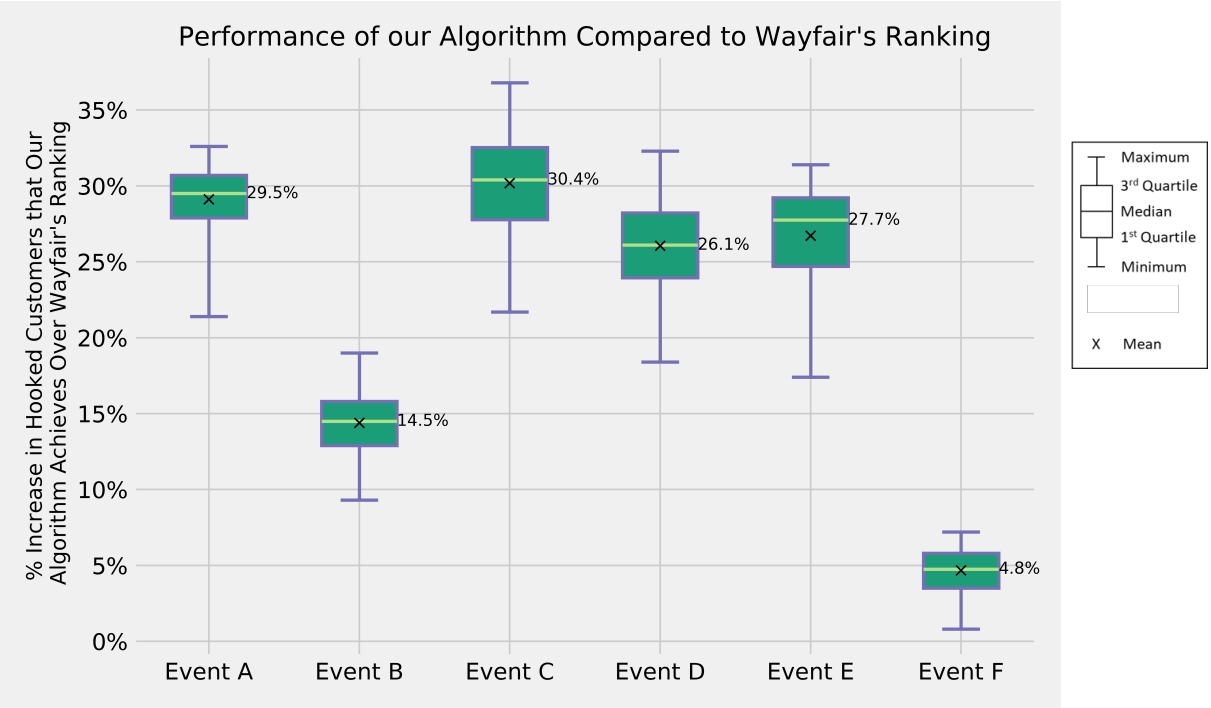

Figure 3 **Box & whisker plots for the percent increase in hooked customers our algorithm achieves over π^{WF}**

had it been offered (as a static ranking) to each of the customers; we use this as an upper bound[7] for the number of customers hooked by our algorithm and π^{WF}. Figure 4 compares the performance of our algorithm and π^{WF} to the static ranking output by the Greedy Algorithm for OffAR (π^g): specifically, (total number of hooked customers by our algorithm (or π^{WF})) / (total number of hooked customers by π^g).

Across all six events, our algorithm has consistently strong performance, hooking an average of 89-95% of the total number of customers that π^g hooks. Although there is variability across simulations as before, the average is fairly consistent and highlights our algorithm's ability to (approximately) converge to and achieve the results in the offline setting. In contrast, there is considerable variability in the performance of Wayfair's ranking: across all six events, π^{WF} hooks an average of 68-90% of the total number of customers that π^g hooks. For events where Wayfair's ranking was quite strong (B and F), our algorithm improved on Wayfair's ranking by 5-15%. For events where Wayfair's ranking was not as strong (A, C, D, and E), our algorithm improved on Wayfair's ranking by 26-30%.

Finally, Figure 5 presents sensitivity analysis on our choice of the sample size L - the number of customers to offer each ranking during the learning phase. The analysis illustrates that when the

[7] Given that the Greedy Algorithm for OffAR is an approximation algorithm, it is technically possible for the performance of a ranking to be larger than this ranking; however, that does not happen in any of our simulations.

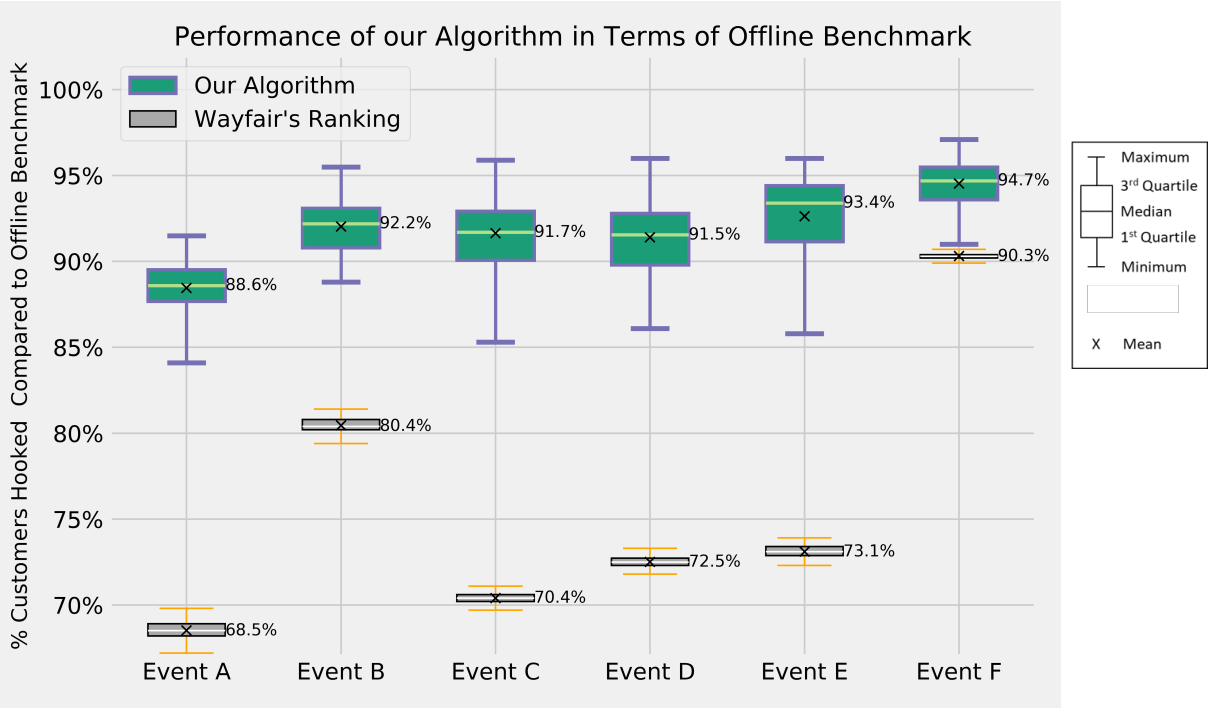

Figure 4 **Box & whisker plots presenting the percent of total customers hooked by π^g (the offline benchmark) that are hooked by our algorithm and π^{WF}**

sample size is small, there is insufficient learning and the algorithm's performance suffers. As the sample size increases up to approximately 400 or 500 customers, the algorithm is able to sufficiently learn the (approximate) best ranking. For the smallest event (A), we see evidence that as the sample size grows too large, the algorithm is unable to learn fast enough to reap the benefits of more precise learning. Interestingly, we see that for all but the smallest event, the performance of our algorithm is remarkably constant for a wide range of sample sizes. This is a desirable artifact since Wayfair can simply choose a single sample size for the algorithm's implementation across all events - likely $\mathsf{L} = 400$ or 500 - without having to estimate the event size T *a priori*.

6. Conclusion

We study a crucial problem faced by many online retailers - that of developing *fast online learning approaches* for computing near-optimal product rankings. We propose a novel model for online learning-to-rank problems that captures a number of features unique to online retailers operating in dynamic environments and which have yet to receive attention in the academic literature. Specifically, we study scenarios where the products have a subjective style and cannot be fully characterized by their attributes, and where customers can be described as "window shoppers". By modeling customers with limited attention windows, we capture the position effect typically observed in online environments. Yet our framework is more challenging as these attention windows

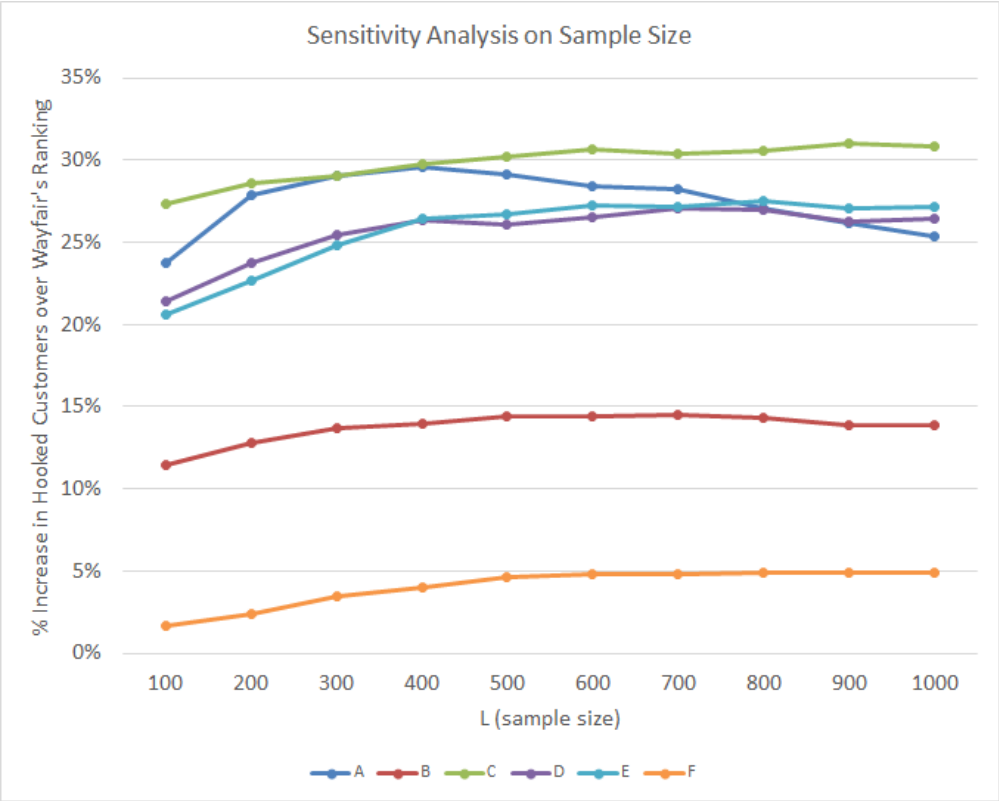

Figure 5 **Sensitivity Analysis on Sample Size,** L

are unobserved and may be correlated with customers' interest in products. As such, our model is a significant departure from existing work on ranking algorithms.

We present two learning algorithms for the online ranking problem: the Simple Learning Algorithm and the Threshold Acceptance Algorithm. The former uses bandit algorithms for best-arm identification as a black-box to achieve a (nearly) $\frac{1}{2}$-approximation guarantee but may require $O(n^4)$ customer samples in the worst case. The latter algorithm is an order of magnitude faster but its approximation guarantee becomes $\frac{1}{2+\alpha}$ for some parameter $\alpha > 0$. Both of these algorithms require no knowledge of customers' intrinsic utilities or attention windows, and circumvent having to learn this distribution by directly learning the optimal ranking instead. Another interesting feature of our algorithms is that they optimally balance popularity and diversity for each event. This is particularly useful in the case of stylistic products as product similarities are subjective.

We estimate the impact of the Threshold Acceptance Algorithm via simulations using actual clickstream data from Wayfair. Across six different product assortments, our algorithm hooks an average of 5-30% more customers than Wayfair's static, popularity-based ranking. Furthermore, our algorithm hooks 89-95% of the total number of customers hooked by an offline benchmark. These simulations using real clickstream data illustrate that our algorithmic contributions can make a significant impact in practice.

Although our model and results are motivated from the perspective of product ranking, we argue that they may be relevant in any setting with dynamic assortments, subjective products, and customers with limited attention windows. As a concrete example, a digital media outlet could employ our algorithm to highlight 'top articles' - by balancing popularity and diversity, the website could hook a larger fraction of its incoming users.

Finally, we share opportunities for future work. First, retailers could benefit from understanding the long-term value of a hooked customer. Although we make the very weak assumption that a hooked customer browses for longer than one who is not hooked, it would be pertinent to understand more details about a hooked customer's browsing, purchasing, and return shopping behavior. Second, although our objective of maximizing the number of hooked customers is a very common objective in practice for companies who want to grow market share, we believe that there would be value in developing alternative algorithms for retailers looking to maximize other objectives, such as maximizing profit. Third, "hedonic browsing" is a common customer behavior that many retailers face, yet much of the academic work focuses primarily on customers who engage in "directed buying", i.e. who shop with the intent to purchase a product and have knowledge of what they are looking for (Moe (2003)). We hope that our work inspires others to consider studying the impact of hedonic browsing behavior on a firm's operational and marketing decisions.

References

Agrawal S, Avadhanula V, Goyal V, Zeevi A (2016) A near-optimal exploration-exploitation approach for assortment selection. *Proceedings of the 2016 ACM Conference on Economics and Computation*, 599–600 (ACM).

Aouad A, Segev D (2015) Display optimization for vertically differentiated locations under multinomial logit choice preferences, working paper.

Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*, 1027–1035.

Asadpour A (2019) Personal Communication.

Asadpour A, Nazerzadeh H (2015) Maximizing stochastic monotone submodular functions. *Management Science* 62(8):2374–2391.

Badanidiyuru A, Vondrák J (2014) Fast algorithms for maximizing submodular functions. *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, 1497–1514.

Bernstein F, Modaresi S, Saure D (2017) A dynamic clustering approach to data-driven assortment personalization, working paper.

Bloch P, Ridgway N, Sherrell D (1989) Extending the concept of shopping: An investigation of browsing activity. *Journal of the Academy of Marketing Science* 17(1):13–21.

Chen W, Wang Y, Yuan Y, Wang Q (2016) Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *The Journal of Machine Learning Research* 17(1):1746–1778.

Chen Y, Yao S (2016) Sequential search with refinement: Model and application with click-stream data. *Management Science* 63(12):4345–4365.

Cho E, Youn-Kyung K (2012) The effects of website designs, self-congruity, and flow on behavioral intention. *International Journal of Design* 6(2).

Chu LY, Nazerzadeh H, Zhang H (2017) Position ranking and auctions for online marketplaces, working paper.

Clauset A, Shalizi CR, Newman MEJ (2009) Power-law distributions in empirical data. *SIAM Review* 51(4):661–703.

Dawson S, Kim M (2010) Cues on apparel web sites that trigger impulse purchases. *Journal of Fashion Marketing and Management: An International Journal* 14(2):230–246.

Dzyabura D, Hauser JR (2016) Recommending products when consumers learn their preferences, working paper.

Economist (2017) Stores are being hit by online retailing. *The Economist* URL `https://www.economist.com/special-report/2017/10/26/stores-are-being-hit-by-online-retailing`.

Even-Dar E, Mannor S, Mansour Y (2006) Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research* 7(Jun):1079–1105.

Gallego G, Li A, Truong VA, Wang X (2016) Approximation algorithms for product framing and pricing, working paper.

Ghose A, Ipeirotis PG, Li B (2012) Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content. *Marketing Science* 31(3):493–520.

Ghose A, Ipeirotis PG, Li B (2014) Examining the impact of ranking on consumer behavior and search engine revenue. *Management Science* 60(7):1632–1654.

Golrezaei N, Manshadi V, Mirrokni V (2018) Two-stage pandora's box for product ranking, working paper.

Hochbaum DS (1996) *Approximation Algorithms for NP-hard problems* (PWS Publishing Co.).

Jamieson KG, Nowak RD (2014) Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. *48th Annual Conference on Information Sciences and Systems, CISS 2014, Princeton, NJ, USA, March 19-21, 2014*, 1–6.

Kalyanakrishnan S, Tewari A, Auer P, Stone P (2012) PAC subset selection in stochastic multi-armed bandits. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*.

Kim JB, Albuquerque P, Bronnenberg BJ (2010) Online demand under limited consumer search. *Marketing Science* 29(6):1001–1023.

Kim JB, Albuquerque P, Bronnenberg BJ (2016) The probit choice model under sequential search with an application to online retailing. *Management Science* 63(11):3911–3929.

Koulayev S (2014) Search for differentiated products: identification and estimation. *The RAND Journal of Economics* 45(3):553–575.

Kveton B, Wen Z, Ashkan A, Szepesvári C (2015) Combinatorial cascading bandits. *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 1450–1458.

Lagrée P, Vernade C, Cappé O (2016) Multiple-play bandits in the position-based model. *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 1597–1605.

Lei YM, Jasin S, Uichanco J, Vakhutinsky A (2018) Randomized product display (ranking), pricing, and order fulfillment for e-commerce retailers, working paper.

Leskovec J, Krause A, Guestrin C, Faloutsos C, VanBriesen J, Glance N (2007) Cost-effective outbreak detection in networks. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 420–429 (ACM).

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

Moe WW (2003) Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of Consumer Psychology* 13(1-2):29–39.

Radlinski F, Kleinberg R, Joachims T (2008) Learning diverse rankings with multi-armed bandits. *Proceedings of the 25th international conference on Machine learning*, 784–791 (ACM).

Raman K, Shivaswamy P, Joachims T (2012) Online learning to diversify from implicit feedback. *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, 705–713.

Rowley J (2002) Window shopping and browsing opportunities in cyberspace. *Journal of Consumer Behaviour: An International Research Review* 1(4):369–378.

Ulu C, Honhon D, Alptekinoglu A (2012) Learning consumer tastes through dynamic assortments. *Operations Research* 60(4):833–849.

Ursu RM (2018) The power of rankings: Quantifying the effect of rankings on online consumer search and purchase decisions. *Marketing Science* .

Weitzman ML (1979) Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society* 641–654.

Zoghi M, Tunys T, Ghavamzadeh M, Kveton B, Szepesvári C, Wen Z (2017) Online learning to rank in stochastic click models. *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 4199–4208.

Appendix A: Proofs

Proof of Proposition 1: Our proof involves showing a reduction from the deterministic maximum coverage problem to a special case of OffAR. Since the deterministic maximum coverage problem is known to be NP-Hard, the same claim follows for the special case of OffAR and thus the more general OffAR problem is also NP-Hard. The deterministic maximum coverage problem is defined by a universe $\mathcal{U}$ of elements $\{u_1, u_2, \ldots, u_m\}$, a collection of n sets $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where $S_i \subseteq \mathcal{U}$ for all i, and finally a single integer k such that $1 \leqslant k \leqslant n$. The objective is to select at most k sets from $\mathcal{S}$ in order to maximize the number of elements in $\mathcal{U}$ that are covered by (present in) at least one of the sets, i.e.

$$\max_{\mathcal{S}' \subseteq \mathcal{S}; |\mathcal{S}'| \leqslant k} \left| \cup_{S_i \in \mathcal{S}'} S_i \right|. \tag{2}$$

The reduction is straightforward. Given an instance of the deterministic maximum coverage problem, we construct an instance of OffAR with: (i) n products labeled by $[n]$, such that product i corresponds to set $S_i \in \mathcal{S}$; (ii) distribution $\mathcal{D}$ comprising of m customer types such that each type occurs with equal probability $\frac{1}{m}$, and for a customer t of the j^{th} type, $p_{it} = 1$ if and only if $u_j \in S_i$ in the original problem. In other words, we transform the problem so that for every set S_i, a new product is created and for every element of the universe $\mathcal{U}$, there is a customer type who prefers all of the sets (products) which contain that element of the universe (are liked by that customer type). Finally, all customers have an attention window equal to k in this reduced instance.

Consider any ranking π and let $S_\pi(k) \subseteq \mathcal{S}$ correspond to the top-k ranked products in π. Recall that $\pi^{-1}(j)$ denotes the j^{th} ranked product in π. Then, formally,

$$S_\pi(k) = \{S_{\pi^{-1}(1)}, \ldots, S_{\pi^{-1}(k)}\}.$$

Let $\mathbb{E}[H_t(\pi)]$ be the probability that a single incoming customer drawn from $\mathcal{D}$ is hooked when she is exposed to the ranking π. We then claim that for the deterministic maximum coverage problem with sets $S_\pi(k)$ chosen, the number of elements in $\mathcal{U}$ that are covered is precisely $m\mathbb{E}[H_t(\pi)]$. Indeed, consider any customer type j belonging to the distribution $\mathcal{D}$ that is hooked by the ranking π: this implies that there exists some i having $\pi(i) \leqslant k$ such that $u_j \in S_i$. However, this in turn implies that the collection $S_\pi(k)$ must cover the element u_j in the universe. We further observe that each customer type in $\mathcal{D}$ (corresponding to an element in the universe) occurs with equal probability $\frac{1}{m}$. Therefore, there is a one-to-one correspondence between the optimal solutions to the original, deterministic maximum coverage problem and the constructed (stochastic) offline assortment ranking problem, which in turn implies the NP-Hardness of the latter. □

Notation for Subsequent Proofs Before proving Theorem 1, we introduce notation that will help streamline our presentation. Specifically, we use $\overline{C}_t(r, \pi)$ to denote the indicator random variable for the event that a customer t does not click on any of the top-r ranked products in π, i.e., for $1 \leqslant r \leqslant n$

$$\overline{C}_t(r, \pi) = \prod_{j \in [r]} \overline{C}_{\pi^{-1}(j)t}(\pi).$$

We define $\overline{C}_t(0,\pi)=1$ for notational convenience. As before, by indexing customer t, we implicitly assume that this random variable is conditional on customer t's type, $(\mathbf{p}_t, k_t)$.

A subtle point bears mentioning here: $\overline{C}_{\pi^{-1}(j)t}(\pi)=1$ implies that either $j>k_t$ and/or the customer did not click on product $\pi^{-1}(j)$. However, if the customer did not click on any of the top k_t ranked products, she is *not hooked* and therefore, cannot have clicked on any of the products with rank larger than k_t. Therefore, for any $r>k_t$, $\overline{C}_t(r,\pi)=\overline{C}_t(k_t,\pi)$ and it is accurate to interpret $\overline{C}_t(r,\pi)=1$ as the event that the customer did not click on any of the top-r ranked products under π.

Finally, we note the following equality that will be used in several proofs.

$$\begin{aligned}\sum_{r=1}^{n}\mathbb{E}\big[\overline{C}_t(r-1,\pi)C_{\pi^{-1}(r)t}(\pi)\big] &= \sum_{r=1}^{n}\mathbb{E}\big[\big(1-\overline{C}_t(r,\pi)\big)-\big(1-\overline{C}_t(r-1,\pi)\big)\big]\\ &= \mathbb{E}\big[\big(1-\overline{C}_t(n,\pi)\big)-\big(1-\overline{C}_t(0,\pi)\big)\big]\\ &= \mathbb{E}[H_t(\pi)]. \end{aligned} \tag{3}$$

Proof of Theorem 1 Our objective is to prove that for any incoming customer $t\sim\mathcal{D}$, the probability that this customer is hooked under the optimal ranking π^* is at most twice the probability that she is hooked under the greedy ranking π^g. Mathematically, we need to prove that

$$\mathbb{E}[H_t(\pi^*)]=\mathbb{E}[1-\overline{C}_t(k_t,\pi^*)]\leqslant 2\,\mathbb{E}[1-\overline{C}_t(k_t,\pi^g)]=2\mathbb{E}[H_t(\pi^g)],$$

where $\overline{C}_t(r,\pi)$ is defined immediately prior to the proof of this theorem.

We remark that the expectation involves two sources of uncertainty: (i) the fact that the arriving customer t's preferences $(\mathbf{p}_t,k_t)$ are drawn from distribution $\mathcal{D}$, (ii) given customer t with preferences $(\mathbf{p}_t,k_t)$, the event that the customer clicks on product i within her attention window is a random variable. Since $H_t(\pi^*)$ is conditional on $(\mathbf{p}_t,k_t)$, one could rewrite the above expectation as $\mathbb{E}_{(\mathbf{p}_t,k_t)}[\mathbb{E}[H_t(\pi^*)]]$, where the inner expectation is the probability that a customer t is hooked given $\mathbf{p}_t$ and k_t. For brevity, we capture both sources of uncertainty under a single expectation in the rest of the proofs. Suppose that π denotes some arbitrary ranking. We begin by decomposing π^* in terms of π and eventually substitute $\pi=\pi^g$,

$$\begin{aligned}\mathbb{E}[H_t(\pi^*)] &= \mathbb{E}\big[1-\overline{C}_t(k_t,\pi^*)\big]\\ &= \mathbb{E}\big[1-\overline{C}_t(n,\pi^*)\big] &(4)\\ &\leqslant \mathbb{E}\big[1-\overline{C}_t(n,\pi^*)\overline{C}_t(n,\pi)\big] &(5)\\ &= \mathbb{E}[1-\overline{C}_t(n,\pi)]+\sum_{r=1}^{n}\mathbb{E}\big[\big(1-\overline{C}_t(n,\pi)\overline{C}_t(r,\pi^*)\big)-\big(1-\overline{C}_t(n,\pi)\overline{C}_t(r-1,\pi^*)\big)\big] &(6)\\ &= \mathbb{E}[H_t(\pi)]+\sum_{r=1}^{n}\mathbb{E}\big[\big(1-\overline{C}_t(n,\pi)\overline{C}_t(r,\pi^*)\big)-\big(1-\overline{C}_t(n,\pi)\overline{C}_t(r-1,\pi^*)\big)\big]\\ &= \mathbb{E}[H_t(\pi)]+\sum_{r=1}^{n}\mathbb{E}\big[\overline{C}_t(n,\pi)\big(\overline{C}_t(r-1,\pi^*)-\overline{C}_t(r,\pi^*)\big)\big]\\ &= \mathbb{E}[H_t(\pi)]+\sum_{r=1}^{n}\mathbb{E}\big[\overline{C}_t(n,\pi)\overline{C}_t(r-1,\pi^*)(1-\overline{C}_{\pi^{*-1}(r)t}(\pi^*))\big]\end{aligned}$$

$$= \mathbb{E}[H_t(\pi)] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(n,\pi)\overline{C}_t(r-1,\pi^*)C_{\pi^{*-1}(r)t}(\pi^*)\big]$$
$$\leqslant \mathbb{E}[H_t(\pi)] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\pi)C_{\pi^{*-1}(r)t}(\pi^*)\big] \tag{7}$$

Equality (4) follows from the definition that $C_{it}(\pi) = 0$ when $\pi(i) > k_t$. Inequality (5) simply comes from the fact that $\overline{C}_t(n,\pi) \leqslant 1$. The equality in (6) is a rearrangement of $\mathbb{E}\big[1-\overline{C}_t(n,\pi^*)\overline{C}_t(n,\pi)\big]$ in the form of a telescoping summation, i.e., $\sum_{r=1}^{n}\mathbb{E}\big[\big(1-\overline{C}_t(n,\pi)\overline{C}_t(r,\pi^*)\big)-\big(1-\overline{C}_t(n,\pi)\overline{C}_t(r-1,\pi^*)\big)\big] = \mathbb{E}\big[1-\overline{C}_t(n,\pi^*)\overline{C}_t(n,\pi)\big]-\mathbb{E}\big[1-\overline{C}_t(n,\pi)\big]$. The intermediate steps between (6) and (7) follow from simple algebraic manipulations and the definition of $C_{\pi^{*-1}(r)t}(\pi^*)$. Inequality (7) follows because $\overline{C}_t(r-1,\pi^*) \leqslant 1$ and

$$\overline{C}_t(n,\pi) = \prod_{j\in[n]} \overline{C}_{\pi^{-1}(j)t}(\pi) \leqslant \prod_{j\in[r-1]} \overline{C}_{\pi^{-1}(j)t}(\pi) = \overline{C}_t(r-1,\pi)$$

since the right-hand-side refers to only a subset of the events on the left-hand-side.

By substituting $\pi = \pi^g$ in (7), we get:

$$\mathbb{E}[H_t(\pi^*)] \leqslant \mathbb{E}[H_t(\pi^g)] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big] \tag{8}$$

To complete our proof, we next show that $\sum_{r=1}^{n}\mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big] \leqslant \mathbb{E}[H_t(\pi^g)]$. Consider $\mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big]$ for any $r \in [n]$. We claim that

$$\mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big] \leqslant \mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{g-1}(r)t}(\pi^g)\big]. \tag{9}$$

To prove (9), we define $i \triangleq \pi^{*-1}(r)$. Then, the left hand side of the above inequality is the marginal contribution of adding product i in position r while fixing the top $r-1$ positions in π^g, i.e., the probability that an incoming customer t does not click on any of the top $r-1$ products in π^g and clicks on product i when it is placed in rank r. However, we know that the Greedy Algorithm for OffAR fixes the product $\pi^{g-1}(r)$ in position r that leads to the maximum marginal increment towards getting an incoming customer hooked. Therefore, the marginal contribution of product i cannot be larger than that of the product actually selected by the greedy algorithm. Equation (9) is formally proved in Lemma 1 which immediately follows the proof of this theorem. From Equation (9) and applying Equation (3), we have

$$\sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big] \leqslant \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{g-1}(r)t}(\pi^g)\big] = \mathbb{E}[H_t(\pi^g)]. \tag{10}$$

Combining (8) and (10), we have

$$\mathbb{E}[H_t(\pi^*)] \leqslant \mathbb{E}[H_t(\pi^g)] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big] \leqslant \mathbb{E}[H_t(\pi^g)] + \mathbb{E}[H_t(\pi^g)] = 2\mathbb{E}[H_t(\pi^g)]. \quad \square$$

The following lemma is used in the proof of Theorem 1.

LEMMA 1. *Suppose that π^* denotes the optimal ranking and π^g denotes the ranking produced by the Greedy Algorithm for OffAR. Let $r \leqslant n$ denote any index. Then, we have that:*

$$\mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\big] \leqslant \mathbb{E}\big[\overline{C}_t(r-1,\pi^g)C_{\pi^{g-1}(r)t}(\pi^g)\big].$$

Proof of Lemma 1: First, we define an alternative ranking $\tilde{\pi}^g$ such that for all $j \leqslant r-1$, $\tilde{\pi}^{g-1}(j) = \pi^{g-1}(j)$ and $\tilde{\pi}^{g-1}(r) = \pi^{*-1}(r)$. That is, $\tilde{\pi}^g$ mirrors the ranking π^g in the first $r-1$ positions and shares the r^{th} rank with ranking π^*. Let $\tilde{\pi}^{g-1}(j) = \emptyset$ when $j > r$, i.e. no product is ranked in positions greater than r.

Suppose that i denotes the r^{th} ranked product in π^*, i.e. $i \triangleq \pi^{*-1}(r)$. Going back to the expression in the left hand side of the lemma statement, we can apply the rule of iterated expectations and get,

$$\mathbb{E}\left[\overline{C}_t(r-1,\pi^g)C_{\pi^{*-1}(r)t}(\pi^*)\right] = \mathbb{E}\left[\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*)\right] = \mathbb{E}_t\left[\mathbb{E}[\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*)|(\mathbf{p}_t,k_t)]\right]. \tag{11}$$

In the final term, the outer expectation is with respect to the randomness in the customer's preferences (i.e., $(\mathbf{p}_t,k_t)$) and the inner expectation is with respect to the randomness in the customer's clicks given $\mathbf{p}_t$ and k_t. Note that since $\overline{C}_t(r-1,\pi^g)$ and $C_{it}(\pi^*)$ are already defined conditional on $(\mathbf{p}_t,k_t)$, the inner expectation is somewhat redundant and just represents the probability that an incoming customer t would not click on any of the top $r-1$-ranked products when presented with π^g but would click on product i when presented with π^*. However, we choose to express the conditional statement explicitly to improve the readability of the proof.

We now consider three cases depending on the relative values of $r, \pi^g(i)$, and k_t. In each of the three cases, we show that for all t, the expression $\mathbb{E}[\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*)|(\mathbf{p}_t,k_t)]$ from (11) is less than or equal to $\mathbb{E}[\overline{C}_t(r-1,\tilde{\pi}^g)C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t,k_t)]$, an intermediate step to complete our proof.

- *Case I:* $k_t < r$:

Since $k_t < r$, $C_{it}(\pi^*) = 0$ by definition. So, we have that $\mathbb{E}[\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*)|(\mathbf{p}_t,k_t)] = 0 \leqslant \mathbb{E}[\overline{C}_t(r-1,\tilde{\pi}^g)C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t,k_t)]$.

- *Case II:* $k_t \geqslant r$ *and* $\pi^g(i) < r$:

Given customer t's preferences $(\mathbf{p}_t,k_t)$, we know that for any ranking π that this customer faces, the event that she clicks on product i within her attention window is the Bernoulli random variable $C_{it}(\pi)$ with probability (mean) p_{it}. Therefore, for any particular instantiation of this Bernoulli random variable, it must be the case that the expression $\overline{C}_{it}(\pi^g)C_{it}(\pi^*) = 0$, since the customer is guaranteed to see product i under both rankings π^g and π^*.

This in turn implies that the quantity $\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*) = 0$ because $\overline{C}_t(r-1,\pi^g)$ is a product of several terms, one of which is $\overline{C}_{it}(\pi^g)$. Finally, it follows that:

$$\mathbb{E}[\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*)|(\mathbf{p}_t,k_t)] = 0 \leqslant \mathbb{E}[\overline{C}_t(r-1,\tilde{\pi}^g)C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t,k_t)].$$

- *Case III:* $k_t \geqslant r$ *and* $\pi^g(i) \geqslant r$:

Since the ranking π^g does not contain the item i in its first $r-1$ ranks, this implies that the random variables $\overline{C}_t(r-1,\pi^g)$ and $C_{it}(\pi^*)$ are independent of each other.

Therefore, we have that:

$$\begin{aligned}
\mathbb{E}[\overline{C}_t(r-1,\pi^g)C_{it}(\pi^*)|(\mathbf{p}_t,k_t)] &= \mathbb{E}[\overline{C}_t(r-1,\pi^g)\ |(\mathbf{p}_t,k_t)]\ \mathbb{E}[C_{it}(\pi^*)|(\mathbf{p}_t,k_t)] \\
&= \mathbb{E}[\overline{C}_t(r-1,\tilde{\pi}^g)\ |(\mathbf{p}_t,k_t)]\ \mathbb{E}[C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t,k_t)] \\
&= \mathbb{E}[\overline{C}_t(r-1,\tilde{\pi}^g)C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t,k_t)].
\end{aligned}$$

In the second equality, we used the fact that $\mathbb{E}[C_{it}(\pi^*)|(\mathbf{p}_t, k_t)] = \mathbb{E}[C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t, k_t)]$ because $\pi^*(i) = \tilde{\pi}^g(i) = r$ by definition. We also used the fact that $\mathbb{E}[\overline{C}_t(r-1, \pi^g)|(\mathbf{p}_t, k_t)] = \mathbb{E}[\overline{C}_t(r-1, \tilde{\pi}^g)|(\mathbf{p}_t, k_t)]$ since $\tilde{\pi}^g$ has the same products in its first $r-1$ ranks as π^g by definition. For the third equality, we once again relied on the independence between customer behavior on different products within the attention window and the fact that $\tilde{\pi}^g(i) = r$.

To summarize, in all three cases, we have $\mathbb{E}[\overline{C}_t(r-1, \pi^g) C_{it}(\pi^*)|(\mathbf{p}_t, k_t)] \leqslant \mathbb{E}[\overline{C}_t(r-1, \tilde{\pi}^g) C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t, k_t)]$. Plugging this back into (11), we can continue with proving the lemma.

$$\begin{aligned}
\mathbb{E}\left[\overline{C}_t(r-1, \pi^g) C_{\pi^{*-1}(r)t}(\pi^*)\right] &= \mathbb{E}_t\left[\mathbb{E}[\overline{C}_t(r-1, \pi^g) C_{it}(\pi^*)|(\mathbf{p}_t, k_t)]\right] \\
&\leqslant \mathbb{E}_t\left[\mathbb{E}[\overline{C}_t(r-1, \tilde{\pi}^g) C_{it}(\tilde{\pi}^g)|(\mathbf{p}_t, k_t)]\right] \\
&= \mathbb{E}[\overline{C}_t(r-1, \tilde{\pi}^g) C_{it}(\tilde{\pi}^g)\big] \\
&\leqslant \mathbb{E}[\overline{C}_t(r-1, \pi^g) C_{\pi^{g-1}(r)t}(\pi^g)\big]. \qquad (12)
\end{aligned}$$

The last inequality comes from the fact that for each rank r, the Greedy Algorithm for OffAR tries all of the remaining items (e.g., i' such that $\pi^g(i') > r-1$) in position r keeping the previous ranks fixed. The product having the maximum marginal utility (probability that the customer clicks on rank r but not the ones before) is selected. Mathematically, this implies that if $\pi^g(i) \geqslant r$, we have $\mathbb{E}[\overline{C}_t(r-1, \tilde{\pi}^g) C_{it}(\tilde{\pi}^g)\big] \leqslant \mathbb{E}[\overline{C}_t(r-1, \pi^g) C_{\pi^{g-1}(r)t}(\pi^g)\big]$. If $\pi^g(i) < r$, we have $\overline{C}_t(r-1, \tilde{\pi}^g) C_{it}(\tilde{\pi}^g) \leqslant \overline{C}_{it}(\tilde{\pi}^g) C_{it}(\tilde{\pi}^g) = 0$, so (12) follows trivially. □

Proof of Theorem 2: We first introduce some notation that will be helpful in our proof. Since the interest probabilities and the attention windows are independent of each other, let $\mathcal{D}_p, \mathcal{D}_k$ denote the distributions of these two parameters, respectively, such that $\mathcal{D} = \mathcal{D}_p \times \mathcal{D}_k$. For any two vectors of click probabilities, $\mathbf{p}_t, \mathbf{p}'_t$, it must then be the case that

$$\Pr_{\mathcal{D}}(k_t = k|\mathbf{p}_t) = \Pr_{\mathcal{D}}(k_t = k|\mathbf{p}'_t),$$

for all $k \in [1, n]$. As before, let π^* denote the optimal ranking and π^g denote the ranking that results from the Greedy Algorithm for OffAR for any instance where $(\mathbf{p}_t, k_t) \sim \mathcal{D}$. With slight abuse of notation and for this proof only, we let π^*_k and π^g_k denote the optimal and greedy rankings, respectively, for instances where $\mathbf{p}_t \sim \mathcal{D}_p, k_t = k$, i.e., where all customers have an attention window equal to k. Finally, for any ranking π, we use $[\pi]_r$ to denote the sub-ranking comprising only of the first r positions in π, i.e., $[\pi]_r(i) = \pi(i)$ if $\pi(i) \leqslant r$ and we define $[\pi]_r(i) = \emptyset$ otherwise.

The proof comprises of two components. First, we show that for every k, $\pi^g_k = [\pi^g]_k$, i.e. the ranking returned by the Greedy Algorithm for OffAR when $\mathbf{p}_t \sim \mathcal{D}_p, k_t = k$ is equivalent to the first k positions[8] in ranking π^g, and hence π^g is a $(1 - \frac{1}{e})$-approximation to π^*_k. Second, we prove that $\sum_{k=1}^{n} \mathbb{E}_{\mathcal{D}_p}[H_t(\pi^*_k)|k_t = k]\Pr(k_t = k) \geqslant \mathbb{E}_{\mathcal{D}}[H_t(\pi^*)]$. That is, the performance of the optimal ranking π^* is not better than a linear

[8] Technically, π^g_k contains products in positions beyond k as well but since all customers have attention windows equal to k we can safely assume that $(\pi^g_k)^{-1}(j) = \emptyset$ when $j > k$.

combination of the performance of rankings π^*_k weighted according to the probability that a customer has an attention window equal to k.

We begin by proving that π^g_k is a $(1-\frac{1}{e})$-approximation to π^*_k. Fix any k and suppose that $k_t = k$ for all t. For a constant attention window ($k_t = k$), OffAR is simply a special case of stochastic submodular optimization and therefore, we can leverage known results (Asadpour and Nazerzadeh 2015) to infer that π^g_k is a $(1-\frac{1}{e})$-approximation to π^*_k.

We next prove that π^g_k coincides with $[\pi^g]_k$ by induction, specifically, $\pi^{g-1}_k(r) = \pi^{g-1}(r)$ for every $1 \leqslant r \leqslant k$. We define $\pi^{-1}(0) = \emptyset$ for every ranking π and thus the claim is trivially true when $r = 0$. Thus, our inductive hypothesis is that $\pi^{g-1}_k(j) = \pi^{g-1}(j)$ for $j = 1, ..., r-1$. Recall that $[\pi]_{r-1}$ refers to the sub-ranking of π with only the first $r-1$ positions filled, and let $[\pi]_{r-1} \cup i$ refer to the ranking where the first $r-1$ positions coincide with π and product i is placed in the r^{th} rank. Now, the r^{th} ranked product in the greedy ranking when $k_t = k$ can be mathematically characterized as:

$$\pi^{g-1}_k(r) = \arg\max_{i\in[n]} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t([\pi^g_k]_{r-1} \cup i) - H_t([\pi^g_k]_{r-1})] = \arg\max_{i\in[n]} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})], \quad (13)$$

where the second equality follows from the inductive hypothesis. Since π^g is the greedy ranking when customer profiles are drawn from $\mathcal{D}$, the product placed in rank r is chosen according to the following equation.

$$\begin{aligned}
\pi^{g-1}(r) &= \arg\max_{i\in[n]} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t\sim\mathcal{D}_k}} [H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})] \\
&= \arg\max_{i\in[n]} \mathsf{E}_{\mathbf{p}_t\sim\mathcal{D}_p} [\mathsf{E}_{k_t\sim\mathcal{D}_k}[H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})|k_t \geqslant r]\mathsf{Pr}(k_t \geqslant r)] \\
&= \arg\max_{i\in[n]} \mathsf{Pr}(k_t \geqslant r)\mathsf{E}_{\mathbf{p}_t\sim\mathcal{D}_p} [H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})|k_t \geqslant r] \\
&= \arg\max_{i\in[n]} \mathsf{E}_{\mathbf{p}_t\sim\mathcal{D}_p} [H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})|k_t \geqslant r] \\
&= \arg\max_{i\in[n]} \mathsf{E}_{\mathbf{p}_t\sim\mathcal{D}_p} [H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})|k_t = k] \quad (14) \\
&= \arg\max_{i\in[n]} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1})] = \pi^{g-1}_k(r). \quad (15)
\end{aligned}$$

The second and third equalities are crucially dependent on the independence between the interest probabilities and attention windows. The second equality follows from the fact that when $k_t < r$, $H_t([\pi^g]_{r-1} \cup i) - H_t([\pi^g]_{r-1}) = 0$ for all $i \in [n]$. The third inequality is consequence of the notion that $\mathsf{Pr}(k_t \geqslant r)$ is a constant that is independent of $\mathcal{D}_p$. Note that we removed the expectation over $\mathcal{D}_k$ because once we condition on $k_t \geqslant r$, the inner expectation does not depend on the exact value of k_t since $H_t([\pi^g]_{r-1} \cup i)$ does not contain any products at ranks beyond r. The fourth equality follows since the index at which a variable is maximized does not change when multiplied by a constant - in this case $\mathsf{Pr}(k_t \geqslant r)$. Equality (14) is due to the fact that both $[\pi^g]_{r-1} \cup i$ and $[\pi^g]_{r-1}$ do not have any products in ranks greater than r, and therefore do not have any products in ranks greater than k, which by definition is larger than or equal to r. Equation (15) follows from (13) and completes the inductive argument. Thus, we have for all $k \in [1, n]$,

$$\mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t(\pi^g)] = \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t(\pi^g_k)] \geqslant (1-\frac{1}{e})\mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t(\pi^*_k)]. \quad (16)$$

We move on to the second part of the proof, where we relate the hook probability of π^*_k to that of π^*. Note that for the instance where $\mathbf{p}_t \sim D_p$ and $k_t = k$, π^*_k is the optimal ranking and therefore, its performance must surpass that of π^*. This gives us,

$$\mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t(\pi^*_k)] \geqslant \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}} [H_t(\pi^*)] \quad \forall k \in [1, n]$$

$$\implies \sum_{k=1}^{n} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}}[H_t(\pi_k^*)]\,\mathsf{Pr}(k_t=k) \geqslant \sum_{k=1}^{n} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}}[H_t(\pi^*)]\,\mathsf{Pr}(k_t=k)$$
$$= \mathsf{E}_{\mathbf{p}_t\sim\mathcal{D}_p}\left[\sum_{k=1}^{n} \mathsf{E}[H_t(\pi^*)|k_t=k]\mathsf{Pr}(k_t=k)\right]$$
$$= \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t\sim\mathcal{D}_k}}[H_t(\pi^*)].$$

Combining the above inequality with (16), the theorem follows. That is,

$$\mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t\sim\mathcal{D}_k}}[H_t(\pi^g)] = \sum_{k=1}^{n} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}}[H_t(\pi^g)]\mathsf{Pr}(k_t=k)$$
$$\geqslant (1-\frac{1}{e})\sum_{k=1}^{n} \mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t=k}}[H_t(\pi_k^*)]\mathsf{Pr}(k_t=k)$$
$$\geqslant (1-\frac{1}{e})\mathsf{E}_{\substack{\mathbf{p}_t\sim\mathcal{D}_p \\ k_t\sim\mathcal{D}_k}}[H_t(\pi^*)]. \quad \square$$

Proof of Theorem 3 We only prove the $\approx \frac{1}{2}$-approximation guarantee for OnAR in the case when the click probabilities and attention windows are correlated[9]; the proof for the $\approx (1-\frac{1}{e})$-approximation guarantee for OnAR with Independence follows in an almost identical fashion using the independence arguments outlined in the proof of Theorem 2, so we avoid reproving it to minimize redundancy. Recall that $\tilde{\pi}$ denotes the output of Algorithm 2 while π^* is the optimal ranking. As in the proof of Theorem 2, suppose that for any ranking π, $[\pi]_r$ is the sub-ranking of π such that $[\pi]_r(i) = \pi(i)$ if $\pi(i) \leqslant r$ and we define $[\pi]_r(i) = \emptyset$ otherwise. Finally, let $[\pi]_r \cup i$ be an augmentation of $[\pi]_r$ such that its first r positions coincide with π and product i is placed in rank $r+1$.

Now, as per Definition 1 and Algorithm 2, we can infer that for any given rank r, the product $\tilde{\pi}^{-1}(r)$ is a $(\frac{2\epsilon}{n}, \frac{\delta}{n})$-PAC approximation to the best product at rank r subject to the products already fixed at previous ranks. Formally, with probability $1-\frac{\delta}{n}$, the following inequality holds for each $r \in [1, n]$:

$$\mathsf{E}[\overline{C}_t(r-1,\tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})] \geqslant \max_{i\in[n]} \mathsf{E}[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1}\cup i)] - \frac{2\epsilon}{n}, \tag{17}$$

We note that even though $\tilde{\pi}$ is itself a random ranking due to sampling differences, (17) holds for any particular instantiation of $[\tilde{\pi}]_{r-1}$ with probability $1-\frac{\delta}{n}$. Recall that $\overline{C}_t(r-1,\tilde{\pi}) = 1$ when customer t does not click on any of the first $r-1$ ranked products in $\tilde{\pi}$ conditional on $(\mathbf{p}_t, k_t)$. Therefore, the left hand side of the above inequality is the probability that an incoming customer t is 'hooked' by the r^{th} ranked product in $\tilde{\pi}$ for some fixed instantiation of $[\tilde{\pi}]_{r-1}$. Suppose that for all r, B_r denotes the event that (17) holds for index r, i.e., $B_r = 1$ if (17) is true and is zero if the inequality is not true. We have $\mathsf{Pr}(B_r = 0) \leqslant \frac{\delta}{n}$ for all $r \in [1, n]$.

From the union bound, we have that:

$$\mathsf{Pr}(B_r = 1 \quad \forall r) = 1 - \mathsf{Pr}(\exists r : B_r = 0) \geqslant 1 - \sum_{r=1}^{n} \mathsf{Pr}(B_r = 0) \geqslant 1 - n\frac{\delta}{n} = 1-\delta. \tag{18}$$

In simple terms, by applying the union bound, we can infer that with probability $1-\delta$, the ranking output by our algorithm satisfies the following condition: *for all* $r \in [n]$*, the product at rank* r *in* $\tilde{\pi}$ *is approximately*

[9] i.e., $\mathcal{D}$ cannot be decomposed into two separate distributions as in the proof of Theorem 2.

optimal for that rank with respect to the products fixed at previous ranks. For the rest of this proof, we only concern ourselves with rankings $\tilde{\pi}$ output by our algorithm where $B_r = 1$ for all r, which happens to be the case with probability $1-\delta$.

Specifically, let us fix the ranking $\tilde{\pi}$ returned by Algorithm 2 such that this ranking satisfies the condition in (18). For any such fixed ranking $\tilde{\pi}$, we aim to show that $\mathbb{E}[H_t(\tilde{\pi})] \geqslant \frac{1}{2}\mathbb{E}[H_t(\pi^*)] - \epsilon$, or equivalently, $\mathbb{E}[H_t(\pi^*)] \leqslant 2\mathbb{E}[H_t(\tilde{\pi})] + 2\epsilon$. By applying (7) from the proof of Theorem 1 with $\pi = \tilde{\pi}$, we get that:

$$E[H_t(\pi^*)] \leqslant \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{\pi^{*-1}(r)t}(\pi^*)\big]. \tag{19}$$

Fix some rank r and suppose that $\pi^{*-1}(r) = i^*$. Now proceeding exactly as in Lemma 1, we have

$$\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{i^*t}(\pi^*)\big] \leqslant \mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{i^*t}([\tilde{\pi}]_{r-1} \cup i^*)\big].$$

Finally, we leverage (17) to obtain

$$\begin{aligned}
\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{i^*t}(\pi^*)\big] &\leqslant \mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{i^*t}([\tilde{\pi}]_{r-1} \cup i^*)\big] \\
&\leqslant \max_{i\in[n]} \mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1} \cup i)\big] \\
&\leqslant \mathbb{E}[\overline{C}_t(r-1,\tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})] + \frac{2\epsilon}{n}.
\end{aligned} \tag{20}$$

Substituting (20) into (19), we have

$$\begin{aligned}
E[H_t(\pi^*)] &\leqslant \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{\pi^{*-1}(r)t}(\pi^*)\big] \\
&\leqslant \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n} \left(\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})\big] + \frac{2\epsilon}{n}\right) \\
&= \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n} \mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})\big] + 2\epsilon \\
&= \mathbb{E}[H_t(\tilde{\pi})] + \mathbb{E}[H_t(\tilde{\pi})] + 2\epsilon \qquad (21) \\
&= 2\mathbb{E}[H_t(\tilde{\pi})] + 2\epsilon
\end{aligned}$$

where (21) follows by applying (3). □

Proof of Theorem 4 Without loss of generality, we assume that $\log_{1+\alpha}(\frac{n}{\epsilon})$ is integral, and so during the course of the algorithm, $\tau \in \{1, \frac{1}{1+\alpha}, \frac{1}{(1+\alpha)^2}, \ldots, \frac{\epsilon}{n}\}$[10]. For the purpose of this proof, we will also implicitly assume that the parameters α, ϵ, δ are chosen such that $n \geqslant \log_{1+\alpha}(\frac{n}{\epsilon})$. Indeed, when this is not the case, the length of the learning phase for the Threshold Acceptance Algorithm becomes $\Theta(n^4)$ and it is advantageous to run Algorithm 2 instead.

Recall that for a fixed threshold τ, our algorithm takes one pass through the set of available products and fixes any product at the smallest open rank as long as its marginal benefit at that rank is at least τ.

[10] Since $\log_{1+\alpha}(\frac{n}{\epsilon})$ is integral, this implies that there exists an integer a such that $\tau^{\max}\big(\frac{1}{1+\alpha}\big)^a = \tau^{\min} = \frac{\epsilon}{n}$. Therefore, Algorithm 3 tries out exactly $a+1$ different values of τ belonging to the set $\{1, \frac{1}{1+\alpha}, \frac{1}{(1+\alpha)^2}, \ldots, \frac{\epsilon}{n}\}$.

Therefore, for a given threshold, the algorithm could assign multiple products to successive ranks as long as the marginal benefit of all of these products meet the threshold at the corresponding ranks. Subsequently, the threshold is lowered to $\frac{1}{1+\alpha}\tau$ for the given parameter $\alpha > 0$, and the above process is repeated on the remaining, unranked products.

We first introduce pertinent notation. For any rank $r \in [1, n]$, let τ_r be the threshold value that the product $\tilde{\pi}^{-1}(r)$ had exceed in order to be assigned to the corresponding rank. Conversely, let $r^{(\tau)}$ be the smallest rank at which our algorithm considers the threshold τ or, equivalently, the rank when the algorithm transitions from a threshold of $(1+\alpha)\tau$ to τ. Due to the fact that the thresholds are monotonically decreasing, this implies that for all $r' < r^{(\tau)}$, $\tau_{r'} > \tau$. As in earlier proofs, suppose that for any ranking π, $[\pi]_r$ is the sub-ranking of π such that $[\pi]_r(i) = \pi(i)$ if $\pi(i) \leqslant r$ and we define $[\pi]_r(i) = \emptyset$ otherwise. Finally, let $[\pi]_r \cup i$ be an augmentation of $[\pi]_r$ such that its first r positions coincide with π and product i is placed in rank $r+1$.

In the first part of our proof, we will use Hoeffding's inequality to bound the expected number of new customers hooked at each rank. We move the details of Hoeffding's inequality to Lemmas 2 and 3 immediately following this proof for ease of exposition. Consider the product $\tilde{\pi}^{-1}(r)$ that our algorithm fixes at rank r. We know that ranking $[\tilde{\pi}]_r$ was displayed to $L = \frac{n^2}{\epsilon^2}\log(\frac{n}{\sqrt{\delta}})$ customers, and at least a fraction τ_r of these customers clicked exclusively on product $\tilde{\pi}^{-1}(r)$. Consider the Bernoulli random variable $X = \overline{C}_t(r-1, \tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})$ which equals one if the customer clicks on the product at rank r without clicking on any of the previously ranked products and is zero otherwise. We have $\frac{1}{L}\sum_{i=1}^{L} X_i \geqslant \tau_r$, where X_i denotes the i–th sample of the Bernoulli random variable X corresponding to customer $i \leqslant L$. Applying Lemma 2, we have that with probability at least $1 - \frac{\delta}{n^2}$, the following inequality is true:

$$\mathbb{E}\left[\overline{C}_t(r-1, \tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})\right] \geqslant \frac{1}{L}\sum_{i=1}^{L} X_i - \frac{\epsilon}{n} \geqslant \tau_r - \frac{\epsilon}{n}. \tag{22}$$

Note that even though $[\tilde{\pi}]_{r-1}$ is itself a random ranking, the above inequality holds with probability at least $1 - \frac{\delta}{n^2}$ for any particular instantiation of $[\tilde{\pi}]_{r-1}$. Let $U^{(\tau)}$ be the set of unranked products when the algorithm lowers the threshold from $(1+\alpha)\tau$ to τ. For any $i \in U^{(\tau)}$, there must exist some rank $r' \leqslant r^{(\tau)}$ at which L customers were shown the ranking $[\tilde{\pi}]_{r'-1} \cup i$ and the fraction that clicked on product i without clicking on any of the previously ranked products was smaller than $(1+\alpha)\tau$. Applying Lemma 3, we have that with probability at least $1 - \frac{\delta}{n^2}$

$$\mathbb{E}\left[\overline{C}_t(r'-1, \tilde{\pi})C_{it}([\tilde{\pi}]_{r'-1} \cup i)\right] \leqslant \frac{1}{L}\sum_{j=1}^{L} X_j + \frac{\epsilon}{n} < (1+\alpha)\tau + \frac{\epsilon}{n},$$

where $(X_j)_{j=1}^{L}$ is the instantiation of the random variable $\overline{C}_t(r'-1, \tilde{\pi})C_{it}([\tilde{\pi}]_{r'-1} \cup i)$ when the ranking $[\tilde{\pi}]_{r'-1} \cup i$ was displayed to each of the L customers. Since $r' \leqslant r^{(\tau)}$, we also have that $\mathbb{E}\left[\overline{C}_t(r^{(\tau)}-1, \tilde{\pi})C_{it}([\tilde{\pi}]_{r^{(\tau)}-1} \cup i)\right] \leqslant \mathbb{E}\left[\overline{C}_t(r'-1, \tilde{\pi})C_{it}([\tilde{\pi}]_{r'-1} \cup i)\right]$ by monotonicity. Thus, for all τ and all $i \in U^{(\tau)}$, the following inequality holds with probability at least $1 - \frac{\delta}{n^2}$:

$$\mathbb{E}\left[\overline{C}_t(r^{(\tau)}-1, \tilde{\pi})C_{it}([\tilde{\pi}]_{r^{(\tau)}-1} \cup i)\right] \leqslant (1+\alpha)\tau + \frac{\epsilon}{n}. \tag{23}$$

Once again, we highlight that the above inequality holds for any particular instantiation of $[\tilde{\pi}]_{r^{(\tau)}-1}$. Let us use $E_i^{(\tau)}$ to denote the event (indicator variable) that (23) is true for a given τ and $i \in U^{(\tau)}$ for the ranking

output by our algorithm. We know that $\Pr(E_i^{(\tau)}=0)\leqslant\frac{\delta}{n^2}$. Analogously let F_r be the event that (22) is true for a given rank r at which $\tilde{\pi}^{-1}(r)\neq\emptyset$. We also have $\Pr(F_r=0)\leqslant\frac{\delta}{n^2}$. Applying the union bound as we did in the proof of Theorem 3, we get that for a given τ:

$$\begin{aligned}\Pr\Big(\exists i\in U^{(\tau)}\text{ s.t. }E_i^{(\tau)}=0\bigcup\exists r:\tau_r=(1+\alpha)\tau\text{ s.t. }F_r=0\Big)&\leqslant\sum_{i\in U^{(\tau)}}\Pr(E_i^{(\tau)}=0)+\sum_{\tau_r=(1+\alpha)\tau}^{r}\Pr(F_r=0)\\&\leqslant\sum_{i\in U^{(\tau)}}\frac{\delta}{n^2}+\sum_{\tau_r=(1+\alpha)\tau}^{r}\frac{\delta}{n^2}\\&\leqslant n\Big(\frac{\delta}{n^2}\Big)=\frac{\delta}{n}.\end{aligned}$$

The final inequality comes from the fact that $|U^{(\tau)}|+|r|\tau_r=(1+\alpha)\tau|\leqslant n$ because the products that are fixed in the ranks specified by the set $\{r\mid\tau_r=(1+\alpha)\tau\}$ must by definition belong to the set $[n]\setminus U^{(\tau)}$. We can apply the above inequality over all feasible threshold values τ to get

$$\Pr\Big(\exists\tau:\{\exists i\in U^{(\tau)}\text{ s.t. }E_i^{(\tau)}=0\bigcup\exists r:\tau_r=(1+\alpha)\tau\text{ s.t. }F_r=0\}\Big)\leqslant\log_{1+\alpha}(\frac{n}{\epsilon})\frac{\delta}{n}\leqslant\delta. \tag{24}$$

Here, we used the fact that the number of feasible values of τ is exactly $\log_{1+\alpha}(\frac{n}{\epsilon})$, which by our assumption is no larger than n. Informally, (24) gives an upper bound on the (error) probability that the empirical marginal click probabilities estimated by our algorithm are not close to the true click probabilities for all thresholds and all ranks and products corresponding to those thresholds.

Now that we have bounded the expected number of new customers hooked at each rank, for the rest of this proof we will consider the regime where these empirical estimates are close to their expected values, i.e. we only consider rankings $\tilde{\pi}$ output by our algorithm such that $E_i^{(\tau)}=1\ \forall i\in U^{(\tau)}\bigcap F_r=1\ \forall r:\tau_r=(1+\alpha)\tau$ for all τ. We know that with probability $1-\delta$, our algorithm returns such a ranking. We aim to show that in this regime, any ranking $\tilde{\pi}$ computed by our algorithm satisfies: $\mathbb{E}[H_t(\tilde{\pi})]\geqslant\frac{1}{2+\alpha}\mathbb{E}[H_t(\pi^*)]-\epsilon$.

Consider a fixed ranking $\tilde{\pi}$ returned by our algorithm that satisfies the conditions mentioned above. We apply (7) from the proof of Theorem 1 with $\pi=\tilde{\pi}$ to get that

$$\begin{aligned}\mathbb{E}[H_t(\pi^*)]&\leqslant\mathbb{E}[H_t(\tilde{\pi})]+\sum_{r=1}^{n}\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{\pi^{*-1}(r)t}(\pi^*)\big]\\&\leqslant\mathbb{E}[H_t(\tilde{\pi})]+\sum_{r=1}^{n}\max_{i\in[n]}\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1}\cup i)\big]\\&=\mathbb{E}[H_t(\tilde{\pi})]+\sum_{r=1}^{n}\max_{i\in U^{(\tau_r)}}\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1}\cup i)\big] &(25)\\&\leqslant\mathbb{E}[H_t(\tilde{\pi})]+\sum_{r=1}^{n}\Big((1+\alpha)\tau_r+\frac{\epsilon}{n}\Big). &(26)\end{aligned}$$

For (25), we replaced the set $[n]$ by $U^{(\tau_r)}$, the latter being a (superset) of the set of unassigned products when the algorithm considers the rank r. Note that for all $i\notin U^{\tau_r}$, $\tilde{\pi}(i)<r$ and so, $\mathbb{E}\big[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1}\cup i)\big]=0$. Finally, (26) comes from (23).

Now, suppose that Algorithm 3 returns a ranking $\tilde{\pi}$, with products assigned to ranks one through $n'<n$. In this case, we can once again apply (23) and infer that the marginal benefit for all of the unranked products at rank $n'+1$ must be strictly smaller than $\tau^{\min}+\frac{\epsilon}{n}$ - if this were not the case, these products would have

already been fixed at rank $n'+1$ or earlier. Thus our algorithm ends by updating $\tau_{n'+1} = \frac{\tau^{min}}{1+\alpha}$; equivalently, when $r > n'$, $(1+\alpha)\tau_r = \frac{\epsilon}{n}$ since $\tau^{min} = \frac{\epsilon}{n}$ in the theorem statement. We now continue with (26):

$$\begin{aligned}
\mathbb{E}[H_t(\pi^*)] &\leqslant \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n} \left((1+\alpha)\tau_r + \frac{\epsilon}{n} \right) \\
&= \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n'} (1+\alpha)(\tau_r) + \sum_{r=n'+1}^{n} (1+\alpha)(\tau_r) + \epsilon \\
&= \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n'} (1+\alpha)(\tau_r) + (n-n')\frac{\epsilon}{n} + \epsilon \\
&\leqslant \mathbb{E}[H_t(\tilde{\pi})] + \sum_{r=1}^{n'} (1+\alpha)(\mathbb{E}\left[\overline{C}_t(r-1,\tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})\right] + \frac{\epsilon}{n}) + (n-n')\frac{\epsilon}{n} + \epsilon \qquad (27) \\
&\leqslant \mathbb{E}[H_t(\tilde{\pi})] + (1+\alpha)\sum_{r=1}^{n'} \mathbb{E}\left[\overline{C}_t(r-1,\tilde{\pi})C_{\tilde{\pi}^{-1}(r)t}(\tilde{\pi})\right] + (1+\alpha)\epsilon + \epsilon \\
&= \mathbb{E}[H_t(\tilde{\pi})] + (1+\alpha)\mathbb{E}[H_t(\tilde{\pi})] + (2+\alpha)\epsilon
\end{aligned}$$

Inequality (27) comes from (22). In the final step, we used (3).

To conclude, we are left with

$$\mathbb{E}[H_t(\pi^*)] \leqslant (2+\alpha)\mathbb{E}[H_t(\tilde{\pi})] + (2+\alpha)\epsilon,$$

which in turn implies the statement of the theorem. □

The following lemmas are used in the preceding proof of Theorem 4, and both their proofs follow from the standard Hoeffding inequality.

LEMMA 2. *Given* $i \in [n]$ *and* $r \in [n]$*, suppose that* $X_1, X_2, \ldots, X_L$ *are independent samples of the random variable* $\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1} \cup i)$*. Then, we have that:*

$$\Pr\left(\sum_{j=1}^{L} \frac{X_j}{L} - \mathbb{E}\left[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1} \cup i)\right] > \frac{\epsilon}{n} \right) \leqslant \exp(-2\frac{\epsilon^2}{n^2}L).$$

LEMMA 3. *Given* $i \in [n]$ *and* $r \in [n]$*, suppose that* $X_1, X_2, \ldots, X_L$ *are independent samples of the random variable* $\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1} \cup i)$*. Then, we have that:*

$$\Pr\left(\mathbb{E}\left[\overline{C}_t(r-1,\tilde{\pi})C_{it}([\tilde{\pi}]_{r-1} \cup i)\right] - \sum_{j=1}^{L} \frac{X_j}{L} > \frac{\epsilon}{n} \right) \leqslant \exp(-2\frac{\epsilon^2}{n^2}L).$$

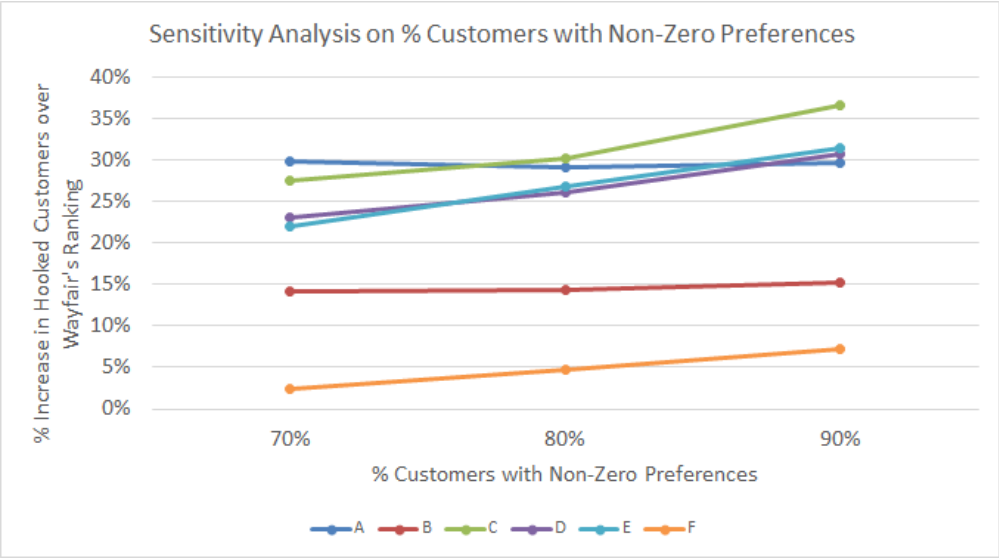

Figure 6 **Sensitivity Analysis on Percent of Customers with Non-Zero Preferences**

Appendix B: Sensitivity Analysis for Simulations

In this appendix, we perform sensitivity analysis on the default parameter set (recommended by Wayfair) that we used to generate customer preference vectors from observed clickstream data. First, we varied the percent of customers with non-zero preferences between 70-90% (specifically, {70%, 80%, 90%} - recall that the main results presented are for 80%). Figure 6 presents our sensitivity analysis on the percent of customers with non-zero preferences. The analysis illustrates that our results are robust to this parameter choice, and in most cases, that our algorithm performs marginally better as the percent of customers with non-zero preferences increases. Intuitively, when there are more customers with non-zero preferences, our algorithm has more opportunity to hook additional customers compared to Wayfair's ranking where the number of hooked customers is fixed. In reality, it is impossible to know the percent of customers with non-zero preferences, and our results show that over a reasonable and wide parameter range, our algorithm still attains significant performance improvement over Wayfair's ranking.

Second, we varied the percent of customers who view all n products between 2.5-10% (specifically, {2.5%, 5%, 7.5%, 10%} - recall that the main results presented are for 5%). When the percent of customers who view all products was 2.5% or 10%, we were unable to find a power law distribution to generate customers that recovered the actual clickstream data for some of the events, i.e. we could not find matches between actual and simulated clicks as we did in Figure 2; thus, these parameter values are likely unrealistic. Figure 7 presents our sensitivity analysis on the percent of customers who view all products. The analysis illustrates that our results are robust to this parameter choice. In reality, it is impossible to know the percent of customers who view all products, and our results show that over a reasonable parameter range, our algorithm still attains significant performance improvement over Wayfair's ranking.

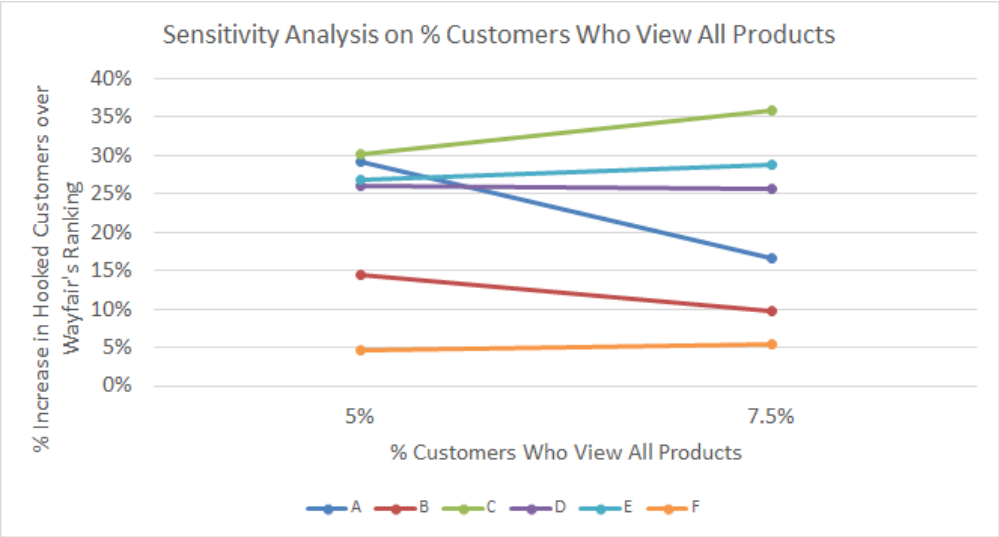

Figure 7 **Sensitivity Analysis on Percent of Customers Who View All Products**

Appendix C: Additional Simulations: Finite Customer Types

In Section 5, we presented our main empirical results, where we evaluated the performance of our Threshold Acceptance Algorithm on customer preferences inferred based on actual clickstream data from six distinct events. Despite the rather conservative assumptions made in that section, we observed that our proposed algorithm outperformed the default static ranking employed by Wayfair by moderate amounts (average improvement between 5 and 30%; refer to Figure 3). In this section, we adopt an alternative and less-conservative methodology where we assume that customers belong to a finite number of types and use the clickstream data to infer the types and the probability that an incoming customer belongs to each of these types. We then use the derived customer preference distribution to estimate the performance of our method in comparison to Wayfair's static ranking as well as the omniscient offline benchmark as defined earlier (the static ranking output by the Greedy Algorithm for OffAR, π^g).

We begin by highlighting a few of the restrictive assumptions from Section 5 below and outline our reasoning for relaxing these assumptions. Following this, we present the technical details surrounding the implementation. A comprehensive description of our partner Wayfair, customer behavior on the platform, characteristics of the six stylistically-themed events that were chosen for these simulations, and the clickstream data can be found in Section 5.1, and we avoid repeating these details here.

In the process of learning customer preferences and attention windows from the clickstream data, our implementation in Section 5 required the following behavioral assumptions. First, our inference was based on the premise that once a customer was hooked, she views all 48 products in the event. Consequently, each unique click vector was taken to represent a distinct customer type[11]; this resulted in each event catering to

[11] Formally, a preference distribution $\mathcal{D}$ can be partitioned into mutually exclusive customer types, where each type z is characterized by an interest probability vector $\mathbf{p}^{(z)}$, attention window distribution $\mathcal{D}_k^{(z)}$ and a type probability $q^{(z)}$. An incoming customer t belongs to type z with probability $q^{(z)}$ and conditional upon her type being z, her click vector is $\mathbf{p}_t = \mathbf{p}^{(z)}$ and attention window k_t is drawn independently from $\mathcal{D}_k^{(z)}$.

anywhere between 3500 and 14000 customer types. Second, the customer interest probabilities were assumed to be independent of their attention windows. Third, for each customer t and product i, we had only integral click probabilities, i.e., $p_{it} \in \{0, 1\}$. These assumptions are somewhat conservative as they tend to be inimical to the performance of our algorithm. For example, the premise that hooked customers view all 48 products severely limits the customer's response to any alternative ranking as the customer is assumed to not be interested in any product that she did not click on.

In the following simulations, we make a different set of assumptions regarding customer interests and behavior and similarly compare our algorithm to Wayfair's static ranking and offline benchmark:

- We suppose that there is a finite set $\mathcal{Z}$ of customer types, where each type $z \in \mathcal{Z}$ is specified by its parameters $(\mathbf{p}^{(z)}, \mathcal{D}_k^{(z)}, q^{(z)})$. Crucially, the distribution of customer attention windows is not independent of product interest. This is a natural consideration in e-commerce domains, e.g., the demographics of customers may skew both their product preferences and their attention windows. Finally, we assume that the average number of products per type that a customer has non-zero interest in is two. Mathematically, we have:

$$\frac{1}{|\mathcal{Z}|} \sum_{z \in \mathcal{Z}} \sum_{i \in [n]} \mathbb{1}\{p_i^{(z)} > 0\} = 2,$$

where $\mathbb{1}\{\}$ is the indicator variable that is one if the condition inside is true and zero otherwise. Note that some types may have only one product and others may have many.

- There exists a customer click probability parameter $p \in (0, 1)$ such that $p_{it} \in \{0, p\}$ for all customers t and products $i \in [1, n]$. A simple interpretation for this parameter is that a customer t who is interested in a product i that falls within her attention window only clicks on this product with some finite probability p and does not click on this product with probability $1 - p$.

- Customer t only clicks on products displayed in ranks $[1, k_t]$, regardless of whether or not they are hooked. This assumption is in contrast with the one made in Section 5 that hooked customers browse and can click on products in the entire assortment. In reality, customer behavior and the performance of our algorithm may fall somewhere in-between.

C.1. Implementation

In Section 5, the primary difficulty in the inference process was to identify the (censored) interest probability vector $\mathbf{p}_t$ for customers who did not click on any product by assigning them to one of the click vectors corresponding to the hooked customers. Here, the challenge is two-fold: first, we need to infer a representative set of customer types including their preferences and attention windows; second, we need to assign each of the customers to one of the types based on the observed click vector. Once again, the second part is compounded by the censored nature of the observations for all customers (hooked or otherwise).

We now outline the methodology used for constructing the empirical distribution $\hat{\mathcal{D}}$ of customer types based on the clickstream data. Where it is applicable, we use the same notation introduced in Section 5, e.g., $\mathbf{c}_t$ represents the click vector for customer t, $\tilde{\mathcal{T}}_c$ and $\tilde{\mathcal{T}}$ are the customers with non-zero clicks and non-zero preference vectors respectively, and so on. To reiterate, our goal is to use the set of customer click vectors $(\mathbf{c}_t)_{t \in \mathcal{T}}$ to construct the customer types that make up the distribution $\hat{\mathcal{D}}$, i.e., $(\mathbf{p}^{(z)}, \mathcal{D}_k^{(z)}, q^{(z)})_{z \in \mathcal{Z}}$.

1. *Fixing parameters a priori*: As we did in Section 5, we assume that the total number of customers with non-zero preferences $|\tilde{\mathcal{T}}|$ is fixed, as is the customer click probability parameter p and the number of types $|\mathcal{Z}|$. We will vary the parameter p within an appropriate range to recover the customer types that best fit the observed click data. The exact choice or range of the other parameters is discussed subsequently.

2. *Clustering to partition click vectors into types*: We identify a subset of the customers in $\tilde{\mathcal{T}}_c$ who clicked on two or more products ($\sum_{i\in[n]} c_{it} > 1$) and run a standard k-means++ (Arthur and Vassilvitskii 2007) algorithm to cluster these click vectors into $|\mathcal{Z}|$ partitions.

3. *Identifying representative set of interest vectors* $(\mathbf{p}^{(z)})_{z\in\mathcal{Z}}$: We extract the centroids of each cluster $(\mathbf{C}^{(z)})_{z\in\mathcal{Z}}$, which are fractional and tend to be overly dispersed as each centroid has non-zero entries corresponding to many products. Thus, we take each product $i\in[n]$ and assign it to the w clusters corresponding to the w largest values in $(C_i^{(z)})_{z\in\mathcal{Z}}$. Therefore, $p_i^{(z)} = p$ if z is one of the top-w clusters for product $i\in[n]$ as per the centroid, and $p_i^{(z)} = 0$ otherwise. Finally, the parameter w is chosen in order to satisfy our assumption that the average number of products that a customer type is interested in is two, i.e., $w = \frac{2|\mathcal{Z}|}{48}$.

4. *Assigning customers (click vectors) to types:* Now that we have derived the interest vectors $\mathbf{p}^{(z)}$ for each type, we need to infer the attention window distributions as well as type probabilities. As a first step, we take each click vector belonging to the customers in $\tilde{\mathcal{T}}_c$ and assign it to one of the types using a maximum likelihood technique. Specifically, consider a click vector $\mathbf{c}_t$ and let r_t denote the position of the largest ranked product that the customer clicks on. For every type $z\in\mathcal{Z}$, customer $t\in\tilde{\mathcal{T}}_c$, and product $i\in[r_t]$, we define a probabilistic score $s_{it}^{(z)}$ as follows: (i) $s_{it}^{(z)} = p$ if $c_{it}, p_i^{(z)} > 0$; (ii) $s_{it}^{(z)} = 1-p$ if $p_i^{(z)} > 0 = c_{it}$; (iii) $s_{it}^{(z)} = \eta$ if $p_i^{(z)} = 0 < c_{it}$ and (iv) $s_{it}^{(z)} = 1$ if $p_i^{(z)} = c_{it} = 0$. In the above definition $\eta < \min\{p, 1-p\}$ is a penalty term corresponding to the case where a customer clicks on a product that a type is not interested in. The type $z(t)$ that a customer t is assigned to is the solution to the following expression that captures the likelihood of belonging to each type given the observed click vector $\mathbf{c}_t$:

$$z(t) = \arg\max_{z\in\mathcal{Z}} \prod_{i=1}^{r_t} s_{it}^{(z)}. \tag{28}$$

5. *Joint, iterated Bayesian inference of attention windows and type probabilities*: We use an iterative approach that starts with an initial assumption (uniform distribution) on the attention windows and type probabilities and jointly and iteratively updates both of these parameters until the process converges. Specifically, suppose that $(\hat{\mathcal{D}}_k^{(z)}, \hat{q}^{(z)})_{z\in\mathcal{Z}}$ are the current estimates for the corresponding distributions in a given iteration. We use the following two steps to update these parameters:

- For every $t\in\tilde{\mathcal{T}}_c$, we identify $z(t)$ as per (28). For the customers who did not click on any product but have non-zero preferences ($t\in\tilde{\mathcal{T}}\setminus\tilde{\mathcal{T}}_c$) we randomly assign them to a type $z(t)\in\mathcal{Z}$ using Bayes rule, which in turn is dependent on $(\hat{\mathcal{D}}_k^{(z)}, \hat{q}^{(z)})$:

$$\Pr(z(t) = z|\mathbf{c}_t = 0) = \frac{\hat{q}^{(z)}}{\frac{|\tilde{\mathcal{T}}\setminus\tilde{\mathcal{T}}_c|}{\tilde{\mathcal{T}}}} \sum_{k\in[n]} \Pr(\mathbf{c}_t = 0|\mathbf{p}^{(z)}, k_t = k)\Pr(k_t = k|\hat{\mathcal{D}}_k^{(z)}).$$

Once every customer t is assigned to a type $z(t)$, we can update $\hat{q}^{(z)}$ to be the fraction of customers assigned to type z.

- Given $z \in \mathcal{Z}$, for every customer $t \in \tilde{\mathcal{T}}$ such that $z(t) = z$, we compute the probability that the customer's attention window is k given her type z. That is for all $k \geqslant r_t$, we can use the (conditional) distribution $\hat{\mathcal{D}}_k^{(z)}$ to infer the probability that her attention window is $k_t = k$ subject to $k \geqslant r_t$, where r_t is the rank of the last product that the customer clicked on ($r_t = 0$ if $t \in \tilde{\mathcal{T}} \setminus \tilde{\mathcal{T}}_c$).

$$\Pr(k_t = k | z(t) = z, k \geqslant r_t) = \frac{\Pr(k_t = k | \hat{\mathcal{D}}_k^{(z)})}{\sum_{r=r_t}^{n} \Pr(k_t = r | \hat{\mathcal{D}}_k^{(z)})}.$$

Finally, we can update the distribution $\hat{\mathcal{D}}_k^{(z)}$ by aggregating the probability that $\Pr(k_t = k | z(t) = z)$ over all $k \in [n]$ and all t such that $z(t) = z$.

We terminate the iterative process when the root mean square errors of the values of both $\hat{\mathcal{D}}_k^{(z)}$ and $\hat{q}^{(z)}$ between successive rounds are smaller than a specified tolerance level

Parameter Choices First, we assume that 70% of the customers have non-zero preference vectors, i.e., $\frac{|\tilde{\mathcal{T}}|}{|\mathcal{T}|} = 0.7$. This reflects a conservative modeling choice that is at the lower end of the range of this parameter that we analyzed in Appendix B. By assuming that 30% of the customers cannot be hooked by any ranking, we seek to compensate for our other, less conservative assumptions. Second, for the rest of this section, we fix the number of customer types or clusters to be $|\mathcal{Z}| = 75$; we tested and found similar results for other values such as $|\mathcal{Z}| = \{25, 50, 100\}$ but present only $|\mathcal{Z}| = 75$ for brevity. This is in stark contrast to the experiments in Section 5 with thousands of types and only a handful of customers per type. Finally, we assume that the customer click probability $p \in [0.5, 0.75]$, which corresponds to an average click probability over all products of $\frac{2p}{48} \approx [0.02, 0.03]$ per Wayfair's recommendation.

C.2. Results

The implementation of the Threshold Acceptance Algorithm is identical to our earlier simulations and we refer the reader to Section 5.2.2 for a detailed description. As in Section 5, we conducted the following simulations 100 times for each event. First, we generated $|\tilde{\mathcal{T}}|$ customers from the inferred distribution $\hat{\mathcal{D}} = (\mathbf{p}^{(z)}, \hat{\mathcal{D}}_k^{(z)}, \hat{q}^{(z)})_{z \in \mathcal{Z}}$. We applied our Threshold Acceptance Algorithm for each customer profile that was generated assuming that customers arrive in a random order and then compared the algorithm's performance to that of Wayfair's static ranking (π^{WF}) and the offline benchmark static ranking. Owing to a judicious choice of the parameters mentioned earlier, the click behavior of customers on Wayfair's ranking was a close fit to the true click distribution observed in the data. For example, in all of our simulations, the percent of customers hooked by Wayfair's ranking was within $[0.98, 1.06]$ times the actual percentage of customers hooked.

Figure 8 compares the performance of our algorithm and π^{WF} to the static ranking output by the Greedy Algorithm for OffAR (π^g): specifically, (total number of hooked customers by our algorithm (or π^{WF})) / (total number of hooked customers by π^g). We see that our Threshold Acceptance Algorithm yields a significant improvement over Wayfair's static ranking. Averaged over 100 simulations, the number of customers hooked by our algorithm was at least a multiplicative factor of 1.5-times the number hooked by Wayfair's ranking for all six events. There are two main reasons why we see such a significant improvement over Wayfair's ranking. First, unlike in Section 5, here we allow product interests to be correlated with attention windows.

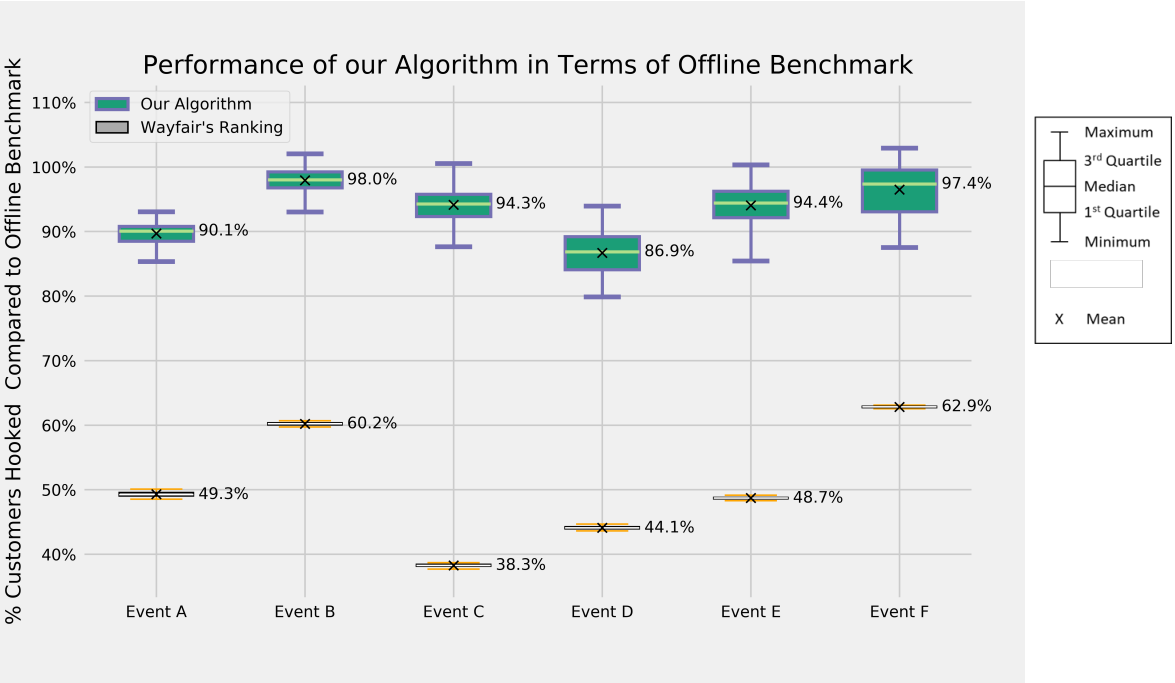

Figure 8 **Box & whisker plots presenting the percent of total customers hooked by π^g (the offline benchmark) that are hooked by our algorithm and π^{WF}**

Our Threshold Acceptance Algorithm is able to exploit these dependencies better than Wayfair's static, popularity-based ranking. Second, our assumptions on browsing and clicking behavior for these simulations are less conservative than those in Section 5, and provide more opportunities for our algorithm to hook those customers who were not hooked by Wayfair's ranking. It is likely that the actual improvement in practice lies somewhere between these results and the more conservative ones in Section 5.

Another key finding is that our Threshold Acceptance Algorithm's performance is comparable to the offline benchmark: On average over the six events, the number of customers hooked by our algorithm is at least 87% of the amount hooked by π^g. We note that the variability in the performance of our Threshold Acceptance Algorithm is primarily due to the randomness that comes from sampling only $L = 500$ customers to estimate the marginal benefit of each product. A surprising by-product of this variability is that in a few instances, our algorithm actually outperforms the benchmark ranking, by up to 3%. Indeed, it is worth noting that since the benchmark ranking π^g is computed on the full distribution $\hat{\mathcal{D}}$, its performance can vary considerably on individual populations that are randomly sampled from $\hat{\mathcal{D}}$. Beyond randomness, this can also occur due to the fact that the offline greedy algorithm is only a $\frac{1}{2}$-approximation to the true optimal ranking[12]. As a result, our algorithm may converge to a ranking better than π^g by making some locally suboptimal decisions.

[12] Recall that the true optimal ranking is NP-Hard to compute even for a special case of our problem