Fairness of Interaction in Ranking under Position, Selection, and Trust Bias

ZOHREH OVAISI, University of Illinois Chicago, Chicago, United States
PARSA SAADATPANAH, Meta Inc, Washington DC, United States
SHAHIN SEFATI, Meta Inc, New York, United States
MESROB OHANNESSIAN, University of Illinois Chicago, Chicago, United States
ELENA ZHELEVA, University of Illinois Chicago, Chicago, United States

Ranking algorithms in online platforms serve not only users on the demand side, but also items on the supply side. While ranking has traditionally presented items in an order that maximizes their utility to users, the uneven interactions that different items receive as a result of such a ranking can pose item fairness concerns. Moreover, interaction is affected by various forms of bias, two of which have received considerable attention: position bias and selection bias. Position bias occurs due to lower likelihood of observation for items in lower ranked positions. Selection bias occurs because interaction is not possible with items below an arbitrary cutoff position chosen by the front-end application at deployment time (i.e., showing only the top-k items). A less studied, third form of bias, trust bias, is equally important, as it makes interaction dependent on rank even after observation, by influencing the item's perceived relevance. To capture interaction disparity in the presence of all three biases, in this article, we introduce a flexible fairness metric. Using this metric, we develop a post-processing algorithm that optimizes fairness in ranking through greedy exploration and allows a tradeoff between fairness and utility. Our algorithm outperforms state-of-the-art fair ranking algorithms on several datasets.

CCS Concepts: • **Information systems** → **Learning to rank;**

Additional Key Words and Phrases: Recommender systems, ranking systems, fairness, bias

ACM Reference Format:
Zohreh Ovaisi, Parsa Saadatpanah, Shahin Sefati, Mesrob Ohannessian, and Elena Zheleva. 2024. Fairness of Interaction in Ranking under Position, Selection, and Trust Bias. *ACM Trans. Recomm. Syst.* 3, 2, Article 20 (November 2024), 28 pages. https://doi.org/10.1145/3652864

1 INTRODUCTION

Ranking algorithms used in recommender system platforms connect users on the demand side to ranked items on the supply side. With the expansion of online marketplaces, these algorithms have become central not only to users seeking to find their desirable items (e.g., rentals, movies, job applicants), but also to items seeking to get enough visibility and interaction by users. Thus,

Authors' addresses: Z. Ovaisi, University of Illinois Chicago, Chicago, Illinois, United States; e-mail: zovais2@uic.edu; P. Saadatpanah, Meta Inc, Washington DC, United States; e-mail: parsasp@fb.com; S. Sefati, Meta Inc, New York, United States; e-mail: shahinsefati@fb.com; M. Ohannessian and E. Zheleva, University of Illinois Chicago, Chicago, United States; e-mails: mesrob@uic.edu, ezheleva@uic.edu.

ACM 2770-6699/2024/11-ART20
https://doi.org/10.1145/3652864

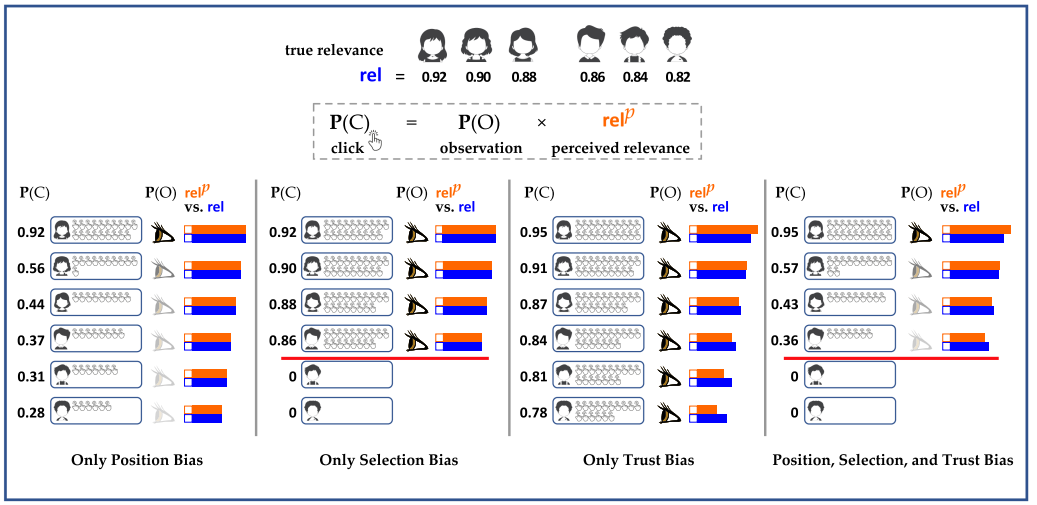


Fig. 1. Impact of biases on fairness of interactionreflecting employment chance for male and female applicants.

ranking algorithms have an impact on both user satisfaction and the amount of interaction each ranked item receives. Traditionally, the main focus of ranking algorithms has been to rank items in decreasing order of their relevance to users, with the aim of maximizing user satisfaction. However, naïvely ranking items based on their relevance scores may lead to unfairness to some items. This is because user interaction with items is heavily influenced by the item position in the ranking and users are less likely to interact with lower ranked items. As a result, more relevant items in upper positions collect a disproportionately larger number of user interactions than (sometimes slightly) less relevant items in lower positions. There are three main sources of user-item interaction bias that contribute to unfairness: position bias, selection bias, and trust bias. We first introduce these biases with an illustrative example which highlights their impact on the amount of interaction an item receives in a ranked list.

Consider a toy recommender system that connects employers and applicants for a job position, as illustrated in Figure 1. We assume that applicants (shown on top of the figure) belong to one of two groups, female or male, with the first three applicants being female and the other three male. Each applicant has a resume that has a relevance score (shown under each face) for the job position. All resumes compete for interaction (e.g., click) by employers, which occurs if the employers observe them, and find them relevant [3]. The platform ranks resumes in decreasing order of their relevance to the job, which results in female resumes placed in the first three positions and male resumes in the last three positions. **Position bias** refers to the fact that users are more likely to observe and, therefore, interact with higher-ranked items [4, 22, 44]. In Figure 1, the observation probability $\mathbf{P(O)}$ is shown with the decreasing eye opacity from top to bottom. The interaction probability $\mathbf{P(C)}$ of higher-ranked resumes is higher accordingly. **Selection bias** refers to the fact that users may not be able to see the full ranking and only the top-k items may be displayed by the front-end application (in the example, the cutoff $k = 4$) [32, 34]. Consequently, tail items beyond cutoff k will not be observed (e.g., most of the male resumes). Thus, tail items have interaction probability $\mathbf{P(C)}$ equal to zero. **Trust bias** refers to the fact that users may perceive top-ranked items as more relevant *even after observing all items* [3, 33, 41, 42]. This is because they overtrust the effectiveness of the system to rank relevant items higher. This is shown with the employers' perceived relevance rel^p (trust) of applicant resumes being amplified near the top

and attenuated near the bottom, relative to their true relevance rel. As a result, the interaction probability $\mathbf{P}(\mathbf{C})$ of higher-ranked resumes is higher. Finally, in practice, all three biases coexist, as in the fourth column of Figure 1. Note that trust bias is fundamentally different from position bias and selection bias. Under position bias and selection bias, the user has a realistic perception of the true relevance score ($\text{rel}^p = \text{rel}$). Thus, after observing an item, the user will choose whether to interact with it based on its true relevance regardless of its position. Under trust bias, however, the user perception of the item relevance score is position-dependent and may deviate from the actual relevance score ($\text{rel}^p \neq \text{rel}$). That is, after observing a lower ranked item, the user may mistakenly perceive it as much less relevant than it actually is and skip it. As a result, the top-ranked resumes get far more clicks than they deserve given their true relevances. We provide more detail on trust models in Section 3.

As a result of these existing biases, the recommender system causes the average number of interactions that female resumes receive to be much higher than those of male resumes although the relevances of male and female resumes are not significantly different, which is arguably unfair to male applicants. The discrepancies in interaction that items with certain properties (e.g., female applicants) receive could propagate further through rich-get-richer dynamics [10, 13]. In the case of two-sided marketplaces, this helps popular and relevant suppliers to receive the majority of available interaction in the long run, leaving the suppliers in the long tail struggling to attract users' interactions. These suppliers may then switch to other platforms, which in turn may limit user choices and consequently drive users to quit the platform. Besides such dire consequences, unfairness toward items (suppliers) may expose the online platform to legal and reputation risks, due to the resulting polarization and monopoly of popular suppliers.

Recent studies propose frameworks that take into account not only user satisfaction but also fairness toward items, considering item utility, such as exposure or interaction, as a *resource* provided by users to items. Most past studies focus on exposure as a resource [17, 35, 36, 47], thus providing fair exposure opportunity. We focus on interaction as a resource, also studied without trust bias in [36], thus providing fair interaction opportunity. Note that fairness of interaction more accurately reflects the effective impact of ranking, compared to fairness of exposure. This is because an item being interacted with/clicked has a higher chance of having a more effective engagement by the user (e.g., bought, invited to interview), in contrast to simply being shown to the user. This is especially prominent in e-commerce systems where the number of interactions of an item plays a key role in whether this item is further advertised to users in the future.

Once a resource is specified, then fairness can be considered under various scopes. *Scope* refers to how we account for the amount of received resource, before we compare whether two groups received the desired amounts. For example, we could pool across all users, or we could pool per user, or even per position. The concept of scope is illustrated with the example in Figure 2. Consider the same platform, with two employers (i.e., users) and multiple female and male applicant resumes (i.e., items), which aims at minimizing the difference in expected interaction that female and male applicant resumes would receive. $\mathbf{P}(\mathbf{C})$ for a resume decreases down the ranks due to position and trust bias effects. For each ranking, the aggregated interactions (in terms of expected number of clicks $\overline{\mathbf{C}}$) with male and female resumes per user is depicted on the right-hand side. For simplicity, we assume that the fairness criterion is loosely defined as "both groups receive the same expected amount of interaction." One possible scope that characterizes such fairness is by considering the expected interactions across *all users* (Figure 2(a)) [6, 35]. According to this scope, the expected interactions are fair, (female $\overline{C} = 3.06 + 0$, and male $\overline{C} = 0 + 3.06$ considering rankings shown to both users). However, this notion of fairness can be problematic when not all users are the same. For example, if the first employer is less active or plans to hire less people, female resumes would receive less interaction.

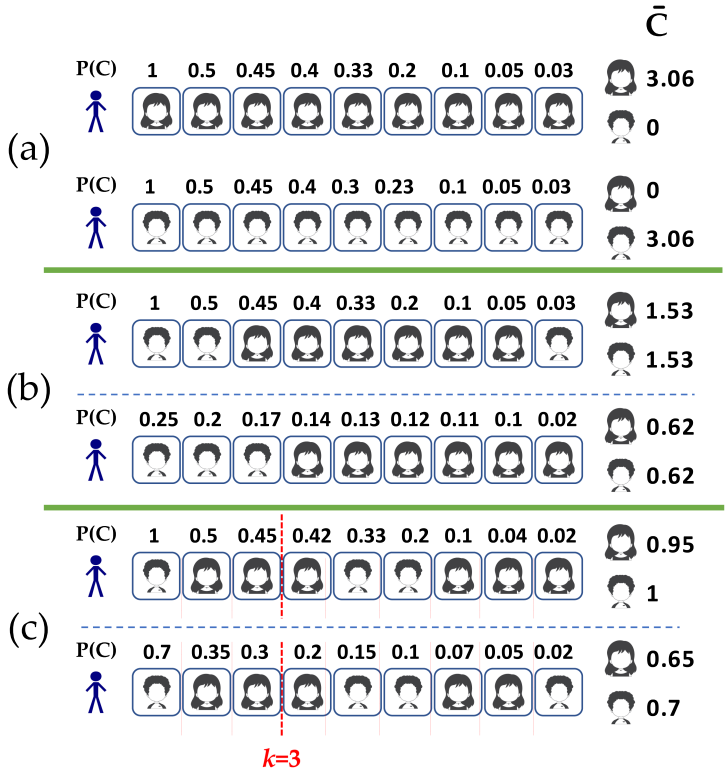


Fig. 2. Fairness criterion is met (a) over all users, (b) at each user, and (c) as much as possible at each k for each user.

One way to overcome this issue is by aiming at ensuring fairness at the user level and consider *per-user* fairness criteria [36, 37]. In the example, per-user item fairness is satisfied, as illustrated in Figure 2(b) where male $\overline{C}$ is identical to female $\overline{C}$ for each user. However, if the employer is presented with a truncated list of resumes, e.g., $k = 3$, then resumes below the cutoff get zero exposure, and the proposed fairness solution does not apply because of the large resulting discrepancy between male and female $\overline{C}$. One could argue that if the truncation cutoff is known in advance, then it could be taken into consideration while designing the ranking. However, front-end user interfaces where the recommendations appear can have varying truncation cutoffs, e.g., by device and application type, which requires the back-end recommendation algorithm to be versatile and handle multiple possible cutoffs k. To address this problem of varying cutoffs, some ranking algorithms aim at ranking items for *each user* such that the fairness criterion is satisfied as much as possible *per ranking position* [17, 47]. As shown in Figure 2(c), at $k = 3$ the discrepancies between male and female $\overline{C}$ are very small for the rankings shown to either user (0.95 vs. 1 and 0.65 vs. 0.7).

There are three fundamental questions that we aim at addressing in this work:

— **Q1**: Can a ranking method allow for item *interaction* fairness as opposed to item exposure fairness while maintaining high utility for users?

— **Q2**: Is such a fair ranking algorithm able to adjust the level of tradeoff between utility and fairness, thus enabling recommender platform decision makers to freely specify how much utility vs. fairness they desire based on their needs?

— **Q3**: How robust can this fair recommender system be to arbitrary selection bias cutoffs, while addressing interactions under user position and trust bias?

The perspective of the current article is that by refining the scope of interaction resource and simultaneously taking into account each user, each ranking index, and the effect of each position in terms of both observation and perceived relevance, we can achieve an effective exploration of the fairness-utility landscape. We place a more realistic model of interaction at the core of the notion of item fairness and offer the following main contributions:

— A per-user item fairness metric that captures **position**, **selection**, and **trust** bias (Section 3).
— A post-processing fair ranking algorithm, **Fairness Optimization for Ranking via Greedy Exploration** (**FORGE**), that improves item fairness while maintaining high user utility (Section 4.1).
— Theoretical (Sections 4.2) and empirical insight (Section 5.6) for FORGE's near-optimality at all **selection** cutoffs.
— Experimental evidence that incorporating all three types of biases achieves fairer ranking than the state-of-the-art algorithms that focus on a subset of them (Section 5).

We provide public access to our experimental implementation to enhance the reproducibility of the reported results.[1]

2 RELATED WORK

Controlling unfairness in ranking has been extensively studied in past literature [29, 36, 37, 47]. Some studies focus on in-processing algorithms, where fairness is incorporated in the learning algorithm [37, 48], while others focus on post-processing algorithms where they re-rank the final ranking produced by the ranking system to remove discrimination [17, 36]. We can categorize these by the scope of exposure resources.

First, there is work that considers wide exposure resource, namely *all users*. That is, fairness criteria are defined based on the exposure that is provided by all rankings for all users. Biswas and Barman [6] study the problem of fair recommender system and propose a greedy post-processing algorithm that ranks items in a round-robin fashion such that the fairness criterion is satisfied for both items and users when ranking top-k items to users. Later, Patro et al. [35] propose a slightly modified approach for the same goal, improving upon Biswas and Barman [6]. This study, while addressing the impact of selection bias, neglects to consider the influence of position bias. Wu et al. [45] tackles the same goal but proposes a fair method that accounts for position bias. Zhu et al. [49] proposes fair ranking using adversarial learning for top-k recommendation, handling a form of selection bias. The claimed performance of wide exposure methods is, however, arguable when there are inactive users, as items shown to them would get lower attention. In contrast, our method handles such scenarios by aiming at removing discrepancy at *each user* (in addition to accounting for all the biases).

Indeed, this is the theme of work that considers a narrower exposure resource where the fairness criterion is defined *per user*. That is, fairness is based on the exposure provided by the ranking for each user. These works account for position bias effects. Singh and Joachims [36] introduce a post-processing framework that maximizes the utility of ranking system subject to fairness constraints where fairness of interaction is defined in addition to fairness of exposure, but without accounting for trust bias. Basu et al. [5] extend [36] and present a framework that is fair to both users and items. Later, Zehlike and Castillo [48] propose an in-processing algorithm based on ListNet [8], a well-known ranking algorithm to minimize the inequality of opportunity in ranking,

[1] https://github.com/edgeslab/FORGE

but focuses only on the first position in ranking. Singh and Joachims [37] propose a novel in-processing policy-gradient approach that maximizes an objective function defined based on utility and fairness. Wang and Joachims [43] present a ranking algorithm that enforces user fairness, item fairness, and diversity. Singh et al. [38] propose a novel definition of fairness that incorporates uncertainty about items' merit, and present a fairness framework that maximizes utility subject to the fairness definition. These methods fail to incorporate the selection bias effect.

Lastly, there is literature that considers a highly limited exposure resource and thus accounts for selection bias. Zehlike et al. [47] present a post-processing fair top-k ranking algorithm that guarantees a required representation for the under-represented group. But their method is limited to binary attributes. Later, Geyik et al. [17] presents an efficient post-processing fair top-k ranking that meets fairness criteria at each ranking index and allows for multiple attributes. Celis et al. [9] proposes a theoretical analysis of fair ranking computation that similarly meets fairness requirements at each ranking index. More recently, Naghiaei et al. [30] propose a post-processing two-sided fairness method that accounts for both user and item fairness, but without considering position bias (they assume binary exposure, such that all shown items have the same exposure score). These lines of work do not account for trust bias, as they consider that items at different positions, once observed, get the attention they deserve. In contrast, we provide an algorithm that accounts for all three types of biases. It is also worth mentioning the work of Steck [39], where recommendations are calibrated to that the selection list reflects the various interests of users, as interpreted from their past interaction history. The goal of that work is not to directly impose fairness to items, though it may indirectly lead to it. Thus it is not expressly guaranteed to achieve fairness toward item groups. Note that [39] does not solve a ranking problem, but rather a selection problem. That said, the analysis of the approach relies on submodular optimization, and in that respect has some commonality with the theoretical insights that we provide in the present article.

Recent work also considers real-time recommender systems where users enter the system dynamically [15, 18, 40, 46]. These define and study fairness with the goal of satisfying the fairness of items over a time window, which is thematically related but outside the scope of the current article. Lastly, within the literature, there are studies exploring fairness in relation to users, whereas our work is distinctly concentrated on ensuring fairness towards items [24, 25].

3 PROBLEM SETUP

Consider the problem of fairly ranking items—such as movies, songs, search results, resumes, and so on.— to deliver to a user u based on a specific user query. Without loss of generality, in the rest of the article, we call these items *documents* and denote them by $\mathcal{D}$. A *ranking* of these documents is a permutation $\sigma : \{1, \ldots, |\mathcal{D}|\} \rightarrow \mathcal{D}$ that assigns to each *position* $\ell \in \{1, \ldots, |\mathcal{D}|\}$ a distinct document σ_ℓ. For each $d \in \mathcal{D}$ let $\text{rel}(d) \in \mathbb{R}_+$ be its *relevance* to user u. The front-end application may arbitrarily truncate the ranking consideration to only up to position k, we refer to this as *selection cutoff* k. Lastly, user u has a function rel^p that captures their *perceived relevance* of a document based on its position. This function has an unobserved *trust parameter* t, which thus influences how the ranked and selected documents are interacted with. Our goal is to achieve fairness of documents for any given user u, even if different users have different relevance vectors, selection cutoffs, and trust parameters. We formalize this in Section 3.4. In what follows, unless needed for clarity, we omit specifically referring to user u since they are fixed. Table 1 summarizes the notation we use in the article.

3.1 Biases

The quality of a ranking is whether or not each document is interacted with / clicked. We model this by describing the click probability based on each document's position and relevance, and how

Table 1. Glossary of Notations

Symbol	Description
u	user
D	set of documents to be ranked for user u
d	a document to be ranked
σ	a ranking function that assigns to each position a document
ℓ	position in a ranking
σ_ℓ	document positioned at position ℓ
k	cutoff (only top k documents are displayed to user)
rel^p	user-document perceived relevance score: $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t)$
rel	user-document true relevance score
r	possible value of relevance score
t	possible value of trust parameter
T	population of trusts
ϵ_ℓ^+	perceived relevance at rank ℓ of an observed relevant document
$\epsilon_\ell^-(t)$	perceived relevance at rank ℓ of an observed non-relevant document
$\overline{\mathbf{C}}$	expected number of clicks for a group of documents
G_i	i^{th} document group
α	interpolation level for trading off fairness and utility
P	relative number of interactions of each document group
Q	relative merit of each document group

the user finds documents relevant (trusts them). In recent work (see [3] and [42], for example), it is customary to assume that a click occurs when a document is both observed and trusted (perceived to be relevant) and that trust is independent of observation. We formalize this through the model

$$\mathbf{P}(\sigma_\ell \text{ clicked} \mid \text{trust } t) = \mathbf{P}(\sigma_\ell \text{ observed })\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t). \tag{1}$$

where $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}^p$ captures the perceived relevance of the user. Here we will clarify how this model portrays observation bias, selection bias and trust bias.

Position bias. Under this bias, the observation probability is document-position dependent, where $\mathbf{P}(\sigma_\ell \text{ observed }) = f(\ell)$, with the choice $f(\ell) = 1/\log(\ell + 1)$ (customarily in base-2) commonly used. The user's perceived relevance, however, is independent of document position, and is simply the same as the true relevance score. Thus $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}^p = \text{rel}(\sigma_\ell)$.

Selection bias. selection bias voids observation probability beyond cutoff k. A statistical model is not appropriate for this cutoff, because it is arbitrarily set at deployment time (e.g., depending on front-end application or device) and not reflected in user data. Following the approach of Ovaisi et al. [34] and Oosterhuis and de Rijke [32], we adopt an adversarial model, by characterizing performance at each k. Under this bias $\mathbf{P}(\sigma_\ell \text{ observed }) = 1$ for all documents that appeared in top k results, and $\mathbf{P}(\sigma_\ell \text{ observed }) = 0$ otherwise. The user's perceived relevance is simply the same as the true relevance score. Thus $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}^p = \text{rel}(\sigma_\ell)$.

Trust bias. When dealing only with trust bias, observation probability is position-independent, which leads $\mathbf{P}(\sigma_\ell \text{ observed }) = 1$. However, it effectively alters the perceived relevance of documents across positions. Thus perceived relevance now depends on both the true relevance and position, as opposed to position bias where it only depends on the true relevance. We can write this as $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}^p(\text{rel}(\sigma_\ell), \ell; t)$. We can understand the function $\text{rel}^p(r, \ell; t)$ as

taking in a specific relevance value r and modifying it per position ℓ, according the the trust parameter t of the user. We use a semicolon before t, to remind us that this describes a conditional quantity, that changes with the amount of trust. Our methodology applies for general rel^p, but for concreteness, we consider a realistic trust bias empirically determined by [3] and expressed in simplified form by [42], cf. Equation (11) and Equation (45):

$$\text{rel}^p(r, \ell; t) = r\epsilon_\ell^+ + (1-r)\epsilon_\ell^-(t), \quad \text{where } \epsilon_\ell^+ = 1 - \tfrac{\min\{\ell,25\}+1}{100} \text{ and } \epsilon_\ell^-(t) = \tfrac{t}{\min\{\ell,10\}}. \tag{2}$$

This simultaneously captures two forms of distortions to the perceived relevance: reduced trust in relevant documents at larger ℓ and increased undeserved trust in irrelevant documents at smaller ℓ.

Position, selection, and trust bias. Finally, we incorporate all biases in a single interaction/click model:

$$\mathbf{P}(\sigma_\ell \text{ clicked} \mid \text{trust } t) = \begin{cases} \mathbf{P}(\sigma_\ell \text{ observed})\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t), & \text{for } \ell \le k \\ 0, & \text{otherwise} \end{cases}$$

where $\mathbf{P}(\sigma_\ell \text{ observed}) = 1/\log(\ell+1)$ and $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}^p$ is derived from Equation (2).

3.2 Fairness

What is a *meaningful notion of fairness* in this model? Recall that we wish for rankings to give *fair interaction across groups of documents*. Denote these groups by $G_i \subset \mathcal{D}$, for $i = 1, \ldots, n$. These partition the set of documents $\mathcal{D}$. At a high level, fairness is measured by comparing the interaction provided by the ranking to a given notion of parity or equity across groups. For example, demographic parity aims at affecting — provide interaction with — each group equally [7, 19, 36]. Alternatively, one may aim at affecting groups proportionally to their utility to offer treatment parity [12, 27, 36].

In the present context, any flexible group fairness metric needs three components: a description of the actual *interaction* each document group receives from the ranking, a description of the desired *parity* (the ideal interaction), and a *distance* to compare them. Most importantly, the way interaction is measured needs to *(a)* account for the three highlighted types of bias (position, selection, and trust) and *(b)* be at the scope of users and not of the entire population, i.e., *per-user fairness*. To highlight the importance of *(b)*, one scenario where it is critical is when some users have wildly different activity levels, resulting in actual interaction with the document groups that is very different from the one assuming everyone actively interacts.

Interaction. Building on prior work [36], we propose using the *relative interaction* to describe differences across groups. This can be represented as a probability measure P over all groups, which is a vector:

$$P(\sigma, k) := \left(\tfrac{\text{interaction}(G_1, \sigma, k)}{\text{interaction}(k)}, \ldots, \tfrac{\text{interaction}(G_n, \sigma, k)}{\text{interaction}(k)} \right). \tag{3}$$

Here, the total expected interaction resource up to position k is defined as

$$\text{interaction}(k) := \sum_{\ell=1}^{k} \underbrace{\tfrac{1}{\log(\ell+1)}}_{\mathbf{P}(\sigma_\ell \text{ observed})} \underbrace{\text{rel}^p(\text{rel}(\sigma_\ell), \ell; t)}_{\mathbf{E}_T[\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } T)]}, \tag{4}$$

where rel^p is specifically given by Equation (2) and $t = \mathbf{E}[T]$ is the ensemble-averaged trust parameter. Intuitively, $\text{interaction}(k)$ represents the expected number of interactions/clicks received

by documents if the ranking is cut off at k, paralleling $\bar{\mathbf{C}}$ in the introductory example of Figure 2. Similarly, the restriction of this for each group G_i is

$$\text{interaction}(G_i, \sigma, k) := \sum_{\ell=1}^{k} \mathbb{1}\{\sigma_\ell \in G_i\} \, \frac{\text{rel}^P(\text{rel}(\sigma_\ell), \ell; t)}{\log(\ell+1)}.$$

This measures the total interaction received by group G_i, which represents the expected number of documents interacted with in that group. Thus, $\frac{\text{interaction}(G_i,\sigma,k)}{\text{interaction}(k)}$ indicates the fraction of total interactions the ranking allocates to G_i.

Note how: *(a)* the interaction resource depends on all three types of biases, **position bias** through the $\frac{1}{\log(\ell+1)}$ term, **selection bias** through the cutoff at k, and **trust bias** through the perceived relevances, and *(b)* the scope is per-user, owing to the use of individual relevances.

Parity. We represent parity via a *base measure* Q, which is a probability vector that we aim for P to resemble. As particular examples, we can parallel the notions used in [36]:

— *Demographic parity* can be represented by choosing Q_i proportionally to the size $|G_i|$ of group G_i,

$$Q = \tfrac{1}{|\mathcal{D}|} \left(|G_1|, \ldots, |G_d|\right).$$

— *Treatment parity* can be represented by choosing Q_i proportionally to the the merit $\sum_{d \in G_i} \text{rel}(d)$ of group G_i:

$$Q = \tfrac{1}{\sum_{d \in \mathcal{D}} \text{rel}(d)} \left(\textstyle\sum_{d \in G_1} \text{rel}(d), \ldots, \sum_{d \in G_n} \text{rel}(d)\right). \tag{5}$$

Although fairness in this article is illustrated mainly through these two notions, the choice of Q allows for flexible modeling. We particularly adhere to treatment parity, as it aligns fairness with the user's preferences.

Distance. To compare P in (3) to a choice of Q, we choose a distance $\text{dist}(P, Q)$ between probability measures, which we assume to be bounded, $\text{dist}(P, Q) \leq 1$ without loss of generality. We consider two distances:

— *Jensen-Shannon Distance* — This is the main distance used in our algorithm and experimental results. Let $M = \frac{1}{2}(P + Q)$ be the arithmetic midpoint of P and Q, then the Jensen-Shannon distance is defined as [14]:

$$\text{dist}_{\text{JSD}}(P, Q) := \text{JSD}(P\|Q) = \tfrac{1}{2}\text{KL}(P\|M) + \tfrac{1}{2}\text{KL}(Q\|M).$$

Recall that KL-Divergence between μ and ν is defined as $\text{KL}(\mu\|\nu) = \sum_i \mu_i \log_2 \frac{\mu_i}{\nu_i}$.

— *L^1 Distance* — This distance lends itself more easily to analysis, and is used for theoretical insight. It is equivalent to (twice) the total variation distance, and is defined as

$$\text{dist}_{L^1}(P, Q) := \|P - Q\|_1 = \textstyle\sum_{i=1}^{n} |P_i - Q_i|.$$

Both distances are straightforward to implement and the observed behavior in terms of fairness-utility tradeoffs is very similar. This can be attributed to the fact that JSD is (up to constant factors) upper bounded by L^1 and lower bounded by L^1-squared (cf. Pinsker's inequality). The L^1 distance is very beneficial to simplifying the theory. But we focus only on JSD for experimental evaluations, since it more closely relates to other reference fairness notions.

Fairness Metric. We now define the *fairness* of a ranking at cutoff k as follows:

$$\text{fairness}(\sigma, k) := 1 - \text{dist}\left(P(\sigma, k), Q\right). \tag{6}$$

This gives us a number in $[0, 1]$, as both distances are in the $[0, 1]$ range. Note that the closer the relative interaction of each group is to the desired parity—the smaller the distance between P (at k) and Q—the larger is this fairness metric.

We end by mentioning that this notion of fairness is grounded in and extends—notably by simultaneously accounting for all three types of biases per-user—prior suggestions in the literature (cf. the use of clickthrough rate for fairness as in Equation (9) of [36], the use of JSD in [28], and the use of selection bias with arbitrary cutoffs in [17].)

3.3 Utility

The goal of recommender systems is to maximize the utility of ranking for users by exposing them to relevant documents. In the present context, this continues to be a key objective, along with providing the desired fairness toward documents. We adhere to classical measures of utility. For each $d \in \mathcal{D}$ let $\text{rel}(d) \in \mathbb{R}_+$ be its *relevance*. For every cutoff k, the *utility* of ranking is given by the *discounted cumulative gain* of the ranking σ:

$$\text{DCG}(\sigma, k) := \sum_{\ell=1}^{k} \frac{2^{\text{rel}(\sigma_\ell)}-1}{\log(\ell+1)}.$$

It is customary to normalize DCG by its largest achievable value, to get the *normalized* discounted cumulative gain:

$$\text{nDCG}(\sigma, k) := \frac{\text{DCG}(\sigma, k)}{\text{I}(k)}, \qquad \text{I}(k) := \max_\sigma \text{DCG}(\sigma, k). \tag{7}$$

If document relevances are accurately predicted, and sorted from most to least relevant, nDCG reaches its highest value. That is, $\text{nDCG}(\sigma, k) = 1$, which translates to the highest user satisfaction from ranking.

3.4 Fairness-utility Tradeoff

Now that we have defined the fairness and utility objectives, we are ready to define the main goal of our article, obtaining a ranking that achieves an optimal tradeoff between fairness and utility. At each cutoff k, there is a multi-objective optimal Pareto frontier which can be characterized in one of two dual forms: (1) by fixing one of fairness/utility and optimizing the other, or (2) by interpolating between fairness and utility and optimizing the resulting objective. Most prior work attempting to achieve such a tradeoff in fair ranking adhere to (1). Here we propose form (2), as it lends itself to an elegant algorithmic solution.

Optimal tradeoff. Given an interpolation level $\alpha \in [0, 1]$, a ranking that achieves an optimal tradeoff at cutoff k is

$$\sigma^{\star,k} \in \arg\max_\sigma (1-\alpha)\, \text{nDCG}(\sigma, k) + \alpha\, \text{fairness}(\sigma, k), \tag{8}$$

where $\alpha = 0$ and 1 respectively correspond to ignoring fairness or utility entirely, and $0 < \alpha < 1$ achieves a certain desired tradeoff between the two objectives. The optimal algorithm is a brute-force algorithm that considers all possible permutations of $|\mathcal{D}|$ documents, and at each cutoff k, chooses a permutation that has the highest desired combination of $\text{nDCG}(\sigma, k)$ and $\text{fairness}(\sigma, k)$. However, the best permutation at cutoff k and cutoff $k' > k$ may order documents differently in the range $1, \ldots, k$. Thus, there may not be a single ranking that is optimal at *every* cutoff k. The full Pareto optimality frontier is indeed over $2|\mathcal{D}|$ objectives, i.e., the utility and fairness at every possible cutoff $k = 1, \ldots, |\mathcal{D}|$. Yet, the algorithm we present next produces a single ranking. While we can't expect it to achieve the tradeoff of $\sigma^{\star,k}$ at every k, the theoretical insight in Section 4.2 as well as empirical results in Section 5 suggest that one could at least hope to be within a constant factor of this optimal tradeoff with a single ranking.

4 FORGE: FAIRNESS OPTIMIZATION FOR RANKING VIA GREEDY EXPLORATION

We now propose our fair ranking algorithm that aims at approaching the optimal fairness-utility tradeoff. It is a post-processing fairness-aware re-ranking method, which we call FORGE.

ALGORITHM 1: FORGE

Input: $\mathcal{D}, u, (G_i), Q, \alpha$, t;
rel ← Run the base-ranker to get predicted relevance scores for user-document pairs;
for $i = 1, \ldots, n$ **do**
  interaction$(G_i) = 0$;
end
$k = 0$; $\sigma = \emptyset$;
for $k = 1, \ldots, |\mathcal{D}|$ **do**
  $d^\star = \arg\max_{d \in \mathcal{D} \setminus \sigma} (1-\alpha)\ \text{nDCG}([\sigma, d], k) + \alpha\ \text{fairness}([\sigma, d], k)$;
  $\sigma \leftarrow [\sigma, d^\star]$;
  for $i = 1, \ldots, n$ **do**
    if $d^\star \in G_i$ **then**
      interaction(G_i)+ $= \frac{\text{rel}^p(\text{rel}(d^\star), k; t)}{\log_2(k+1)}$;
      break;
    end
  end
end
$\sigma \leftarrow$ Final fair ranking;
Return σ

4.1 Algorithm

In addition to not producing a single ranking at all cutoffs k, the optimization in (8) is combinatorial in nature. A brute-force search is not feasible unless we have small values of k and $|\mathcal{D}|$. Instead, we give a greedy ranking algorithm that iterates over cutoffs k and works as follows: it chooses only one of the remaining documents which, when appended to the current ranking, maximizes the interpolated objective. It then updates the interaction of the corresponding group, allowing fairness and nDCG to be calculated at the next cutoff. The iterations can continue either to the end of documents $|\mathcal{D}|$, or may be stopped at a specified maximal ranking index. We describe this formally in pseudocode in Algorithm 1. Note that brackets [,] indicate concatenation. In this pseudocode, σ is represented as a list, which can be interpreted in this article's equations as a function mapping each position to the corresponding document in the list.

The arg max in Algorithm 1 is written in terms of the main tradeoff objective as in (8). But because the search only affects ranking index k, and cumulative terms 1 through $k-1$ in nDCG (7) and fairness (6) only depend on the (unchanging) ranking indices up to $k-1$, we can equivalently write it as

$$\arg\max_{d \in \mathcal{D} \setminus \sigma} \frac{1-\alpha}{\text{l}(k)} \frac{2^{\text{rel}(d)}-1}{\log_2(k+1)} + \alpha\ \text{fairness}([\sigma, d], k).$$

By maintaining a non-normalized P, this objective can be computed in $O(1)$ time for every remaining document, by updating only one coordinate of P and its normalization. The algorithm thus runs only in $O(|\mathcal{D}|^2)$ time.

4.2 Theoretical Insight

The position and trust bias aspects of the FORGE algorithm are straightforwardly optimized. The main question about its usefulness is whether a *single ranking* can compete with an optimal ranking at *every cutoff* k, and thus account for selection bias as well. The goal of this section is to support the insight that a greedy heuristic as in Algorithm 1 is reasonable for this purpose. We later also support this empirically in Section 5.6.

To focus on selection bias, we simplify the interaction in the fairness notion to take only exposure into account, i.e., we let $P_i(\sigma, k) \propto \sum_{\ell=1}^{k} \frac{\mathbb{1}\{\sigma_\ell \in G_i\}}{\log_2(\ell+1)}$ in Equation (3). What makes our task challenging is the combinatorial nature of the problem. To alleviate this, we turn the ranking problem to a *choice* problem by assuming oracle access to the optimal ranking, once the best k documents for the task are chosen. The question becomes: **how well can greedy choice compete with optimal choice at every cutoff** k? To simplify, we work with a proxy objective that lends itself to easier analysis, while retaining key properties of the original objective.

Proxy Objective. Assume documents $\mathcal{D}$ are partitioned into two groups of equal size, G and G^c. Let $\beta > 0$ and $\gamma > 0$ be two constants, used to abstract both normalizations and tradeoff parameter α. Define two set-valued functions F_+ and F_- on subsets $A \subseteq \mathcal{D}$:

$$F_\pm(A) := \max_{\sigma \in \mathrm{perm}(A)} \sum_{\ell=1}^{|A|} \left[\beta \frac{2^{\mathrm{rel}(\sigma_\ell)}-1}{\log_2(\ell+1)} \pm \gamma \left(\frac{\mathbb{1}\{\sigma_\ell \in G\}}{\log_2(\ell+1)} - \frac{\mathbb{1}\{\sigma_\ell \notin G\}}{\log_2(\ell+1)} \right) \right].$$

Using these, define also $F(A) := \min\{F_+(A), F_-(A)\}$. For the remainder of this section, *let the maximization of F be a proxy to the original tradeoff objective.* To justify this choice, let us establish a qualitative relationship between the two. Lower bounding min max by max min:

$$F(A) \geq \max_{\sigma \in \mathrm{perm}(A)} \beta \sum_{\ell=1}^{|A|} \frac{2^{\mathrm{rel}(\sigma_\ell)}-1}{\log_2(\ell+1)} - \gamma \left| \sum_{\ell=1}^{|A|} \left(\frac{\mathbb{1}\{\sigma_\ell \in G\}}{\log_2(\ell+1)} - \frac{\mathbb{1}\{\sigma_\ell \notin G\}}{\log_2(\ell+1)} \right) \right|.$$

The first term in the bound is the cumulative discounted gain. The second term is our fairness notion with the L^1 distance between P and $Q = \left(\frac{1}{2}, \frac{1}{2}\right)$, the special case of demographic parity with equal-sized groups. This is because:

$$\begin{aligned} \left|P_1 - \tfrac{1}{2}\right| + \left|P_2 - \tfrac{1}{2}\right| &= \left|P_1 - \tfrac{1}{2}(P_1 + P_2)\right| + \left|P_2 - \tfrac{1}{2}(P_1 + P_2)\right| \\ &= 2\left|\tfrac{1}{2}P_1 - \tfrac{1}{2}P_2\right| = |P_1 - P_2|. \end{aligned}$$

Note that nDCG and fairness with the chosen P have the same normalization $\mathsf{I}(k)$, which varies with k. However, this affects neither the greedy choice nor the constant-factor guarantee of Theorem 1, which is why we can consider β and γ to be constants throughout. As a result F is an upper bound on the objective function in this special case, and can be thought of as a proxy objective.

Proxy Algorithm. Assume oracle access to F: this places the onus of the ranking algorithm on choice rather than permutation, by making the latter 'free'. Namely, for any given cutoff k, the best ranking up to position k can be determined by first selecting the best subset A, by maximizing F, and then finding the best permutation σ on A. Towards this goal, consider the following procedure:

— Start with $A_0 = \varnothing$
— Iterate over $\ell = 1, \ldots, k_{\max}$:
$A_\ell = A_{\ell-1} \cup \left\{ \arg\max_{d \in \mathcal{D} \setminus A_{\ell-1}} F(A_\ell \cup \{d\}) \right\}$

This procedure maintains the spirit of Algorithm 1 but idealizes it, namely selecting greedily but allowing to re-rank after every new choice. Even in this idealized form, because the choice is performed greedily, we can formally pose the key question: **at every cutoff k, how close is $F(A_k)$ to the true maximal objective** $\max_{A:|A|=k} F(A)$**?**

Submodularity. Observe that if the objective were to maximize either F_+ or F_-, then greed would be sufficient (see proof of Lemma 1.) The same cannot be done with F. However, the construction of F from F_+ and F_- confers it key desirable properties, as given by Lemma 1:

Lemma 1. *F is* submodular*, i.e., all $A \subset B \subseteq \mathcal{D}$ and every $d \notin B$, $F(A \cup \{d\}) - F(A) \geq F(B \cup \{d\}) - F(B)$. F is also* monotone*, i.e., for all $d \in \mathcal{D}$, $F(A \cup \{d\}) - F(A) > 0$.*

Proof. (Sketch) The key to this property is to rely on the fact that F_+ and F_- can be optimized by sorting the elements of A, then using the telescoping properties of the sum to compare the impact of an addition to the set. In place of a sketch, we illustrate this by proving that F_+ is monotone and submodular. The proof for F is more elaborate but follows the same essential steps. (Of course the steps trivially extend to F_-, but it's not sufficient to stop there because in general the minimum of two submodular functions is not itself submodular). Consider two nested choices $A \subset B$. The optimizing permutation σ for each is obtained by sorting documents $d \in A$ or B according to decreasing

$$u(d) := \beta(2^{\text{rel}(d)} - 1) + \gamma\left(\mathbb{1}\{d \in G\} - \mathbb{1}\{d \notin G\}\right).$$

Consider next a new document $d \notin B$, and write $\Delta(A) := F_+(A \cup \{d\}) - F_+(A)$ and $\Delta(B) := F_+(B \cup \{d\}) - F_+(B)$. Since documents ranked higher than d do not affect Δ, it suffices to compare the contributions of all documents at the position of d and lower. Number such documents in A as $(d_0 \equiv d), d_1, \ldots, d_m$ in their u-sorted order. Say the optimal position of d in $A \cup \{d\}$ is j, then that of d_ℓ would be $j + \ell$. We then have that:

$$\Delta(A) = \frac{u(d)}{\log(j+1)} + \sum_{\ell=1}^{m} \left(\frac{u(d_\ell)}{\log(j+\ell+1)} - \frac{u(d_\ell)}{\log(j+\ell)} \right).$$

By using the fact that these documents are sorted, we can telescope this sum to obtain that $\Delta(A) \geq 0$, thus establishing monotonicity. Next, let $\varphi(\ell)$ indicate the optimal position of d_ℓ in B, $\varphi(0)$ being that of d by convention. Since $B \supset A$, we have $\varphi(\ell) \geq j + \ell$. By telescoping between successive $\varphi(\ell)$ positions, we get that:

$$\Delta(B) \leq \frac{u(d)}{\log(\varphi(0)+1)} + \sum_{\ell=1}^{m} \left(\frac{u(d_\ell)}{\log(\varphi(\ell)+1)} - \frac{u(d_\ell)}{\log(\varphi(\ell-1)+1)} \right).$$

Finally, by telescoping the difference and taking advantage of the decay of the logarithmic discounts, we obtain that $\Delta(A) - \Delta(B) \geq 0$, thus establishing submodularity. This approach extends to F. □

Intuitively, submodularity means that a new addition is more valuable to a smaller than a larger set and monotonicity means that every new addition increases the objective.

Near-Optimality of Greed. The submodularity property of this proxy objective means that the corresponding greedy algorithm is near-optimal.

Theorem 1. *At every $k = 1, \ldots, k_{\max}$, we have:* $F(A_k) \geq \left(1 - \frac{1}{e}\right) \max_{A:|A|=k} F(A)$.

This follows directly from classical submodularity results [31]. See also Theorem 1 in [23] for a more recent use in computer science. What is remarkable in this setting is that despite the fact that, generally, entirely different (non-nested) choices may be needed to achieve the optimal solution at every cutoff k, yet **growing (nested) choices are sufficient for a constant-factor approximation to the optimal solution**. As our experiments demonstrate, the same appears to also hold for rankings. In general, an entirely different ranking is needed to achieve the optimal tradeoff at each cutoff. But Algorithm 1, which produces a single ranking, performs well across all cutoffs. Thus, while the proxy objective and procedure presented here do not correspond to this algorithm exactly, they do validate its overall qualitative behavior.

5 EXPERIMENTS

We now provide experimental results to demonstrate how our methodology navigates the fairness and utility landscape. We introduce datasets, the base-rankers employed in post-processing fair rankings, summarize the baselines that we compare against, discuss the significance of our

results compared to the baselines, and assess optimality empirically by comparing to brute-force search.

5.1 Datasets

We perform evaluations using datasets tailored to various use cases, encompassing both personalized and non-personalized datasets. Personalized datasets focus on individual users' preferences for tailored recommendations, while non-personalized datasets emphasize general trends and patterns, making them suitable for ranking tasks. Specifically, we employ MovieLens1M, MovieLens100K, and Epinion as examples of personalized datasets, while the German Credit Dataset [11] and COMPAS datasets [21] serve as illustrations of non-personalized datasets. Further details about each dataset are presented below.

MovieLens1M Dataset. This publicly available dataset contains $1,000,000$ ratings (1–5 stars) from $6,000$ users on $4,000$ movies collected from the MovieLens website [2, 20]. This data also includes user demographic features such as age, gender, occupation, and so on, as well as movie features including movie title, release date, action, and so on. We binarize the interaction such that it is 1 if the user rated the movie and 0 otherwise. Also, to reduce the sparsity, we drop users and movies with less than 150 number of interactions.

MovieLens100k Dataset. This dataset is smaller than MovieLens1M dataset that contains $100,000$ ratings (1–5 stars) from 943 users on $1,682$ movies. Similar to the MovieLens1M dataset, we binarize the interaction such that it is 1 if the user rated the movie and 0 otherwise. Also, to reduce the sparsity, we drop users and movies with less than 80 number of interactions.

Epinion Dataset. This is a publicly available dataset that contains $664,824$ ratings (1–5 stars) from approximately $40,163$ users on $139,738$ items collected from the shopping website [1]. We binarize the interaction such that it is 1 if the user rated the item and 0 otherwise. Also, to reduce the sparsity, we drop users and items with less than 24 interactions.

German Credit Dataset. We use the *German Credit* dataset, which is publicly available [11]. This dataset contains $1,000$ applicants with corresponding non-PII features, including age, gender, job, marital status, and so on. Each applicant is assigned a binary credit rating based on their set of features by a German credit agency. This data reveals that, on average, women receive a lower credit score [47]. Motivated by this fact, we use gender as the protected attribute.

COMPAS Dataset. This dataset is also publicly available [21]. It contains $6,167$ criminal defendants with corresponding non-PII features, including age, gender, race, offense date, arrest date, and so on. Each defendant is assigned a binary two-year recidivism score based on their set of features. This data reveals that, on average, African-American defendants were far more likely than white defendants to be incorrectly judged to be at a higher risk of recidivism. Motivated by this fact, we use race as the protected attribute.

Table 2 illustrates the datasets used in this article

5.2 Base-rankers

We conduct evaluations using different base-rankers customized for diverse scenarios, covering both personalized and non-personalized domains. Specifically, we utilize *VAECF* [26], a base-ranker designed for personalization and recommender systems. Additionally, we employ *FeatNet*, which can serve both personalization and non-personalization purposes based on whether user features are included as input or not. Below, we provide descriptions for each of the base-rankers used in our evaluations:

Table 2. Datasets used in Experimental Analysis

Dataset	MovieLens1M	MovieLens100K	Epinion	German Credit	COMPAS
Protected attribute	popularity	popularity	popularity	gender	race
Public/Private	public	public	public	public	public
Personalized/ Non-personalized	Personalized	Personalized	Personalized	Non-Personalized	Non-Personalized
Size	1, 000, 000 ratings 4, 000 movies 6, 000 users	100, 000 ratings 1, 682 movies 943 users	664, 824 ratings 139, 738 items 40, 163 users	1, 000 applicants	6, 167 defendants

VAECF *[26].* This base-ranker is specifically designed for personalization and recommender systems. It extends variational autoencoders to collaborative filtering. It introduces a generative model with a multinomial likelihood and employs Bayesian inference for parameter estimation. This model enables us to forecast relevance scores for user-item pairs within the test dataset, which can subsequently be utilized by post-processing baselines to construct a fair ranking.

FeatNet. This base-ranker employs a neural network model to predict user-item relevance scores, utilizing their features as inputs, with a focus on optimizing for nDCG. We refer to it as 'FeatNet' for convenient reference throughout the article. To ensure a fair assessment alongside Fair-PG Rank [37], we adopt the same model architecture to the one used in Fair-PG Rank. However, note that the model used by Fair-PG Rank is designed to predict a hybrid score for user-item pairs, taking both fairness and *nDCG* into account. To obtain the FeatNet base ranker, we deactivate its fairness component ($\lambda = 0$) to solely optimize for *nDCG*. This gives us relevance scores for user-item pairs without any fairness consideration, which can then be utilized by the post-processing baselines to achieve a fair ranking. Note that when provided with both user and item features, this base-ranker can consider individual preferences, resulting in a personalized ranking algorithm. Without this input, it is unable to generate personalized rankings and can only produce pure rankings.

5.3 Baselines

Here, we present fair baselines in both in-processing and post-processing domains for fair ranking. In-processing fair ranking baselines can be divided into two categories: those originating from the recommender system domain and those from the pure ranking domain. The former pertains to personalized ranking, while the latter pertains to non-personalized ranking. Conversely, post-processing fair ranking baselines are generally applicable to both recommender systems and pure ranking domains. The reason for this adaptability is that post-processing fair rankings do not directly aim at predicting user-item relevance labels themselves as opposed to in-processing fair ranking baselines. Post-processing methods focus on the final stage of ranking, where they utilize the predicted user-item relevance values (obtained either through personalized or non-personalized approaches using the base-ranker) to re-rank the items and achieve a fairer final ranking.

Fairness-aware ranking. [17]: This is a post-processing method, and thus is applicable to both the recommender system and pure ranking domain. We use the exact same learned relevance score function on test queries. **Similarities:** This method addresses selection bias, by providing as much fairness as possible at each cutoff k. The fairness metric can be chosen to represent treatment parity. **Differences:** The fairness metric does not address position bias and trust bias, relying merely on counts (size of groups) to measure exposure differences across groups. This method does not explicitly explore tradeoffs, and produces only a single feasible ranking, with specific fairness and utility at each cutoff k.

PostProc. [36]: This is also a post-processing method which is applicable to both the recommender system and pure ranking domain. We use the exact same learned relevance score function on test queries. **Similarities:** The fairness metric takes into account position bias. It is chosen to represent treatment parity. We follow the lead of [37] that enables this method to strive to explore utility-fairness tradeoffs by adjusting the level ζ of linear combination with the fairness metric. **Differences:** Fairness is accounted for only at the last position, as it focuses on full ranking. That is, it relies on exposure resources over the whole spectrum of ranking, meaning that selection bias is not addressed. This approach also does not account for trust bias.

CPFair. [30]: This is also a post-processing approach that is relevant to both the domain of recommender systems and the domain of pure ranking. We use the exact same learned relevance score function on test queries. This method is a two-sided fairness re-ranking method rooted in optimization principles, which simultaneously incorporate fairness constraints from both the consumer and producer perspectives within a unified objective framework. To have a fair comparison with our approach, we deactivated its consumer fairness component to focus solely on producer fairness (fairness toward items). **Similarities:** The fairness metric takes into account selection bias where a truncated ranking might be displayed to the user. **Differences:** The underlying goal of this method is to rank S items from a pool of D items, where $|S| < |D|$. Consequently, it relies on the user interface to display $k = |S|$ items to the user. Due to this reliance, it may encounter challenges in upholding fairness requirements if fewer items are presented to the user on the front-end. Additionally, the approach does not consider position bias and trust bias.

Fair-PG Rank. [37]: This is an in-processing method: instead of learning relevances, this approach learns how to rank by using a parametrized distribution over rankings, conditionally on the features of the query/user. Learning occurs via empirical risk minimization on the training data, where the risk, as in this article, is also a linear combination of both the utility of the ranking and its fairness. While not inherently originating from the realm of recommender systems, it can still offer personalized prediction if equipped with both user and item features. **Similarities:** The fairness metric takes into account position bias. It can also be chosen to represent treatment parity. By adjusting the level λ of linear combination with this metric, this method also strives to explore utility-fairness tradeoffs. **Differences:** Fairness is accounted for only at the last position, as it focuses on full ranking. That is, it relies on exposure resources over the whole spectrum of ranking, meaning that selection bias is not addressed. This approach also does not account for trust bias.

DELTR. [48]: This is also an in-processing method, which reduces unfairness at training time. It extends ListNet [8], a well-known listwise learning-to-rank method, with a fairness objective that reduces discrimination by focusing on the top position in the ranking. Although not originating from the recommender system field, it possesses the ability to provide personalized prediction when user and item features are available. **Similarities:** The fairness metric takes into account position bias. By adjusting the level γ of linear combination with this metric, this method also strives to explore utility-fairness tradeoffs. **Differences:** It does not consider selection bias and trust bias effects.

Table 3 provides information about these baselines.

5.4 Conducting Ranking Tasks for each Dataset

In order to rank documents for each ranking task, we need to predict a score for all documents under a query/user, and rank them accordingly. In-processing baselines (Fair-PG Rank and DELTR)

[2] As opposed to FORGE and Fairness-aware ranking, CPFair does not aim at producing a fair ranking at each cutoff k, but rather to achieve fairness in ranking at a specific cutoff k

Table 3. Fair Ranking Algorithms Evaluated in Experimental Analysis

Fairness method	FORGE	Fairness-aware ranking	CPFair	PostProc	Fair-PG Rank	DELTR
In-processing/ Post-processing	post-processing	post-processing	post-processing	post-processing	in-processing	in-processing
Bias considered	position bias selection bias trust bias	selection bias	selection bias [2]	position bias	position bias	position bias
Interpolation level for fairness and utility	α	–	β	ζ	λ	γ

use training data to learn a fair ranking function. This function predicts a hybrid score that accounts for both fairness and utility, which can be applied to the test set.

Unlike in-processing methods, post-processing fair rankings (FORGE, Fairness-aware ranking, CPFair, and PostProc) require pre-existing relevance scores. Consequently, a base-ranker is essential to generate these scores. The relevance score can be learned through FeatNet or VAECF, depending on the specific dataset. We then use the predicted relevance score for ranking the test queries using FORGE as well as other post-processing baselines.

Below we describe how we conduct ranking tasks for each dataset:

MovieLens1M Dataset. We randomly split each dataset into train and test sets by $4:1$. We categorize items into two groups. Following the lead of [30], we select the top 5% of movies according to the number of interactions they received from the training set as the popular movies, i.e., short-head movies, and the rest as the unpopular items or long-tail movies for which we aim at imposing fairness. Finally, for the re-ranking experiment, we aim at providing a fair ranking for the top 50 movies with the highest number of ratings which potentially include movies from both short-head and long-tail groups. To achieve this, we train the base-ranker VAECF to forecast user-movie relevance scores specifically for these 50 movies within the test dataset. Subsequently, these predicted relevance scores are utilized for the FORGE algorithm and other post-processing baselines. We repeat this experiment 5 times and report the average, minimum, and maximum results.

Epinion Dataset. We take a similar approach as with the MovieLens1M dataset, and randomly split each dataset into train and test sets by $4:1$. We categorize items into two groups i.e., popular (short-head) movies, and unpopular (long-tail) movies. We train base ranker VAECF to predict user-movie relevance labels for the top 30 movies with the highest number of ratings in test data. Similarly, we later use these predicted relevance scores for FORGE and other post-processing baselines. We repeat this experiment 5 times and report the average, minimum, and maximum results.

MovieLens100K Dataset. For this dataset, we apply both VAECF and FeatNet as the base-rankers.

When predicting user-movie relevance score with VAECF, we take a similar approach as with the MovieLens1M dataset, and randomly split each dataset into train and test sets by $4:1$. Items are classified into two distinct groups: popular (short-head) movies, and unpopular (long-tail) movies. The base ranker VAECF is trained to anticipate user-movie relevance labels for the top 30 movies with the highest number of ratings within the test data. Analogously, these forecasted relevance scores are subsequently applied in FORGE and other post-processing baselines. We repeat this experiment 5 times and report the average, minimum, and maximum results.

When predicting user-movie relevance score with FeatNet, we randomly choose 150 users and 150 movies, and for each user and for each movie, we concatenate user and movie features, and assign label 1 if the user interacted with the document, and 0 otherwise. This helps FeatNet predict personalized relevance scores. Finally, for each user, we split the movies into training and testing sets by $4:1$, which leads to test data having 30 movies to be ranked for each user. Following the work of [16], we assign each movie a popularity score based on the number of interactions it

received in total and use movie popularity as the protected attribute. We select the bottom 20% of movies with the lowest number of interactions as unpopular and the rest as the popular movies. We repeat this experiment 5 times and report the average, minimum, and maximum results.

German Credit Dataset. The dataset lends itself naturally to a binary (credit-worthy or not) classification task. We follow the lead of [37] to create a synthetic ranking task from this data. We do this by learning to rank, on random queries generated as follows. Each query consists of 25 random applicants, sampled at a ratio of 4:1 of credit-worthy or not. Training data has 500 such queries, while test has 100. As previously stated, we consider gender as the protected attribute. We repeat this experiment 5 times and report average, minimum, and maximum results.

COMPAS Dataset. Similarly to the German Credit dataset, this data lends itself naturally to a binary (two-year recidivism or not) classification task. We create a synthetic ranking task from this data analogous to the German Credit dataset, where train and test data have 500 and 100 queries each with 25 randomly chosen defendants respectively. As previously stated, we consider race as the protected attribute. We repeat this experiment 5 times and report the average, minimum, and maximum results.

5.5 Evaluation

To highlight the ability of our approach to tradeoff utility with a fairness metric that takes into account position bias, trust bias, and selection bias, we plot average 1 − fairness (fairness violation) and average nDCG achieved by each method. We report these results when $\alpha \in [0, 1]$ for FORGE, when $\lambda \in [0, 100]$ for Fair-PG Rank, and $\gamma \in [0, 10^6]$ for DELTR, when $\beta \in [0, 0.01]$ for CPFair, and when $\zeta \in [0, 0.1]$ for PostProc. Fairness-aware ranking does not aim for a tradeoff and is plotted as a single point, i.e., a single pair (nDCG, 1 − fairness). Similarly to [42], we set the average trust parameter $t = 0.65$. Q is chosen to address treatment parity, as defined in Equation (5). We segment our evaluation into two sections: one where we employ FeatNet as the base-ranker, and the other where we use VAECF as our base-ranker.

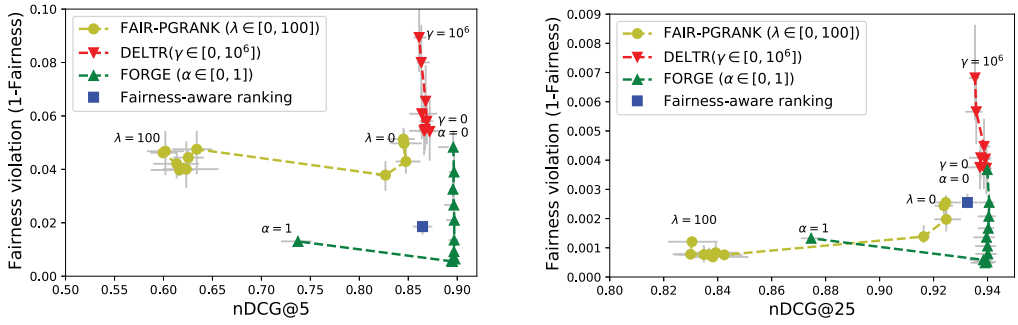
When FeatNet is used as the base ranker, Figures 3, 4, and 5 illustrate the results on the German Credit dataset, COMPAS dataset, and MovieLense100K dataset respectively at the intermediate cutoff ($k < |\mathcal{D}|$) and maximal cutoff ($k = |\mathcal{D}|$).

When VAECF is used as the base ranker, Figures 6, 7, and 8 illustrate the results on the MovieLens100K dataset, Epinion dataset, and MovieLense1M dataset respectively.

We now elaborate on how each type of bias may promote some amount of unfairness, and on why fair algorithms that do not take into account each of these biases have undesirable performance.

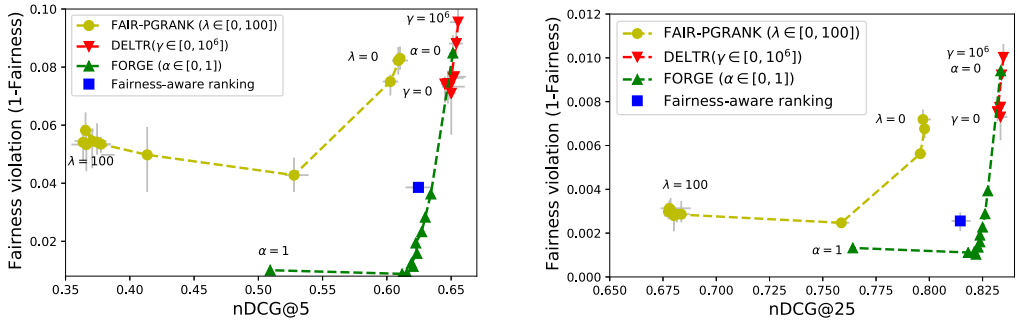
5.5.1 When FeatNet is the Base-ranker. Here, we evaluate fair baselines when we choose FeatNet as our base-ranker.

Impact of position bias and trust bias. Fairness-aware ranking accounts for selection bias but fails to account for position bias and trust bias. Thus, to highlight the effect of these two biases, we compare the performance of our approach with Fairness-aware ranking for all datasets. Figures 3, 4, and 5 show that FORGE provides an effective tradeoff between utility and fairness. In addition, a good performance at both intermediate and maximal cutoffs is achieved for both our algorithm and Fairness-aware ranking. Thus if the front-end application shows a truncated list of documents to a user, we can expect a strong fairness vs. utility guarantee. In contrast, since Fairness-aware ranking does not take into account position bias and trust bias, our approach dominates it (outperforms it in both nDCG and 1 − fairness) both at intermediate and maximal cutoff k within a certain range of α. More specifically, FORGE outperforms Fairness-aware ranking in both nDCG and 1 − fairness



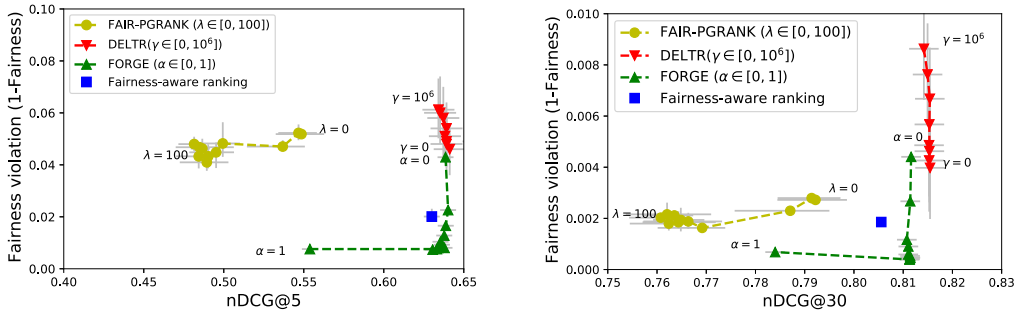
(a) $k = 5, |\mathcal{D}| = 25$ (b) $k = 25, |\mathcal{D}| = 25$

Fig. 3. German credit data with FeatNet as base-ranker— average nDCG vs. average 1 – fairness at intermediate and maximal cutoff k on test data. Variance is shown in gray.



(a) $k = 5, |\mathcal{D}| = 25$ (b) $k = 25, |\mathcal{D}| = 25$

Fig. 4. COMPAS data with FeatNet as base-ranker — average nDCG vs. average 1 – fairness at intermediate and maximal cutoff k on test data. Variance is shown in gray.



(a) $k = 5, |\mathcal{D}| = 30$ (b) $k = 30, |\mathcal{D}| = 30$

Fig. 5. MovieLens100K data with FeatNet as base-ranker— average nDCG vs. average 1 – fairness at intermediate and maximal cutoff k on test data. Variance is shown in gray.

for German Credit dataset when $\alpha \in [0.5, 0.9]$ at $k = 5$ and when $\alpha \in [0.1, 0.9]$ at $k = 25$, for the COMPAS dataset when $\alpha \in [0.1, 0.3]$ at $k = 5$ and when $\alpha \in [0.3, 0.9]$ at $k = 25$, and for the MovieLense dataset when $\alpha \in [0.2, 0.9]$ at $k = 5$ and when $\alpha \in [0.2, 0.9]$ at $k = 30$. Also, Fairness-aware ranking is not designed to achieve a tradeoff between utility and fairness, and

Table 4. Fairest Results Targeting a Specified Desirable Fairness (Fairness Violation < 0.01), at Intermediate Cutoff k when FeatNet is the Base-ranker

Method \ Dataset	German Credit		COMPAS		MovieLens100K	
	nDCG	fairness violation	nDCG	fairness violation	nDCG	fairness violation
FORGE	0.8950	0.00553	0.611	0.0087	0.6331	0.00758
Fair-PG RANK	0.826	0.0378	0.527	0.0427	0.48939	0.04097
DELTR	0.8718	0.0541	0.645	0.0741	0.6411	0.04602
Fairnes-Aware Ranking	0.8645	0.0185	0.624	0.038	0.630	0.0201

Only FORGE achieves this (in blue), while its nDCG (in red) remains comparable to all the baselines.

cannot, as FORGE does, significantly improve fairness for only a small reduction in utility. Thus, if highly-ranked applicants are over-trusted, this baseline would not only be less fair across groups but would also not achieve the highest utility possible, for the amount of fairness it provides.

Impact of selection and trust bias. Fairness-PG Rank and DELTR account for position bias but do not account for selection bias and trust bias. Thus to highlight the effect of selection bias and trust bias, we compare the performance of our approach with these two algorithms for all datasets. We chose a fairness metric for these approaches to closely match the choice of treatment parity in FORGE.

For all datasets (Figures 3, 4, and 5) and for both intermediate and maximal cutoff k, FORGE achieves a good tradeoff between utility and fairness, and outperforms Fair-PG Rank in both nDCG and $1 -$ fairness. In other words, for a minimal change in nDCG, FORGE improves the fairness noticeably. Also, although Fair-PG Rank falls behind FORGE in both nDCG and $1 -$ fairness for both intermediate and maximal cutoff k, the worse performance of Fair-PG Rank over FORGE is more pronounced at the intermediate cutoff. This is because, at the intermediate cutoff ($k = 5$), we introduce selection bias, while Fair-PG Rank ignores selection bias, and assumes all ranking indices are shown by the user's interface. Thus it relies on the whole ranking spectrum to achieve fairness across all documents. In contrast, we aim at accounting for the fact that documents below a certain cutoff may (arbitrarily) not be shown to the user. We believe that the reason Fair-PG Rank still lags behind FORGE at maximal cutoff k, where no selection bias exists, is that it does not account for the effect of trust bias.

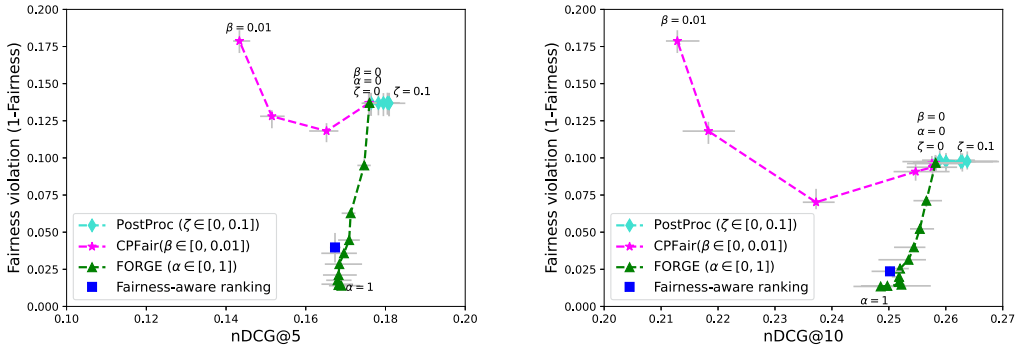
To compare FORGE with DELTR, DELTR fails to achieve a reasonable tradeoff between utility and fairness. While it illustrates a more reasonable tradeoff for the COMPAS dataset, it shows an unreasonable counter-trade-off behavior for other datasets. This counter-trade-off behavior was also observed in Figure 3(b) in [37]. DELTR also lags behind FORGE significantly in the fairness it achieves. In other words, it fails to improve fairness noticeably. Also, although DELTR falls behind FORGE for both intermediate and maximal cutoff k, the performance gap reduces at maximal cutoff, illustrating once again that it is not sufficient to optimize over the full range to guarantee good performance when the ranking is cut off due to selection bias.

Next, to evaluate how fair each baseline is, we report the minimum fairness violation ($1 -$ fairness) and its corresponding nDCG at both intermediate and maximal cutoff k for each algorithm on each dataset. This aims at capturing the practical use of this algorithm when it is required to guarantee fairness, up to a threshold. Table 4 and Table 5 show the results for this evaluation at intermediate and maximal cutoff k respectively. In blue, we highlight the lowest fairness violation achieved by these methods. In red, we highlight the nDCG values achieved when fairness violation is below a given threshold, which we set to fairness violation < 0.01 for intermediate cutoff k and fairness violation < 0.001 for maximal cutoff k. FORGE outperforms all baseline methods in terms of fairest results while it maintains its high nDCG for the given fairness violation threshold.

Table 5. Fairest Results Targeting a Specified Desirable Fairness (Fairness Violation < 0.001), at Maximal Cutoff k when FeatNet is the Base-ranker

Method \ Dataset	German Credit		COMPAS		MovieLens100K	
	nDCG	fairness violation	nDCG	fairness violation	nDCG	fairness violation
FORGE	0.9396	0.00048	0.821	0.0010	0.8113	0.00039
Fair-PG RANK	0.83818	0.00068	0.758	0.0024	0.76917	0.00162
DELTR	0.93959	0.00371	0.8334	0.0073	0.81545	0.00397
Fairnes-Aware Ranking	0.9325	0.0025	0.814	0.0025	0.8055	0.0018

FORGE achieves this (in blue) every time, and its nDCG (in red) remains comparable to all the baselines. The only baseline and dataset pair where the fairness target is achieved is Fair-PG RANK on German Credit, but nDCG is then markedly lower.



(a) $k = 5, |\mathcal{D}| = 30$ (b) $k = 10, |\mathcal{D}| = 30$

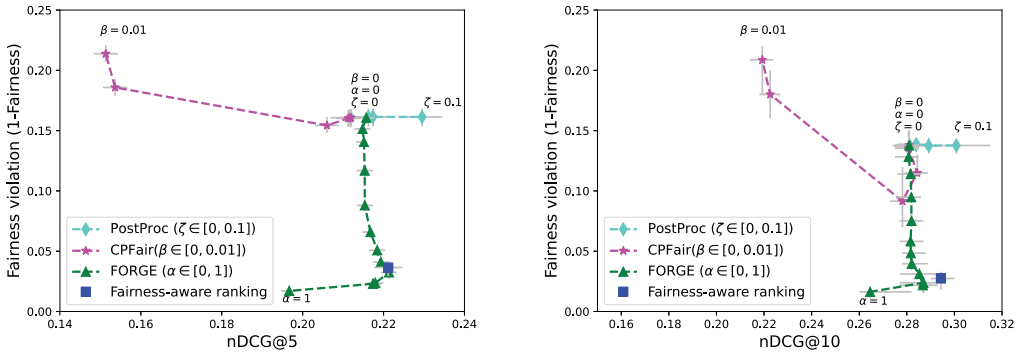
Fig. 6. MovieLens100K data with VAECF as base-ranker— average nDCG vs. average 1 − fairness at $k = 5$, and $k = 10$ on test data. Variance is shown in gray.

5.5.2 When VAECF is the Base-ranker. Here, we evaluate fair baselines when we choose VAECF as our base-ranker. The primary objective here is to exclusively compare FORGE as a post-processing baseline with other post-processing baselines, ensuring a direct and fair comparison specific to this category of fair methods. Additionally, since post-processing fair ranking baselines are generally relevant to the personalization and recommender systems domain, this section exclusively addresses the realm of personalization.

Impact of position bias and trust bias. Fairness-aware ranking and CPFair address selection bias, yet overlook position bias and trust bias. To emphasize the impact of these two biases, we conduct a performance comparison between FORGE and these two baselines across the MovieLens1M, MovieLens100K, and Epinion datasets, each requiring sorting of 50, 30, and 30 movies per user, respectively.

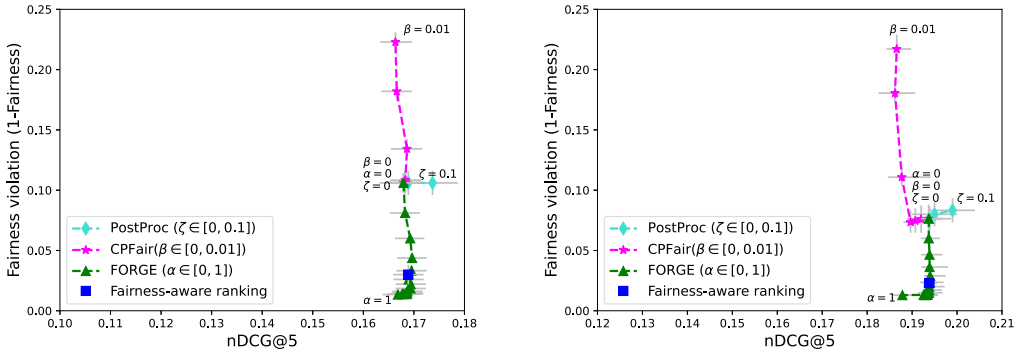
Note that, CPFair is designed to rank S items out of D, with unfairness quantified as the disparity in the number of items from each group among the top S items. Similar to [30], we choose $S = 10$. While CPFair considers selection bias, its assumption of predefined truncation criteria can lead to limitations: If truncation on the user end occurs at a lower cutoff ($k \leq S$), it may not effectively mitigate selection bias. Thus, we present results at intermediate cutoffs of $k = 5$ and $k = 10$. We omit evaluation at maximal cutoffs ($k = 30$ and $k = 50$) as CPFair's objective does not extend to sorting all D items.

Figures 6, 7, and 8 show that FORGE provides an effective tradeoff between utility and fairness. In addition, a good performance at both $k = 5$ and $k = 10$ cutoffs is achieved for both our



(a) $k = 5, |\mathcal{D}| = 30$ (b) $k = 10, |\mathcal{D}| = 30$

Fig. 7. Epinion data with VAECF as base-ranker— average nDCG vs. average 1 – fairness at $k = 5$, and $k = 10$ on test data. Variance is shown in gray.



(a) $k = 5, |\mathcal{D}| = 50$ (b) $k = 10, |\mathcal{D}| = 50$

Fig. 8. MovieLens1M data with VAECF as base-ranker— average nDCG vs. average 1 – fairness at $k = 5$, and $k = 10$ on test data. Variance is shown in gray.

algorithm and Fairness-aware ranking. Thus if the front-end application shows a truncated list of documents to a user, we can expect a strong fairness vs. utility guarantee. In contrast, since Fairness-aware ranking does not take into account position bias and trust bias, our approach can dominate it in nDCG and 1 – fairness within a certain range of α in most cases. Specifically, across the MovieLens100K dataset, FORGE exhibits superior performance to Fairness-aware ranking in both nDCG and 1 – fairness metrics when α falls within the range of $[0.4, 1]$ at $k = 5$, and within $[0.6, 0.8]$ at $k = 10$. Within the MovieLens1M dataset, FORGE mainly demonstrates a comparable nDCG score with Fairness-aware ranking, while it surpasses it in terms of 1 – fairness when α lies within $[0.6, 0.8]$ at both $k = 5$ and $k = 10$. In the context of the Epinion dataset, when $k = 5$, FORGE and Fairness-aware ranking yield similar *nDCG* and 1 – fairness values at $\alpha = 0.8$, with FORGE outperforming Fairness-aware ranking in 1 – fairness at the expense of a minor reduction in *nDCG*. At $k = 10$, FORGE exhibits a moderate superiority over Fairness-aware ranking solely in the 1 – fairness metric, albeit with a moderate reduction in the *nDCG* metric. In general, Fairness-aware ranking showcases a satisfactory performance, yet it is surpassed by FORGE in most scenarios. Nevertheless, it is not designed to achieve a tradeoff between utility and fairness, and cannot, as FORGE does, allow us to adjust the level of desired *nDCG* and fairness.

Table 6. Fairest Results Targeting a Specified Desirable Fairness (Fairness Violation < 0.02), at Intermediate Cutoff $k = 5$ for Post-processing Fair Methods when VAECF is the Base-ranker

Method \ Dataset	MovieLens100K		Epinion		MovieLens1M	
	nDCG	fairness violation	nDCG	fairness violation	nDCG	fairness violation
FORGE	0.1680	0.0139	0.2170	0.0199	0.1668	0.0134
CPFair	0.1652	0.1181	0.2060	0.1543	0.1679	0.1060
PostProc	0.1809	0.1367	0.2118	0.1606	0.1736	0.1060
Fairnes-Aware Ranking	0.1673	0.0397	0.2211	0.0364	0.1689	0.0299

Only FORGE achieves this (in blue), while its nDCG (in red) remains comparable to all the baselines.

Table 7. Fairest Results Targeting a Specified Desirable Fairness (Fairness Violation < 0.02), at Intermediate Cutoff $k = 10$ for Post-processing Fair Methods when VAECF is the Base-ranker

Method \ Dataset	MovieLens100K		Epinion		MovieLens1M	
	nDCG	fairness violation	nDCG	fairness violation	nDCG	fairness violation
FORGE	0.2485	0.0134	0.2645	0.0162	0.1878	0.0129
CPFair	0.2372	0.0701	0.2781	0.0915	0.1896	0.0734
PostProc	0.2581	0.0964	0.2892	0.1375	0.1939	0.0763
Fairnes-Aware Ranking	0.2501	0.0236	0.2943	0.0243	0.1938	0.0228

Only FORGE achieves this (in blue), while its nDCG (in red) remains comparable to all the baselines.

When it comes to CPFair, FORGE outperforms CPFair in both nDCG and 1 − fairness. In other words, for a minimal change in nDCG, FORGE improves the fairness noticeably, a result that CPFair fails to replicate. This is because CPFair does not account for position and trust bias.

Impact of selection and trust bias. PostProc accounts for position bias but does not account for selection bias and trust bias. Thus to highlight the effect of selection bias and trust bias, we compare the performance FORGE with it for MovieLens100K, Epinion, and MovieLens1M datasets. PostProc concentrates on optimizing a fairness metric that, while excluding trust bias, closely corresponds to FORGE's emphasis on treatment parity.

Figures 6, 7, and 8 illustrate that unlike FORGE which reduces disparity with increasing α, PostProc does not produce a better ranking in terms of 1 − fairness when ζ is increased (in subsection 5.5.4 we will show that PostProc produces a ranking that becomes even worse in terms of its fairness metric as ζ is increased). While it demonstrates a slightly higher nDCG, its significant deficiency in fairness emphasizes FORGE's superiority over it. This poor performance can be attributed to the neglect of selection and trust bias.

Next, to evaluate how fair each baseline is, we report the minimum fairness violation (1 − fairness) alongside its corresponding nDCG at $k = 5$ and $k = 10$ for post-processing fair algorithms across the MovieLens100K, Epinion, and MovieLens1M datasets. This evaluation aims at reflecting the practical utility of the algorithm, demonstrating its ability to ensure fairness within a specified threshold. Table 6 and Table 7 show the results for this evaluation at cutoff $k = 5$ and $k = 10$ respectively. In blue, we highlight the lowest fairness violation achieved by these methods. In red, we highlight the nDCG values achieved when the fairness violation is below a given threshold, which we set to fairness violation < 0.02. Likewise, FORGE outperforms all baseline methods in terms of fairest results while it maintains its high nDCG for the given fairness violation threshold.

5.5.3 Decomposing Bias Effects in FORGE (Isolating and Combining Variations). Thus far, our evaluations of FORGE's performance have demonstrated the impact of considering all three biases simultaneously. In this section, we provide an illustration of how FORGE would have performed in terms of fairness if it had not accounted for the effect of each bias when re-ranking. This analysis

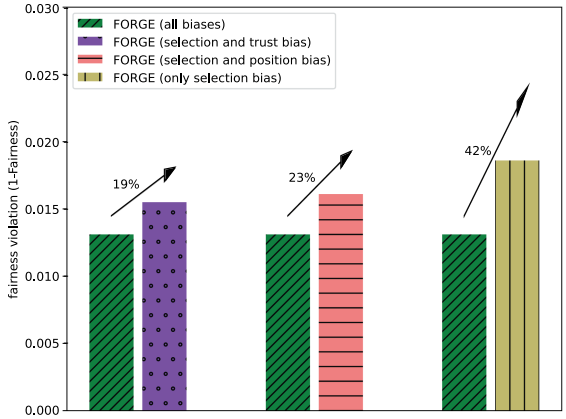


Fig. 9. MovieLens1M data with VAECF as base-ranker— performance of FORGE variations in 1 − fairness with comparable *nDCG* at $k = 10$.

aims at revealing the effects of ignoring each bias when providing a fair ranking, and how it would have influenced FORGE's overall performance.

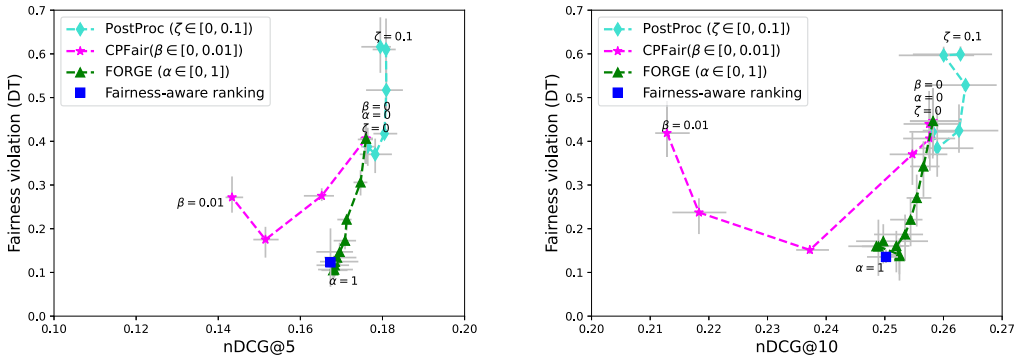
As FORGE is a greedy-based algorithm, it inherently incorporates considerations for selection bias issues. Consequently, we cannot prevent it from doing so. Hence, we make modifications to the FORGE algorithm to account for different scenarios: (i) accounting for only trust bias and selection bias while ignoring position bias, which translates to $\mathbf{P}(\sigma_\ell \text{ observed}) = 1$ for top-k items, (ii) accounting for only position bias and selection bias while ignoring trust bias, which translates to $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}(\sigma_\ell)$ for top-k items, and (iii) accounting for only the selection bias and ignoring position and trust bias, which translates to $\mathbf{P}(\sigma_\ell \text{ observed}) = 1$ and $\mathbf{P}(\sigma_\ell \text{ trusted} \mid \text{trust } t) = \text{rel}(\sigma_\ell)$. We then compare each of these cases individually against the scenario where FORGE accounts for all three biases simultaneously, including position, trust, and selection bias. In order to demonstrate the impact of disregarding each bias on fairness, we identify a specific value for α that yields comparable (or nearly comparable) *nDCG* across all the cases. Subsequently, we report the corresponding fairness metrics obtained for the same *nDCG* in each instance. Figure 9 visually presents the described FORGE variations using the MovieLens1M dataset at $k = 10$. As shown, disregarding the position bias effect raises unfairness by 19%, while ignoring the trust bias effect results in a 23% increase in unfairness. Furthermore, the omission of both position bias and trust bias effects leads to a notable 42% rise in unfairness.

5.5.4 Exploring an Alternative Fairness Metric in Evaluation. So far, we evaluated all fair baselines with the fairness metric that we optimize the FORGE with (Equation (6)), which other baselines do not optimize for. While it is wise to optimize the ranking for the same metric as the one used in the evaluation, we illustrate the performance of our algorithm FORGE under other circumstances. Specifically, we evaluate FORGE and other fairness baselines on a fairness metric used by Fair-PGRANK [37] and PostProc [36]. This is a **disparate treatment (DT)** constraint that enforces the exposure of the two groups to be proportional to their average utility:

$$DT := \left| \frac{\overline{Exposure(G_0, \sigma, k)}}{Utility(G_0|u)} - \frac{\overline{Exposure(G_1, \sigma, k)}}{Utility(G_1|u)} \right|. \tag{9}$$

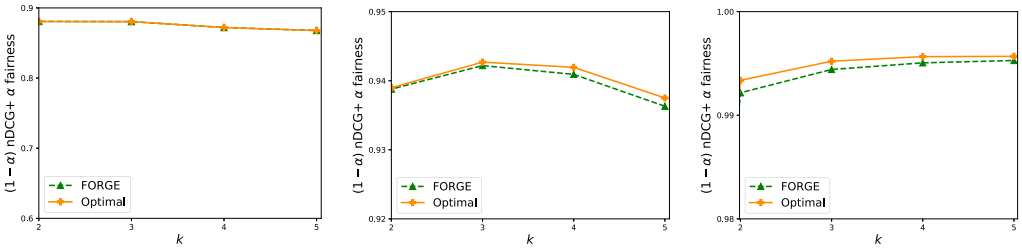
where $Exposure(G_i, \sigma, k) := \sum_{\ell=1}^{k} \mathbb{1}\{\sigma_\ell \in G_i\} \frac{1}{\log(\ell+1)}$, and $Utility(G_i|u) = \frac{1}{|G_i|} \sum_{d \in G_i} rel(d)$

Figure 10 illustrates the performance of fair methods on MovieLens100K at both $k = 5$ and $k = 10$. Surprisingly, PostProc becomes even less fair as ζ is increased though it optimizes for DT,



(a) $k = 5, |\mathcal{D}| = 30$ (b) $k = 10, |\mathcal{D}| = 30$

Fig. 10. MovieLens100K data with VAECF as base-ranker— average nDCG vs. average disparate treatment (DT) at $k = 5$, and $k = 10$ on test data. Variance is shown in gray.



(a) $\alpha = 0, |D| = 10$ (b) $\alpha = 0.5, |D| = 10$ (c) $\alpha = 1, |D| = 10$

Fig. 11. FORGE vs. optimal brute-force algorithm.

the metric that it is evaluated with. This unexpected behavior corroborates what has already been reported in the literature, e.g., in Figure 3 of [37]. Similarly, CPFair illustrates a poor performance when evaluated with this fairness metric. Conversely, both FORGE and Fairness-aware ranking excel in performance, despite being evaluated with a metric different from their respective optimization objectives. This highlights the remarkable resilience of FORGE's good performance even when evaluated using a distinct fairness metric.

5.6 Comparison with Optimal Brute-Force Search

Figure 11 illustrates the performance of FORGE versus the optimal brute-force algorithm for the German Credit dataset at cutoffs $k = 2, \ldots, 5$ and for $\alpha = 0, 0.5, 1$, when $|\mathcal{D}| = 10$. Note that a brute-force search over all permutations is only tractable with such modest choices of k and $|\mathcal{D}|$. As expected, the greedy algorithm is optimal for $\alpha = 0$, since it results in sorting documents in order of decreasing relevance. More interestingly, when $\alpha > 0$, the algorithm still maintains an acceptable margin from the optimal ranking in terms of the combined objective $(1 - \alpha) \cdot \text{nDCG} + \alpha \cdot \text{fairness}$. This is true despite the optimal ranking at different k's being different as explained in Section 4.1. These results experimentally illustrate the insights of Section 4.2.

5.7 Main Takeaways from Experimental Results

— Comparing FORGE with in-processing baselines, FORGE offers a more favorable tradeoff in terms of both nDCG and fairness, compared to Fair-PG Rank and DELTR. Furthermore,

FORGE exhibits exceptional fairness performance in comparison to PostProc. This difference stems from the fact that Fair-PG Rank, DELTR, and PostProc do not address selection bias or trust bias. The performance gap between FORGE and Fair-PG Rank, as well as DELTR, narrows at maximal cutoffs due to the absence of selection bias effects. The remaining performance gap at maximal cutoffs can be attributed to the unaddressed trust bias effect in DELTR and Fair-PG Rank.

— Comparing FORGE with in-processing baselines, FORGE outperforms Fairness-aware ranking in nDCG and fairness across most scenarios, and it outperforms CPFair in all cases due to Fairness-aware ranking and CPFair not addressing position bias or trust bias.

— FORGE provides the overall highest amount of fairness compared to all baselines while retaining high nDCG. We can quantify this by being stringent in our fairness requirement. We then see that FORGE is often the only method that meets these requirements. Furthermore, its nDCG is always comparable to the fairest of the baselines, which themselves do not meet the fairness requirement. (Tables 4 and 5).

— FORGE performance is experimentally verified to be near optimal, by comparing to the brute-force exhaustive search algorithm.

6 CONCLUSION

In this article, we considered the problem of fair ranking by introducing a fairness metric that incorporates position and trust bias, and proposed a greedy ranking algorithm (FORGE) that aims for a fairness-utility tradeoff when the ranking is cut off arbitrarily, thus addressing selection bias. We gave theoretical insight on why a greedy algorithm could provide a single ranking that competes with the optimal ranking despite the latter being possibly different at different cutoffs. We demonstrated our approach on six datasets, highlighting the effectiveness of FORGE in contrast to state-of-the-art baselines that either fail to account for some of these biases. This work paves the way for exploring the interplay between trust behavior and fairness, as well as developing algorithms with robust fairness-utility tradeoffs.

On the applied side, measurement and validation of trust behavior by users is crucial to achieve the true potential of this work. This can parallel and extend some of the validation methodologies for position bias. On the theoretical front, direct guarantees for greedy ranking would be most welcome, by shedding light on how one can efficiently approach the high-dimensional Pareto frontier across all cutoffs.

Some of the basic aspects of this methodology may be readily extended. For example, to adapt to non-disjoint groups, a few different approaches may be taken. A principled but inefficient approach is to work with the partition that refines the groups. These would be disjoint, and though the dimensionality increases, the exposure distributions remain well-defined. A less principled but more efficient possibility is to normalize the exposures over the overlapping groups, and compare them the same way. The disadvantage is that groups with areas of large overlap will be disproportionately represented. Lastly, if groups can be embedded in a meaningful way (e.g., as in topic models), then exposure distribution over the embedding space can be used instead.

While the focus is on enhancing fairness in ranked items, altering the fairness metric (e.g., favoring privileged groups with Q) could lead to unintended adverse effects. This fundamentally alters the message of the article and motivates the need to audit ranking systems for such malicious manipulation. In the interim, imposing benevolent fairness in post-processing could combat existing biases, malicious or otherwise.

REFERENCES

[1] Epinion dataset. Retrieved 3 February 2023 from https://www.shopping.com/. (n.d.).

[2] MovieLens dataset. Retrieved 3 February 2023 from https://movielens.org/. (n.d.).
[3] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing trust bias for unbiased learning-to-rank. In *Proceedings of the Web Conference*.
[4] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. *In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018).
[5] Kinjal Basu, Cyrus DiCiccio, Heloise Logan, and Noureddine El Karoui. 2020. A framework for fairness in two-sided marketplaces. arXiv:2006.12756. Retrieved from https://arxiv.org/abs/2006.12756
[6] Arpita Biswas and Siddharth Barman. 2018. Fair division under cardinality constraints. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.
[7] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. 2009. Building classifiers with independency constraints. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*.
[8] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*.
[9] L. Elisa Celis, Damian Straszak, and Nisheeth K. Vishnoi. 2018. Ranking with fairness constraints. *International Colloquium on Automata, Languages and Programming* (2018).
[10] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*.
[11] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI machine learning repository. 12 (2017). Retrieved 3 February 2023 from http://archive.ics.uci.edu/ml
[12] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*.
[13] Francesco Fabbri, Maria Luisa Croci, Francesco Bonchi, and Carlos Castillo. 2022. Exposure inequality in people recommender systems: the long-term effects. In *Proceedings of the 12th ACM conference on year=Proceedings of the International AAAI Conference on Web and Social Media*.
[14] Bent Fuglede and Flemming Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *Proceedings of the 2004 International Symposium on Information Theory*.
[15] Y. Ge, S. Liu, R. Gao, Y. Xian, Y. Li, X. Zhao, C. Pei, F. Sun, J. Ge, W. Ou, and Y. Zhang. 2021. Towards long-term fairness in recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 445–453.
[16] Y. Ge, X. Zhao, L. Yu, S. Paul, D. Hu, C.-C. Hsieh, and Y. Zhang. 2022. Toward pareto efficient fairness-utility trade-off in recommendation through reinforcement learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 316–324.
[17] Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-aware ranking in search & recommendation systems with application to LinkedIn talent search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
[18] Ananya Gupta, Eric Johnson, Justin Payan, Aditya Kumar Roy, Ari Kobren, Swetasudha Panda, Jean-Baptiste Tristan, and Michael Wick. 2021. Online post-processing in rankings for fair utility maximization. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.
[19] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of opportunity in supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*.
[20] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Trans* (2015).
[21] Lauren Kirchner Jeff Larson, Surya Mattu and Julia Angwin. 2016. How we analyzed the compas recidivism algorithm. In *ProPublica*.
[22] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*.
[23] Andreas Krause and Carlos Guestrin. 2007. Near-optimal observation selection using submodular functions. In *Proceedings of the AAAI*. . 1650–1654.
[24] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2021. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*.
[25] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2021. Towards personalized fairness based on causal notion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[26] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*.
[27] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2019. A survey on bias and fairness in machine learning. *ACM Computing Surveys* (2019).

[28] Natwar Modani, Deepali Jain, Ujjawal Soni, Gaurav Kumar Gupta, and Palak Agarwal. 2017. Fairness aware recommendations on Behance. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*.
[29] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. 2020. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[30] Mohammadmehdi Naghiaei, Hossein Rahmani, and Yashar Deldjoo. 2022. CPFair: Personalized consumer and producer fairness re-ranking for recommender systems. *In Proceedings of the International ACM SIGIR Conference on Research & Development in Information Retrieval*.
[31] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functionsI. *Mathematical programming* (1978).
[32] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-aware unbiased learning to rank for top-k rankings. *In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[33] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying online and counterfactual learning to rank: A novel counterfactual estimator that effectively utilizes online interventions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*.
[34] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for selection bias in learning-to-rank systems. In *Proceedings of the 29th International Conference on World Wide Web*.
[35] Gourab K. Patro, Arpita Biswas, Niloy Ganguly, Krishna P. Gummadi, and Abhijnan Chakraborty. 2020. FairRec: Two-sided fairness for personalized recommendations in two-sided platforms. In *Proceedings of the Web Conference 2020*.
[36] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
[37] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. In *Proceedings of the Conference on Neural Information Processing Systems*.
[38] Ashudeep Singh, David Kempe, and Thorsten Joachims. 2021. Fairness in ranking under uncertainty. *Advances in Neural Information Processing Systems* (2021).
[39] Harald Steck. 2018. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*.
[40] Tom Sühr, Asia J. Biega, Meike Zehlike, Krishna P. Gummadi, and Abhijnan Chakraborty. 2019. Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
[41] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. 2021. Mixture-based correction for position and trust bias in counterfactual learning to rank. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.
[42] Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When inverse propensity scoring does not work: Affine corrections for unbiased learning to rank. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.
[43] Lequn Wang and Thorsten Joachims. 2021. User fairness, item fairness, and diversity for rankings in two-sided markets. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*. 23–41.
[44] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*.
[45] Yao Wu, Jian Cao, Guandong Xu, and Yudong Tan. 2021. TFROM: A two-sided fairness-aware recommendation model for both customers and providers. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
[46] Tao Yang and Qingyao Ai. 2021. Maximizing marginal fairness for dynamic learning to rank. In *Proceedings of the Web Conference 2021*.
[47] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. Fa* ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*.
[48] Meike Zehlike and Carlos Castillo. 2020. Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of The Web Conference 2020*.
[49] Ziwei Zhu, Jianling Wang, and James Caverlee. 2021. Fairness-aware personalized ranking recommendation via adversarial learning. arXiv:2103.07849. Retrieved from https://arxiv.org/abs/2103.07849

Received 7 March 2023; revised 22 November 2023; accepted 28 January 2024