OPTIMIZING RANKING EFFECTIVENESS AND FAIRNESS

by

Tao Yang

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

December 2023

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Tao Yang**
has been approved by the following supervisory committee members:

Jeff M. Phillips ,	Chair(s)	**TBD** Date Approved
Qingyao Ai ,	Member	**TBD** Date Approved
Shandian Zhe ,	Member	**TBD** Date Approved
Aditya Bhaskara ,	Member	**TBD** Date Approved
Parth Gupta ,	Member	**TBD** Date Approved

by **Mary W. Hall** , Chair/Dean of
the Department/College/School of **Computing**
and by **David B. Kieda** , Dean of The Graduate School.

ABSTRACT

Advanced ranking techniques have improved AI-powered information services that significantly changed people's lives. For example, search engines that rank information according to their utilities to user's queries have helped billions of people better finish their tasks in daily work; recommendation systems that rank products/movies/news according to the user's interests have completely changed how people discover information every day. Therefore, how to construct and optimize ranking systems is one of the most important research problems in the field of Information Retrieval (IR). When optimizing ranking systems, there are two important criteria to measure the quality of result rankings in IR systems. The first criterion is *ranking effectiveness*, which refers to the ability of a ranking system to effectively present results based on their relevance to the users' needs. The second criterion is *ranking fairness*, which refers to the ability of a ranking system to present results fairly. Many approaches have been proposed to optimize ranking effectiveness and fairness, yet they still suffer from limitations like over-simplistic modeling, computational inefficiency, and suboptimal performance. In this dissertation, we develop practical, efficient, and effective methods to address those issues.

There are two main parts in this dissertation. In the first part, we investigated methods of effectiveness optimization. We first introduced a new type of bias in ranking effectiveness optimization, exploitation bias, that is caused by exploiting user behavior as ranking features and leads to suboptimal performance for existing ranking methods. To address the exploitation bias, we propose an uncertainty-aware empirical Bayes-based ranking algorithm. The proposed algorithm effectively mitigates exploitation bias and reaches superior ranking performance by discriminately treating behavior and non-behavior signals in input features in a Bayes way.

In the second part, we further investigated methods for ranking effectiveness and fairness joint optimization. Firstly, we found that the uncertainty in relevance estimation can make existing fair ranking methods sub-optimal. To address it, we proposed a novel

Marginal-Certainty-aware Fair algorithm that jointly optimizes effectiveness and fairness while mitigating the uncertainty in relevance estimation. Secondly, we found that existing fair ranking methods are mostly greedy algorithms that greedily optimize rankings for the next immediate session. We developed a fair ranking algorithm that plans ahead by jointly optimizing multiple ranklists together to reach a global optimum.

For my parents.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Advanced ranking techniques have revolutionized AI-powered information services, leading to profound changes in people's lives. For instance, search engines that prioritize information relevance to user queries have significantly enhanced productivity for billions in their daily work tasks. Likewise, recommendation systems, which rank products, movies, or news based on individual user preferences, have completely transformed the way people discover information on a daily basis. Consequently, the construction and optimization of ranking systems stand out as one of the foremost research challenges within the field of Information Retrieval (IR).

When it comes to optimizing ranking systems, there are two crucial criteria for assessing the quality of result rankings. The first criterion is known as "ranking effectiveness", which pertains to a ranking system's capacity to efficiently present results according to their relevance to users' needs. For instance, by training a ranking model to directly predict the relevance of each item, we can arrange items in the descending order of their relevance scores, thereby creating ranked lists. This approach facilitates user convenience by allowing them to focus their attention on the top-ranked items, saving them time and effort in satisfying their information needs.

The second criterion is "ranking fairness", which concerns the ability of a ranking system to offer results in a fair manner. For instance, in the context of job recommendations, if a ranking system solely emphasizes ranking effectiveness by ordering items exclusively by relevance, only a limited number of top candidates will consistently receive user attention. This could lead to a situation where other highly qualified candidates are rarely considered for positions. Therefore, it is imperative to strike a balance between effectiveness and fairness in ranking optimization.

Numerous methodologies have been proposed to enhance ranking effectiveness and

fairness. However, these methods often suffer from shortcomings such as oversimplified modeling, computational inefficiencies, and suboptimal performance. In the course of this dissertation, we develop practical, efficient, and effective approaches to address these challenges.

1.1 Challenges in Ranking optimization

Ranking optimization is the key problem in IR, and there are several fundamental challenges.

- Modern ranking systems mostly employ learning-to-rank (LTR) models, where relevance signals derived from user behavior are used to optimize ranking systems. While prior research has illustrated the effectiveness of incorporating user behavior signals, such as clicks, into LTR algorithms as features, we posit that existing LTR algorithms, which treat behavior and non-behavior signals in input features indiscriminately, may yield suboptimal outcomes in practical applications. This is due to the inherent challenge posed by the utilization of user behavior signals since user behavior signals are usually strongly correlated with the ranking objective and can only be gathered for items that have already been exposed to users. Consequently, the ranking system will be biased toward items that have already been exposed to users and those items will have an unfair advantage over others. We refer to the unfair advantage as the exploitation bias that detrimentally impacts the long-term performance of the system.
- Many fair-ranking approaches have been proposed to optimize ranking effectiveness and fairness at the same time. Nevertheless, as demonstrated in this dissertation, these techniques only achieve sub-optimal performance since they predominantly rely on relevance estimation, often without considering the associated uncertainty, namely, the variance associated with the estimated relevance.
- Besides, our investigation also reveals that a majority of the current fair ranking techniques rely on greedy algorithms that prioritize optimizing rankings for the immediate, upcoming session or request. As outlined in this study, adhering to such a short-term, myopic approach will result in suboptimal performance over the long run.

Recognizing the challenge of addressing all the issues at once, we have divided the project into multiple tasks, each dedicated to addressing distinct facets of the problem. Presented below are the principal contributions and outcomes achieved in this dissertation:

1.2 Contributions

The contribution of this thesis can be summarized as follows:

- **Mitigating Exploitation Bias in Learning to Rank with an Uncertainty-aware Empirical Bayes Approach**

  In this project, we address the issue of exploitation bias, and we introduce EBRank, an empirical Bayes principles based ranking algorithm. Specifically, in addressing the exploitation bias stemming from behavioral features within ranking models, EBRank first relies solely on non-behavioral features to establish a prior estimation of relevance. Throughout the deployment and serving of ranking systems, EBRank leverages observed user behaviors to iteratively refine the posterior relevance estimation via empirical Bayes modeling, instead of concatenating these behaviors as features directly into the ranking models. Furthermore, EBRank incorporates an uncertainty-aware exploration strategy, actively seeking to gather user behaviors for empirical Bayesian modeling, thereby enhancing the overall ranking performance. Empirical experiments conducted on three publicly available datasets substantiate the effectiveness, practicality, and notable superiority of EBRank over other state-of-the-art ranking algorithms.

  We will introduce this work in Chapter 3

- **Marginal-Certainty-aware Fair Ranking Algorithm**

  We introduce a novel algorithm called MCFair, which stands for Marginal-Certainty-aware Fair ranking algorithm, to address the inherent uncertainty in relevance estimation when optimizing ranking fairness. MCFair simultaneously optimizes fairness and user utility while the relevance estimation is continuously updated. Within the framework of MCFair, we first formulate a ranking objective that considers uncertainty, fairness, and ranking effectiveness. Subsequently, we employ the gradient of this ranking objective directly as the ranking score. Theoretically, we proved

that MCFair is optimal for the aforementioned ranking objective. Empirically, our findings demonstrate that on semi-synthesized datasets, MCFair exhibits effectiveness and practicality, consistently delivering superior performance in comparison to state-of-the-art fair ranking methods.

We will introduce this work in Chapter 4

- **FARA: Future-aware Ranking Algorithm for Fairness Optimization**
  We introduce a novel fair ranking algorithm called FARA, which stands for Future-Aware Ranking Algorithm. In contrast to the conventional approach of myopically optimizing rankings for the immediate session, FARA adopts a forward-thinking strategy by simultaneously optimizing multiple ranklists and preserving them for future sessions. The key methodology employed by FARA involves leveraging Taylor expansion of the fairness objective to assess how future ranklists will impact the overall fairness of the system. Building on this analysis, FARA implements a two-phase optimization algorithm. In the first phase, it addresses an optimal future exposure planning problem, and in the second phase, it constructs the optimal ranklists based on the previously determined future exposure plan. Theoretically, we provide evidence to establish that FARA is the optimal solution for the joint optimization of ranking relevance and fairness of multiple future ranklists. Empirically, our extensive experiments conducted on three semi-synthesized datasets demonstrate the efficiency and effectiveness of FARA.

  We will introduce this work in Chapter 5

1.3 Dissertation Organization

The rest of this dissertation is structured as follows:

In Chapter 2, we provide some notations and mathematical backgrounds about ranking optimization that are essential to our work. Then we review related prior work in ranking optimization.

In Chapter 3, we introduce the *Mitigating Exploitation Bias in Learning to Rank with an Uncertainty-aware Empirical Bayes Approach* [1], an uncertainty-aware empirical Bayes-based ranking algorithm that optimizes ranking effectiveness.

In Chapter 4, we introduce the *Marginal-Certainty-aware Fair Ranking Algorithm* [2],

which jointly optimizes fairness and ranking effectiveness, while relevance estimation is constantly updated in an online manner.

In Chapter 5, we introduce the *FARA: Future-aware Ranking Algorithm for Fairness Optimization* [3], which plans ahead by jointly optimizing fairness and ranking effectiveness of multiple future ranklists together.

In Chapter 6, at the end, we summarize the dissertation and discuss future directions.

CHAPTER 2

BACKGROUND

Ranking techniques have been extensively studied and used in modern Information Retrieval (IR) systems such as search engines, recommender systems, etc. When optimizing ranking systems, ranking effectiveness and ranking fairness are two important criteria to measure the quality of result rankings. Many approaches have been proposed to optimize ranking effectiveness and fairness, yet they still suffer from limitations like over-simplistic modeling, computational inefficiency, and suboptimal performance.

In this section, we give the background knowledge for this dissertation. A summary of notations is shown in Table 2.1.

2.1 The Workflow of Ranking Service

In Figure 2.1, we show the workflow of ranking service. At time step t, a user issues a query, and there are several candidate items corresponding to this query. Then the relevance estimator predicts the relevance of each candidate item and the ranking optimization methods will generate the ranked list by optimizing the ranking objective based on relevance estimation. There are a few different ranking objectives can be adopted here. For example, the ranking objective can be to jointly optimize effectiveness and fairness [4]. After examining the ranked list, the user provides their feedback, such as clicks. With user's feedback, the relevance estimator updates relevance estimation for future ranking optimization.

2.2 Ranking Utility Measurement

Ranking is a two-sided market from which users and item providers both draw utility. Optimization and evaluation should consider both sides.

Figure 2.1: Workflow of ranking services.

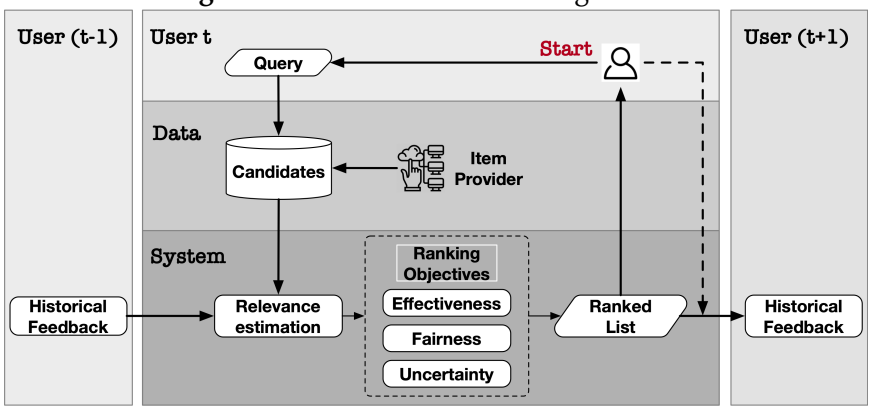


Table 2.1: A summary of notations.

$d, q, D(q)$	For a query q, $D(q)$ is the set of candidates items. $d \in D(q)$ is an item.
e, r, c	All are binary random variables indicating whether an item d is examined ($e = 1$), perceived as relevant ($r = 1$) and clicked ($c = 1$) by a user respectively.
R, ρ_i, E, π, n	$R = P(r = 1\|d)$, is the probability of an item d perceived as relevant. $\rho_i = P(e = 1\|rnk(d) = i, \pi)$ is the examination probability of item d when it is put in i^{th} rank in a ranklist π. E is item's accumulated exposure (see Eq.2.10). n is the number of times item d has been presented to users.
k_s, k_c	Users will stop examining items lower than rank k_s due to selection bias (see Eq. 2.14). k_c is the cutoff prefix to evaluate cNDCG and $k_c \leq k_s$.
x^b, x^{nb}	x^b denotes ranking features derived from user feedback behavior, while x^{nb} denotes ranking features derived from non-behavior features.

2.2.1 The User-side Utility (Effectiveness)

Before introducing the user-side utility, we define the relevance $R(d,q)$ as the probability of an item d to be relevant to query q:

$$R(d,q) = P(r = 1|d,q) \tag{2.1}$$

where r is a binary random variable indicating d perceived by a user as relevant to query q or not. As users are the main clients of ranking systems, it is important to optimize and evaluate ranking performance from the user side. The user-side utility, also referred to as *effectiveness*, is usually used to measure a ranking system's ability to put relevant items on top ranks. One widely-used user-side utility measurement is Discounted Cumulative Gain [5], denoted as DCG. For a ranked list π corresponding to a query q, we define $DCG@k_c$ as:

$$DCG@k_c(\pi) = \sum_{i=1}^{k_c} R(\pi[i], q)\lambda_i \tag{2.2}$$

where $\pi[i]$ indicates the i^{th} ranked item in the ranked list π; $R(\pi[i], q)$ indicates item $\pi[i]$'s relevance to query q; cutoff k_c indicates the top ranks we evaluate; λ_i indicates the

weight we put on i^{th} rank. λ_i usually monotonically decreases as rank i increases since top ranks are generally more important. For example, λ_i is sometimes set to $\frac{1}{\log_2(i+1)}$. We follow [6] and set λ as the examining probability ρ:

$$DCG@k_c(\pi) = \sum_{i=1}^{k_c} R(\pi[i], q) \cdot \rho_i \tag{2.3}$$

where ρ_i is users' examining probability of the i^{th} item in π, i.e.,

$$\rho_i = P(e_i = 1) \tag{2.4}$$

where e_i is a binary variable indicating the i^{th} item being examined or not. Then, we can define the normalized-*DCG* (*NDCG*) by normalizing $DCG@k_c(\pi)$ with $DCG@k_c(\pi^*)$:

$$NDCG@k_c(\pi) = \frac{DCG@k_c(\pi)}{DCG@k_c(\pi^*)} \tag{2.5}$$

where π^* is the ideal ranked list constructed by arranging items by their true relevance. $DCG@k_c(\pi^*)$ is also referred to as $IDCG@k_c$ to normalize $DCG@k_c(\pi_\tau)$ and $NDCG@k_c(\pi) \in [0,1]$. Furthermore, we could define Cumulative NDCG (cNDCG) as:

$$\textit{eff} = cNDCG@k_c = \sum_{\tau=1}^{t} \gamma^{t-\tau} NDCG@k_c(\pi_\tau) \tag{2.6}$$

$$\textit{Cum-NDCG@}k_c = \sum_{\tau=1}^{t} \gamma^{t-\tau} NDCG@k_c(\pi_\tau) \tag{2.7}$$

where $0 \leq \gamma \leq 1$ is the discounted factor, t is the current time step. Note that γ is usually set as a constant for all time steps. Compared to NDCG, cNDCG can better evaluate effectiveness for online ranking services [7]. Besides, by ignoring γ, we can get the average NDCG as,

$$\begin{aligned} \textit{aver-NDCG@}k_c &= \frac{\sum_{\tau=1}^{t} DCG@k_c(\pi_\tau)}{tDCG@k_c(\pi^*)} \\ &= \frac{\sum_{d \in D(q)} R(d) \Big(\sum_{\tau=1}^{t} \sum_{j=1}^{k_c} P_j \mathbb{1}_{\pi_i[j]==d} \Big)}{tDCG@k_c(\pi^*)} \\ &= \frac{\sum_{d \in D(q)} R(d) E^t@k_c(d)}{tDCG@k_c(\pi^*)} \end{aligned} \tag{2.8}$$

where t is the current time step, $E^t@k_c(d)$ is the cumulative exposure at top k_c ranks,

$$E^t@k_c(d) = \sum_{\tau=1}^{t} \sum_{j=1}^{k_c} P_j \mathbb{1}_{\pi_i[j]==d} \tag{2.9}$$

where $\pi_i[j]$ indicates the j^{th} item in ranklist π_i, $\mathbb{1}$ is an indicator function which means we only accumulate item d's exposure.

2.2.2 The Provider-side Utility (Fairness)

Since items' rankings can determine their providers' utility, it is also important to optimize and evaluate ranking performance from the provider's perspective. In the literature, **Provider-side Utility**[1] is used to measure a ranking system's ability to create a fair environment for items and their providers. Since a fair environment should let similar items be treated similarly, items of similar relevance should get similar exposure in this fair ranking system, i.e., the amortized fairness principle [6, 8]. Following existing works [6, 8, 9, 10], item d's exposure is defined as its accumulated examination probability:

$$E(d) = \sum_{i=1}^{t} \sum_{j=1}^{k} \rho_j \mathbb{1}_{\pi_i[j]==d} \tag{2.10}$$

where $\pi_i[j]$ indicates the j^{th} item in ranked list π_i, $\mathbb{1}$ is the indicator function which indicates that ρ_j will contribute to $E(d)$ only when $\pi_i[j]$ is item d. Then we follow [11] to define the unfairness tolerance of a ranking algorithm as:

$$\begin{aligned} unfairness &= \frac{1}{n(n-1)} \sum_{d_x \in D(q)} \sum_{d_y \in D(q)} \Big(E(d_x)R(d_y) - E(d_y)R(d_x) \Big)^2 \\ fairness &= -unfairness \end{aligned} \tag{2.11}$$

where $D(q)$ is the set of candidate items for query q. This unfairness measures the average exposure disparity between item pairs, and fairness is just the negative of unfairness. Besides, we use R or $R(d)$ as $R(d,q)$ for simpler notations in the later formulation. The intuition of the above fairness measurement is that the optimal fairness can be achieved when item exposure is proportional to their relevance, i.e., $\forall d_x, d_y \in D(q)$, $\frac{E^t(d_x)}{R(d_x)} = \frac{E^t(d_y)}{R(d_y)}$. In other words, exposure fairness means we should let items of similar relevance get similar exposure. In this dissertation, we choose the exposure fairness evaluation proposed by Oosterhuis [11] instead of the original evaluation proposed in [6]. The reason for the choice is that the fairness evaluation in [6] needs to divide the exposure of an item by its relevance, i.e., $\frac{E^t(d)}{R(d)}$, which has zero denominator problem when item d is irrelevant, and $R(d)$ is near zero. This dissertation uses average unfairness across different queries to evaluate a ranking algorithm. We also refer to the average unfairness as the unfairness tolerance.

[1] we use item fairness and provider-side fairness interchangeably

2.3 Partial and Biased Feedback

As shown in Figure 2.1, users' feedback is usually used to update the relevance estimation. However, users' feedback could be partial and biased indicator of relevance since users only provide meaningful feedback for items that they have examined,

$$c = \begin{cases} r, & \text{if} \quad e = 1 \\ 0, & otherwise \end{cases} \tag{2.12}$$

where e, r, c are binary random variables indicating whether an item is examined, perceived as relevant, and clicked, respectively. Following existing works on click model [9, 12], we model the probability of click c as:

$$P(c = 1) = P(r = 1)P(e = 1) \tag{2.13}$$

where $P(e = 1)$ is the examination probability, $r = 1$ and $e = 1$ are independent. Following existing works[13, 14], we assume two kinds of biases in the examination in this dissertation.

Positional Bias [15, 16]: The examination probability drops along ranks (also called position), and we model it with $P(rank(d|\pi))$, where the examining probability only depends on the rank.

Selection Bias [13, 17]: This bias exists when not all of the items are selected to be shown to users, or some lists are so long that no user will examine the entire lists. We model this by assuming that items ranked lower than rank k_s won't be examined by the user:

$$P(e = 1|d, \pi) = \begin{cases} P(rank(d|\pi)), & \text{if} \quad rank(d|\pi) \leq k_s \\ 0, & otherwise \end{cases} \tag{2.14}$$

2.4 Related Work

First, I will introduce unbiased and online Learning to Rank (LTR) methods that optimize ranking effectiveness based on user clicks. Then, I will introduce related works that consider uncertainty in ranking optimization. Then, I will introduce different types of proposed definitions of ranking fairness. Finally, I will introduce fair ranking algorithms based on one type of fairness, i.e., amortized fairness, the fairness we consider in this dissertation.

2.4.1 Unbiased and Online LTR

The analysis of how to optimize effectiveness with user clicks for Learning to Rank (LTR) has been extensively studied in the last decade [18]. The major focus of these

studies is on how to effectively learn unbiased LTR models by training them with biased click signals [16, 19]. Specifically, the studies of how to actively remove biases in labels through online interpolations (i.e., online LTR) and how to address click bias from theoretical perspectives (i.e., unbiased LTR) have received considerable attention. For example, previous studies have developed different strategies to explore result relevance with bandit learning [20, 20, 21, 22, 23, 24, 25] or stochastic ranking sampling [26] in online LTR systems. To train LTR models with offline click logs, causal analysis techniques such as counterfactual learning [13, 27, 28, 29, 30] have also been widely adopted in extracting unbiased training objectives for LTR.

2.4.2 Uncertainty in Ranking

Model uncertainty has been widely studied in the community of ML and statistics [31, 32, 33]. In IR, one of the first studies that use model uncertainty for ranking is proposed by Zhu et al. [34], in which they use the variance of a probabilistic language model as a risk-based factor to improve the performance of retrieval models. Instead of optimizing ranking performance directly, there are also studies that use model uncertainty to improve query performance prediction [35, 36] and query cutoff prediction [37, 38]. Recently, as neural retrieval models have become popular in modern IR systems, uncertainty estimation techniques for deep learning models have been introduced into the studies of neural IR [39, 40]. It has been shown that model uncertainty in neural networks can help us better understand and analyze the behaviors of neural rankers, such as BERT-based models [41]. In contrast to previous studies, in this dissertation, I propose to use uncertainty for ranking exploration in LTR. Instead of optimizing short-term ranking metrics directly, my work of uncertainty-aware ranking optimization algorithm focuses on long-term ranking effect.

2.4.3 Ranking exploitation with behavior features

As an important relevance indicator, user behavior signals have been important components for constructing modern IR systems [42]. [43, 44] showed that incorporating user behavior data as features can significantly improve the ranking performance of top results. However, incorporating behavior features without proper treatments could hurt the effectiveness of LTR systems by amplifying the problem of over-exploitation and over-fitting, i.e., exploitation bias [14]. Oosterhuis and de Rijke [45], Li et al. [46], Kveton et al. [47]

discussed the generation problems and cold-start problems when using behavior signals. Some strategies were proposed to overcome the exploitation bias by predicting behavior features with non-behavior features [48, 49] or by actively collecting user behavior for new items [14, 46] via exploration.

2.4.4 Ranking Fairness

Along with the widespread application of machine learning techniques, determinations based on those techniques can significantly affect people's lives. Therefore, there are growing interests in understanding those determinations' social impacts [50, 51, 52, 53, 54, 55]. Among them, fairness gradually becomes one of the major concerns. *Fairness in Ranking* is a relatively new topic and draws a lot of attention in recent years. Given that ranking is a two-sided market, with customers on one side and product producers on another side, we need to consider customers' satisfaction as well as a fair environment for product producers [56, 57, 58]. However, proposed definitions of fairness in ranking vary a lot. Some of them focus on representation learning to achieve algorithmic fairness where relevance rating should be independent of the sensitive attribute [59, 60, 61, 62]. Abdollahpouri et al. [63] and Pitoura et al. [64] give detailed surveys in those areas. Another group of works take significantly different perspectives on the definition of ranking fairness. They typically focus on how to allocate exposure to items fairly. For example, post-processing methods [65, 66, 67] have been proposed to achieve group fairness, which guarantees a certain frequency of items from different groups or producers in top ranks. Patro et al. [68] proposes an allocation method, FairRec, to achieve individual fairness, which guarantees an equal frequency for each items in ranking lists without considering items' relevance. However, only considering the frequency myopically ignores that there exists a large skew in the distribution of exposure for different ranks such as position bias [16, 69, 70].

2.4.5 Amortized Fairness

A more reasonable strategy is to achieve exposure fairness by allocating exposure proportional to items' relevance [6, 8]. In Table 2.2, we make a comparison between several amortized fairness methods to see whether those methods work with personal relevance, whether they need to do down-sampling, and their computational complexity. Note that

Table 2.2: Comparison of different amortized fairness methods. Attributes include whether they can achieve amortized fairness, work with personal relevance or not, need to do down sampling or not, and their computational complexity. FairRec is not an amortized fairness method, but we still include it here for completeness.

Method	Attributes			
	Amortized fairness	Personal relevance	Down sampling	Compu. complex.
LP [6]	✓	✗	✗	High
ILP [8]	✓	✓	✓	Medium
FairCo [71]	✓	✓	✗	Low
FairRec [68]	✗	✓	✗	Low

we assume there are a large number of candidate items when comparing. Specifically, let's consider a ranking task where there are m users, n items, and length of ranking list returned to each user is k. Biega et al. [8] proposed to carry out m rounds integer linear program (ILP) with n^2 decision variables in each round to amortize exposure. Since the size of decision variables is a bottleneck for ILP solvers, Biega et al. [8] proposed a down-sampling step that helps to reduce the size of candidate sets in each round and there are $O(k^2)$ decision variables in each round. Instead of trying to amortize exposure dynamically, Singh and Joachims [6] adopts Linear Programming (LP) with n^2 decision variables to give a static probabilistic ranking, which is mostly infeasible, given a large number of items. Besides, LP methods assume one single relevance distribution for items, while the ILP method can work with personal relevance. Thus, the ILP method is more suitable in personalized ranking given that the distribution of personal relevance vary from person to person. Besides linear programming, Morik et al. [71] proposed a more efficient fair ranking algorithm, FairCo, which first determine each item's unfairness and then boost ranking score of under-exposed items with a proportional controller. Following a similar idea of FairCo, Yang and Ai [9] proposed a fair ranking method MMF which introduced the idea of maximal marginal fairness and used it to achieve better fairness at top ranks. Besides this, Yang et al. [4] proposed an offline post-processing fair ranking method, VerFair. This method took advantage of the knowledge of all users at the beginning in offline settings, to help achieve better fairness on top ranks given the same effectiveness. Unlike the postprocessing method mentioned above, PG-Rank [10] chooses to achieve amortized fairness in learning to rank procedure via policy learning. Oosterhuis

[11] proposed a stochastic Plackett-Luce (PL) ranking models to optimize relevance and amortized fairness.

In this dissertation, the fairness we are considering is not about learning fair representations to achieve a fair model where relevance prediction should be independent of some sensitive attribute [59, 60, 61]. Even if we can get a perfect and fair relevance prediction, unfair exposure distribution still exists since there exists a large skew in exposure of different ranks such as position bias [16, 69, 70]. In this dissertation, I will mainly focus on how to achieve fair exposure according to amortized fairness principle, i.e., a propositional relation between relevance and exposure.

CHAPTER 3

MITIGATING EXPLOITATION BIAS IN LEARNING TO RANK WITH AN UNCERTAINTY-AWARE EMPIRICAL BAYES APPROACH

Ranking is at the core of many artificial intelligence (AI) applications, including search engines, recommender systems, etc. Modern ranking systems are often constructed with learning-to-rank (LTR) models built from user behavior signals. While previous studies have demonstrated the effectiveness of using user behavior signals (e.g., clicks) as both features and labels of LTR algorithms, we argue that existing LTR algorithms that indiscriminately treat behavior and non-behavior signals in input features could lead to suboptimal performance in practice. Because user behavior signals often have strong correlations with the ranking objective and can only be collected on items that have already been shown to users, directly using behavior signals in LTR could create an exploitation bias that hurts the system performance in the long run.

To address the exploitation bias, we propose EBRank, an uncertainty-aware empirical Bayes based ranking algorithm. Specifically, to overcome exploitation bias brought by behavior features in ranking models, EBRank uses a sole non-behavior feature-based prior model to get a prior estimation of relevance. In the dynamic training and serving of ranking systems, EBRank uses the observed user behaviors to update posterior relevance estimation instead of concatenating behaviors as features in ranking models. Besides, EBRank additionally applies an uncertainty-aware exploration strategy to explore actively, collect user behaviors for empirical Bayesian modeling and improve ranking performance. Experiments on three public datasets show that EBRank is effective, practical and significantly outperforms state-of-the-art ranking algorithms.

3.1 Introduction

Ranking techniques have been extensively studied and used in modern Information Retrieval (IR) systems such as search engines, recommender systems, etc. Among different ranking techniques, learning to rank (LTR), which relies on building machine learning (ML) models to rank items, is one of the most popular ranking frameworks [72]. In particular, industrial LTR systems are usually constructed with user behavior feedback/signals since user behaviors (e.g., click, purchase) are cheap to get and direct indicate results' relevance from the user's perspective [73]. For example, previous studies [27, 73, 74] have shown that, instead of using expensive relevance annotations from experts, effective LTR models can be learned directly from training labels constructed with user clicks. Besides using clicks as training labels, many industrial IR systems have also considered features extracted from user clicks for their LTR models. For example, ranking features extracted from clicks are used in search engines like Yahoo and Bing [42, 75]. Agichtein et al. [43] has shown that, by incorporating user clicks as behavior features in ranking systems, the performance of competitive web search ranking algorithms can be improved by as much as 31% relative to the original performance.

However, without proper treatment, LTR with user behaviors can also damage ranking quality in the long term [14, 46, 47]. Specifically, user behavior signals usually have high correlations with the training labels, no matter whether the labels are constructed from expert annotations or from users' behavior. Such high correlation can easily make input features built from user behavior signals, referred to as the behavior features, overwhelm other features in training, dominate model outputs, and be over-exploited in inference. Such an over-exploitation phenomenon would hurt practical ranking systems when user behavior signals are unevenly collected on different candidate items [46, 48]. For example, we can only collect user clicks on items already presented to users. Items that lack historical click data, including new items that have not yet been presented to users, would be at a disadvantage in ranking. The disadvantage, referred to as the exploitation bias [14], can be more severe when we use user clicks/behaviors as both labels and features, which is a common practice in real-world LTR systems [14, 42, 48, 75, 76]. One similar concept is selection bias [17]. However, selection bias usually refers to the bias that occurs when user clicks are used as training labels. In contrast, exploitation bias goes one step further

and considers the bias that arises in a more realistic scenario where user behavior is both the training labels and features.

In this dissertation, we address the above exploitation bias with an uncertainty-aware empirical Bayesian based algorithm, EBRank. Specifically, we consider a general application scenario where a ranking system is built with user behavior signals (e.g. clicks) in both its input and objective function. We show that, without differentiating the treatment of behavior signals and non-behavior signals in input features, existing LTR algorithms could suffer severely from exploitation bias. By differentiating behavior signals and non-behavior signals, the proposed algorithm, EBRank, uses a sole non-behavior feature based prior model to give a prior relevance estimation. With more behavior data collected from the online serving process of a ranking system, EBRank gradually updates its posterior relevance estimation to give a more accurate relevance estimation. Besides, we also proposed a theoretically principled exploration algorithm that joins the optimization of ranking performance with the minimization of model uncertainty. Experiments on three public datasets show that our proposed algorithm can effectively alleviate exploitation bias and deliver superior ranking performance compared to state-of-the-art ranking algorithms. To summarize, our contributions are

- Through simulation experiments, we demonstrate that existing LTR algorithms suffer severely from exploitation bias.
- We propose EBRank, a Bayesian-based LTR algorithm that mitigates exploitation bias.
- We propose an uncertainty-aware exploration strategy for EBRank that optimizes ranking performance while minimizing model uncertainty.

3.2 Proposed Method

In this section, we first introduce some preliminary knowledge of this work, Then, we propose an uncertainty-aware Empirical Bayes (EB) based learning to rank algorithm, EBRank, which can effectively overcome exploitation bias. We formally introduce the proposed algorithm EBRank In Sec .3.2.2. And we dive into the theoretical derivation of EBRank, which includes ranking objectives, Empirical Bayes modeling, and uncertainty reduction in Sec. 3.2.3, 3.2.4 and 3.2.5, repectively.

3.2.1 Prior Knowledge

Features. In this dissertation, LTR features are categorized into two groups based on Qin et al. [42]. The first group is non-behavior features, denoted as x^{nb}, which show items' quality and the degree of matching between item and query. Example x^{nb} can be BM25, query length, tf-idf, features from pre-trained (large) language models, etc. Non-behavior features x^{nb} are usually stable and static. The second group is behavior features, denoted as x^b, which are usually derived from user behavior data, which can be click-through-rate, dwelling time, click, etc. x^b are direct and strong indicators of relevance from the user's perspective since x^b are collected directly from the user themselves. Unlike x^{nb}, x^b are dynamically changing and constantly updated. In this dissertation, we only focus on one type of user behavior data, i.e., clicks. Extending our work to other types of user behaviors is also straightforward, and we leave them for future studies.

Exploitation Bias. In Figure 3.1, we give a toy example to illustrate the exploitation bias. The exploitation bias usually happens because behavior features in LTR models will overwhelm other features since behavior features are strong indicators of relevance and highly correlated with training labels. In this example in Figure 3.1, due to the overwhelming importance of behavior features, newly introduced item G will be discriminated by the LTR model and ranked low when its user behavior information is missing.

Uncertainty in relevance estimation. In real-world applications, the true relevance R is usually unavailable. Relevance estimation, denoted as $\hat{R}$, is usually needed for ranking optimization. However, $\hat{R}$ usually contains uncertainty (variance), denoted as $Var[\hat{R}]$. Furthermore, we introduce the query-level uncertainty in relevance estimation,

$$U(q) = \sum_{d \in D(q)} Var[\hat{R}(d,q)] \tag{3.1}$$

which will be used to guide ranking exploration. We leave more advanced query-level uncertainty formulations for future study.

3.2.2 The proposed algorithm: EBRank

In this section, we give a big picture of the proposed Empirical Bayesian Ranking (EBRank) in Algorithm 3.1. Prior to serving users, we initialize a list, $\mathcal{H}$, to store ranking logs. At each time step, we append four elements, $[u_t, q_t, \pi_t, c_t]$, to $\mathcal{H}$. Here, u_t, q_t, π_t, c_t are the user, the query, the presented ranklist, and the user clicks at time step t. If initial

ranking logs exist prior to using EBRank, we will also append them to $\mathcal{H}$. Besides ranking logs, we initialize a model f_θ, referred to as the prior model, parameterized by θ, which takes non-behavior features as input. f_θ can be any trainable parameterized model, such as a neural network, tree-based model and etc. When EBRank begins to serve users ($t > 0$), new candidates will be constantly appended to each query's candidates set, $D(q_t)$. Then $\forall d \in D(q_t)$, we construct an auxiliary list $\mathcal{A}(d, q_t) = [\alpha, \beta, n, C, E]$, where

$$\begin{aligned}
[\alpha, \beta] &= f_\theta\Big(x^{nb}(d, q_t)\Big) \\
n &= \sum_{\tau \in T(t,d,q_t)} 1 \\
C &= \sum_{\tau \in T(t,d,q_t)} \frac{c(d|\pi_\tau)}{\rho_{rank(d|\pi_\tau)}} \\
E &= \sum_{\tau \in T(t,d,q_t)} \rho_{rank(d|\pi_\tau)}
\end{aligned} \tag{3.2}$$

$$T(t, d, q_t) = \Big[\tau \quad \text{if} \quad \mathbb{I}(d \in \pi_\tau)\mathbb{I}(q_\tau = q_t) \quad \text{for} \quad 0 \le \tau < t\Big]$$

where the prior model f_θ takes the non-behavior feature x^{nb} as input and exports two numbers, i.e., $[\alpha, \beta]$. $T(t, d, q_t)$ is a subset of historical time steps when item d is presented in query q_t's ranklists in the past and $\mathbb{I}$ is the indicator function. n is the number of times that item d has been presented for query q_t by time step $t-1$. $c(d|\pi_\tau)$ indicates whether item d is clicked or not. C is the sum of weighted clicks on item d, and the weight is its examination probability, E is the item's exposure which is the accumulation of examination probability.

3.2.2.1 Relevance estimation.

Based on $\mathcal{A}(d, q_t)$, firstly, we estimate item's relevance $\hat{R}(d, q_t)$ as,

$$\hat{R}(d, q_t) = \frac{C + \alpha}{n + \alpha + \beta} \tag{3.3}$$

which is based on our empirical Bayes modeling in Sec. 3.2.3. The relevance estimation $\hat{R}(d, q_t)$ is a blending between $\frac{C}{n}$ and $\frac{\alpha}{\alpha+\beta}$. When $n > 0$, $\frac{C}{n}$ is an unbiased estimation of true relevance $R(d, q_t))$ (see Eq. (3.17). Relevance estimation $\frac{C}{n}$ is limited as it requires $n > 0$, i.e., item d has been selected for query q_t before. Without this limitation, $\frac{\alpha}{\alpha+\beta}$, referred to as prior relevance estimation, is solely based on non-behavior features. Similar to $\frac{C}{n}$, $\frac{\alpha}{\alpha+\beta}$ also theoretically approximates the true relevance $R(d, q_t)$ given an perfectly optimized prior model f_θ (more details is in later sections).

With the blending of the two parts, $\hat{R}$ can overcome exploitation bias by nature since it will rely more on $\frac{\alpha}{\alpha+\beta}$ when n and C are small, i.e., new items. $\hat{R}$ gradually relies more on $\frac{C}{n}$ when n and C increase and start to dominate, i.e., more user behaviors are observed.

3.2.2.2 Ranking exploration & construction.

Besides relevance, we introduce an exploration score for item d,

$$MC(d,q_t) = \frac{\hat{R}(d,q_t)}{\Big(E+\alpha+\beta\Big)^2} \tag{3.4}$$

which boosts candidates that have a higher estimated relevance $\hat{R}(d,q_t)$ but a lower exposure E. The exploration score helps to gain the greatest certainty at the current time step according to our ranking uncertainty analysis in Sec. 3.2.5, With the relevance estimation and the exploration score, we construct ranklist π_t by sorting $\hat{R}(d,q_t)+\epsilon MC(d,q_t)$ in descending order, i.e.,

$$\pi_t = \arg_{sort\text{-}k}\Big(\hat{R}(d,q_t)+\epsilon MC(d,q_t)|\forall d \in D(q_t)\Big) \tag{3.5}$$

where a possible cutoff k might exist to show the top results only. ϵ is a hyper-parameter to balance the two parts. The ranklist construction is based on the ranking objective proposed in Sec. 3.2.3.

3.2.2.3 Prior model optimization.

The parameters θ in f_θ will be periodically updated with the following loss,

$$l(d,q) = \mathbb{I}_{n>0}\Big(\log B(\alpha,\beta) - \log B(C+\alpha, n-C+\beta)\Big) \tag{3.6a}$$

$$\mathcal{L} = \sum_{q\in Q}\sum_{d\in D(q)} l(d,q) \tag{3.6b}$$

where B denotes the beta function. Note that α,β are f_θ's outputs. The loss is based on our Empirical Bayes modeling in Sec. 3.2.4. The above loss to train f_θ is based on items that have been presented to users before ($n>0$) since only those items could possibly have user clicks, which is the best we can do. According to our empirical results, $f_\theta(x^{nb})$ generalizes well for items not presented before, i.e., $n=0$. If there exist some initial ranking logs, f_θ can also be trained based on them. More analysis of the loss can be found in Sec. 3.2.4.3.

3.2.3 Uncertainty-Aware ranking optimization

In this section, we explain why Eq. 3.5 is optimal for constructing ranklists.

Algorithm 3.1: EBRank

$t \leftarrow 0$;
$\mathcal{H} = [u_t, q_t, \pi_t, c_t]\ \forall t \in history$;
Initialize the trainable parameters θ for prior model f_θ;
Initialize hyper-parameter ϵ;
while *True* **do**
  $t \leftarrow t + 1$;
  User u_t issues a query q_t;
  if *New candidates introduced* **then**
    D_{q_t}.append(new candidates)
  $\forall d \in D_{q_t}$, construct $\mathcal{A}(d, q_t)$ via Eq. (3.2);
  With $\mathcal{A}(d, q_t)$, get $\hat{R}(d, q_t)$ via Eq. (3.16);
  With $\mathcal{A}(d, q_t)$ & $\hat{R}(d, q_t)$, get $MC(d, q_t)$ via Eq. (3.24);
  With $\hat{R}(d, q_t)$ & $MC(d, q_t)$, get π_t via Eq. (3.5);
  Present π_t to User u_t and collect user's clicks c_t;
  $\mathcal{H}.append([u_t, q_t, \pi_t, c_t]$;
  if *Prior-model-update* **then**
    Train f_θ with loss $\mathcal{L}$ in Eq. (3.6)

3.2.3.1 The uncertainty-aware ranking objective.

As shown in Figure 2.1, at time step t, a user issues a query q_t, and we propose the following uncertainty-aware ranking objective to optimize ranklist π_t,

$$\max_{\pi_t} \quad Obj = D\hat{C}G(\pi_t) - \epsilon \Delta U(\pi_t) \tag{3.7}$$

where $\Delta U(\pi_t) = U(q_t|\pi_t) - U(q_t)$ is the query-level uncertainty increment after presenting π_t to the user. $D\hat{C}G$ is the estimated DCG (see Eq.2.3) based on the estimated relevance $\hat{R}$ instead of the unavailable true relevance R. ϵ is the coefficient to balance the two parts. In Eq. (3.7), our real goal is to maximize the true DCG, but we only have the estimated $D\hat{C}G$ which is calculated based on estimated relevance. Hence, to effectively optimize DCG via the proxy of optimizing $D\hat{C}G$, we need an accurate $\hat{R}(d, q_t)$, which is why uncertainty is also minimized in Eq. (3.7). Here we choose to minimize the incremental uncertainty at time step t since only the incremental uncertainty is caused by π_t.

3.2.3.2 Ranking optimization.

To maximize ranking objectives in Eq. (3.7) and optimize π_t, we first reformulate $D\hat{C}G$ in Eq. (2.3) as,

$$D\hat{C}G(\pi_t) = \sum_{d \in D(q_t)} \hat{R}(d, q_t) \cdot \rho_{rank(d|\pi_t)} \tag{3.8}$$

As for $\Delta U(\pi_t)$ in Eq. (3.7), inspired by [77], we carry out a first-order approximation by considering incremental exposure ΔE,

$$\begin{aligned}\Delta U(\pi_t) &\approx \sum_{d\in D(q_t)} \frac{\partial U(q_t)}{\partial E(d,q_t)} \Delta E(d,q_t) \\ &= \sum_{d\in D(q_t)} \frac{\partial Var[\hat{R}(d,q_t)]}{\partial E(d,q_t)} \rho_{rank(d|\pi_t)} \\ &= \sum_{d\in D(q_t)} -MC(d,q_t)\rho_{rank(d|\pi_t)}\end{aligned} \tag{3.9}$$

where $\Delta E(d,q_t)$ is the incremental exposure that item d will get at time t, i.e., $\rho_{rank(d|\pi_t)}$, $MC(d,q_t)$ is the gradient of minus variance, dubbed **M**arginal **C**ertainty, the speed to gain additional certainty. Since $\rho_{rank(d|\pi_t)} \in (0,1)$ is relatively small, the first-order approximation in Eq. (3.9) should approximate well. In Eq. (3.9), we assume that $Var[\hat{R}(d_x,q_t)]$ and $E(d_y,q_t)$ are independent for different items d_x and d_y, so $\frac{\partial Var[\hat{R}(d_x,q_t)]}{\partial E(d_y,q_t)} = 0$. We will introduce how $Var[\hat{R}(d,q_t)]$ and $E(d,q_t)$ are related in Sec 3.2.5.

Finally, the ranking objective in Eq. (3.7) can be rewritten as

$$\max_{\pi_t} \sum_{d\in D(q_t)} \Big(\hat{R}(d,q_t) + \epsilon MC(d,q_t)\Big)\rho_{rank(d|\pi_t)} \tag{3.10}$$

By assuming that $\rho_{rank(d|\pi_t)}$ descends as $rank(d|\pi_t)$ goes lower, the optimal π_t should be generated by sorting items according to $\Big(\hat{R}(d,q_t)+\epsilon MC(d,q_t)\Big)$ in descending order, which validates Eq. (3.5).

3.2.4 Empirical Bayesian relevance model

In this section, we propose an empirical Bayesian relevance model that leads to relevance estimation Eq. (3.3) and loss function Eq. (3.6).

3.2.4.1 The observation modelling

At time step t, users issue the query q_t. When there is no position bias and the binary relevance judgments r are directly observable, the probability of observation of relevance judgments for a (d,q_t) pair prior to time step t is

$$P(r^*|\bar{R}) = \prod_{\tau\in T(t,d,q_t)} \Big((\bar{R})^{r(d|\pi_\tau)} \times (1-\bar{R})^{(1-r(d|\pi_\tau))}\Big) \tag{3.11}$$

where r^* denotes users' relevance judgments. $0 \le \bar{R} \le 1$, and $\bar{R}$ is a random variable that denotes our estimated probability of $r = 1$. $r(d|\pi_\tau)$ is the observation of random binary variable r at time τ.

However, user relevance judgments are not observable, and we can only observe user clicks for $\tau < t$, although clicks are biased indicator of relevance according to Eq. (2.12). In this dissertation, based on observable clicks, we introduce probability $P(c^*|\bar{R})$ as a proxy for $P(r^*|\bar{R})$,

$$\begin{aligned} P(c^*|\bar{R}) &= \prod_{\tau \in T(t,d,q_t)} \left((\bar{R})^{\frac{c(d|\pi_\tau)}{\rho_{rank(d|\pi_\tau)}}} \times (1-\bar{R})^{\left(1-\frac{c(d|\pi_\tau)}{\rho_{rank(d|\pi_\tau)}}\right)} \right) \\ &= (\bar{R})^C \times (1-(\bar{R}))^{n-C} \end{aligned} \tag{3.12}$$

where c^* denotes users' clicks. $c(d|\pi_\tau)$ indicates whether item d is clicked or not in ranklist π_τ, and $\rho_{rank(d|\pi_\tau)}$ is item d's examining probability in presented ranklist π_τ. We use $P(c^*|\bar{R})$ as a proxy of $P(r^*|\bar{R})$ since we noticed that $\log P(c^*|\bar{R})$ is an unbiased estimation of $\log P(r^*|\bar{R})$ [78, 79],

$$\begin{aligned} &\mathbb{E}_e[\log P(c^*|\bar{R})] \\ &= \sum_{\tau \in T(t,d,q_t)} \left(\frac{\mathbb{E}_e[c(d|\pi_\tau)]}{\rho_{rank(d|\pi_\tau)}} \log(\bar{R}) + (1 - \frac{\mathbb{E}_e[c(d|\pi_\tau)]}{\rho_{rank(d|\pi_\tau)}}) \log(1-\bar{R}) \right) \\ &= \log P(r^*|\bar{R}) \end{aligned} \tag{3.13}$$

where expectation $\mathbb{E}_e[c(d|\pi_\tau)] = \rho_{rank(d|\pi_\tau)} r(d|\pi_\tau)$ given Eq. (2.12) and Eq. 2.13. Although the proof above takes a logarithm and the log-likelihood is unbiased instead of likelihood itself, $P(c^*|\bar{R})$ is still an effective proxy for $P(r^*|\bar{R})$, which is validated by our empirical results. One may note that [78, 79] have similar theoretical analysis here, but our method is fundamentally different from theirs. They use $-\log P(c^*|\bar{R})$ as the final ranking loss function, while we use $P(c^*|\bar{R})$ as the observation probability, just one part of the many components of our Bayes model.

3.2.4.2 The prior & posterior distribution.

Because the formulation $P(c^*|\bar{R})$ is similar to a binomial distribution, we choose the binomial distribution's conjugate prior, the Beta distribution, as the prior distribution, i.e., the prior relevance $\bar{R} \sim Beta(\alpha, \beta)$,

$$P(\bar{R}|\theta) = \frac{\bar{R}^{\alpha-1}(1-\bar{R})^{\beta-1}}{B(\alpha,\beta)} \tag{3.14}$$

where B denotes the beta function.[1]. According to the theory of conjugate distribution [80], the posterior distribution of $\bar{R}$ also follows a beta distribution,

$$\bar{R} \sim Beta(C+\alpha, n-C+\beta) \tag{3.15}$$

[1] The beta distribution and the beta function are denoted as *Beta* and *B* respectively.

And we use the expectation of the posterior distribution as the posterior relevance estimation $\hat{R}$ to rank items (see Eq. (3.5),

$$\hat{R}(d,q_t) = \mathbb{E}[\bar{R}] = \frac{C+\alpha}{n+\alpha+\beta} \tag{3.16}$$

When $n > 0$, $\frac{C}{n}$ is an unbiased estimation of true relevance $R(d,q_t)$)

$$\begin{aligned}\mathbb{E}_c[\frac{C}{n}] &= \frac{1}{n}\sum_{\tau\in T(t,d,q_t)}\frac{\mathbb{E}_c[c(d|\pi_\tau)]}{\rho_{rank(d|\pi_\tau)}} \\ &= \frac{1}{n}\sum_{\tau\in T(t,d,q_t)}\frac{\rho_{rank(d|\pi_\tau)}R(d,q_t)}{\rho_{rank(d|\pi_\tau)}} \\ &= R(d,q_t)\end{aligned} \tag{3.17}$$

3.2.4.3 Update prior distribution with observations

(α,β), decided by θ, are critical components in the posterior relevance estimation $\hat{R}$. To get the optimal θ, we perform the maximum a posterior (MAP) by maximizing the marginal likelihood of the observed data [81] i.e., $P(c^*|\theta)$.

$$\begin{aligned}P(c^*|\theta) &= \int_{\bar{R}} P(c^*|\bar{R})P(\bar{R}|\theta)d\bar{R} \\ &= \frac{\int_0^1 \bar{R}^{C+\alpha-1}(1-\bar{R})^{n-C+\beta-1}d\bar{R}}{B(\alpha,\beta)} \\ &= \frac{B(C+\alpha,n-C+\beta)}{B(\alpha,\beta)}\end{aligned} \tag{3.18}$$

Maximizing the above probability is also equivalent to minimizing $-\log P(c^*|\theta)$, which is exactly the loss in 3.6a.

To investigate what is the optimal (α,β) during training, we take the derivative of the loss function,

$$\frac{\partial l(d,q)}{\partial\alpha} = \psi(\alpha) - \psi(\alpha+\beta) - \Big(\psi(C+\alpha) - \psi(n+\alpha+\beta)\Big) \tag{3.19}$$

where ψ is the Digamma function. Usually, by setting, $\frac{\partial l(d,q)}{\partial\alpha} = 0$, we could know the optimal (α^*,β^*). However, to our knowledge, it is difficult to directly get (α^*,β^*) from the above equation. Since $\psi(x) \approx \log(x)$ with error $O(\frac{1}{x})$[82], we use log function to substitute ψ in Eq. (3.19), and set $\frac{\partial l(d,q)}{\partial\alpha} = 0$, and we get,

$$\frac{\alpha^*}{\alpha^*+\beta^*} - \frac{C+\alpha^*}{n+\alpha^*+\beta^*} = 0 \tag{3.20}$$

It is straightforward to see that the optimal prior estimation should output (α^*,β^*) that satisfies $\frac{\alpha^*}{\alpha^*+\beta^*} = \frac{C}{n}$, where $\frac{C}{n}$ is an unbiased estimation of relevance $\bar{R}$ given Eq. (3.17) as long as $n > 0$. In Eq. (3.20), α^*,β^* are not unique. To make α,β in Eq. (3.20) have unique values, we fix β and only learn α for simplicity. We leave how to get unique α,β without

fixing one of them to future works.

3.2.5 Estimation of Marginal Certainty.

In this section, we introduce how to get the exploration score in Eq. 3.4. As indicated in Eq. (3.1&3.9), to get the $MC(d,q_t)$, we first need to get the variance of $\hat{R}(d,q_t)$. For simplicity, we assume that the only random variable in $\hat{R}(d,q_t)$ is $c(d|\pi_\tau)$, where $c(d|\pi_\tau)$ and $\hat{R}(d,q_t)$ are associated via C. Firstly, the variance of $c(d|\pi_\tau)$ is

$$\begin{aligned} Var[c(d|\pi_\tau)] &= \mathbb{E}_c[c^2(d|\pi_\tau)] - \mathbb{E}_c^2[c(d|\pi_\tau)] \\ &= \mathbb{E}_c[c(d|\pi_\tau)] - \mathbb{E}_c^2[c(d|\pi_\tau)] \\ &< \rho_{rank(d|\pi_\tau)} R(d,q_t) \end{aligned} \tag{3.21}$$

where

$$c^2(d|\pi_\tau) = c(d|\pi_\tau) \tag{3.22}$$

since $c(d|\pi_\tau)$ is a binary random variable. $\mathbb{E}_c[c(d|\pi_\tau)] = \rho_{rank(d|\pi_\tau)} R(d|\pi_t)$ according to Eq. 2.13. According to the linearity of expectation, we can get the variance of $\hat{R}(d,q_t)$,

$$\begin{aligned} Var[\hat{R}(d|q_t)] &= \frac{Var[C]}{(n+\alpha+\beta)^2} \\ &= \frac{\sum_{\tau \in T(t,d,q_t)} \frac{1}{\rho^2_{rank(d|\pi_\tau)}} Var[c(d|\pi_\tau)]}{(n+\alpha+\beta)^2} \\ &< \frac{R(d,q_t) \sum_{\tau \in T(t,d,q_t)} \frac{1}{\rho_{rank(d|\pi_\tau)}}}{(n+\alpha+\beta)^2} \\ &< \frac{R(d,q_t) \frac{n}{\rho_{min}}}{(n+\alpha+\beta)^2} \\ &< \frac{R(d,q_t) 1/\rho_{min}}{(n+\alpha+\beta)} \\ &\doteq \frac{R(d,q_t)}{E+\alpha+\beta} \end{aligned} \tag{3.23}$$

where we assume there exists the smallest examining probability ρ_{min} for presented item d. In the last step, we ignore the constant factor, i.e., $1/\rho_{min}$. We use the above upper bound as an approximation of variance, which works well according to our empirical results. Given Eq. (3.9), we can get $MC(d,q_t)$ by taking derivative of $(-Var[\hat{R}])$ with respect to E,

$$\begin{aligned} MC(d,q_t) &= \frac{\partial(-Var[\hat{R}(d|q_t)])}{\partial E} \\ &\approx \frac{\hat{R}(d,q_t)}{(E+\alpha+\beta)^2} \end{aligned} \tag{3.24}$$

where we use $\hat{R}$ to substitute relevance R since R is unavailable.

Table 3.1: Datasets statistics. For each dataset, the table below shows the number of queries, the average number of candidate docs for each query, the number of features, the relevance annotation y's range, and the feature id of the BM25 to be used in our experiments.

Datasets	# Queries	#AverDocs	# Features	y	BM25
MQ2007	1643	41	46	$0-2$	25^{th}
MSLR-10k	9835	122	133	$0-4$	110^{th}
MSLR-30k	30995	121	133	$0-4$	110^{th}

3.3 Experiments

In this section, we evaluate the effectiveness of the proposed method with semi-simulation experiments on public LTR datasets. To facilitate reproducibility, we release our code [2]

3.3.1 Experimental setup

3.3.1.1 Dataset

In the semi-simulation experiments, we will adopt three publicly available LTR datasets, i.e., MQ2007, MSLR-10K, and MSLR-30K, with data statistics shown in Table 3.1. The dataset queries are partitioned into three subsets, namely training, validation, and test according to the 60%-20%-20% scheme. For each query-document pair of each dataset, relevance judgment y is provided. The original feature sets of MSLR-10K/MSLR-30K have three behavior features (i.e., feature 134, feature 135, feature 136) collected from user behaviors. To reliably evaluate our method, we remove them in advance. MQ2007 only contains non-behavior features. Thus, at the beginning of our experiments, all datasets only contain non-behavioral features (i.e., x^{nb}). It is worth mentioning that there are other widely-used large-scale LTR datasets accessible to the public, such as Yahoo! Letor Dataset [75] and Istella Dataset [83]. However, they cannot be used in this dissertation because they contain behavior features but do not reveal their identity.

3.3.1.2 Simulation of Search Sessions, Click and Cold-start

Similar to prior research [18, 20, 27, 84], we create simulated user engagements to evaluate different LTR algorithms. The advantage of the simulation is that it allows us to do online experiments on a large scale while still being easy to reproduce by researchers without access to live ranking systems [13]. Specifically, at each time step, we randomly

[2] https://github.com/Taosheng-ty/EBRank/

select a query from either the training, validation or test subset and generate a ranked list based on ranking algorithms. Following the methodology proposed by Chapelle et al. [85], we convert the relevance judgement to relevance probability according to $P(r = 1|d,q,\pi) = 0.1 + (1-0.1)\frac{2^y-1}{2^{y_{max}}-1}$, where y_{max} is 2 or 4 depending on the dataset. Besides relevance probability, the examination probability $\rho_{rank(\pi,d)}$ are simulated with

$$\rho_{rank(\pi,d)} = \frac{1}{\log_2(rank(\pi,d)+1)} \tag{3.25}$$

The $\rho_{rank(\pi,d)}$ is also the same examination probability used in Eq. 2.3 to compute *DCG*. For simplicity, we follow [9, 13, 14] to assume that users' examination $\rho_{rank(\pi,d)}$ is known in our experiment as many existing methods [74, 86, 87] have been proposed for it. With $P(r = 1|d,q,\pi)$ and $\rho_{rank(\pi,d)}$, according to Equation 2.13, clicks can sampled. We simulate clicks on the top 5 items, i.e., $k_s = 5$ in Equation 2.14. User clicks are simulated and collected for all three partitions. And we only use the sessions sampled from training partitions to train LTR models and sessions sampled from validation partitions to do validation. LTR models are evaluated and compared only based on test partitions. We collect clicks on validation and test queries in the simulation to construct the behavior features for their candidate document, which is used for inference only. In real-world LTR systems, behavior features are widely used in the inference of LTR models [42, 75].

The cold start scenario in ranking is an important part of our simulation experiments. We found two factors are essential for cold start simulation. Firstly, in real-world applications, new documents/items frequently come to the retrieval collection during the serving of LTR systems. To simulate new documents/items' coming, at the beginning of each experiment, we randomly sample only 5 to 10 documents for each query as the *initial candidate sets* D_q and mask all other documents. Then, at each time step t, with probability η ($\eta = 1.0$ by default), we randomly sample one masked document and add it to the candidate set D_q. Depending on the averaged number of document candidates and η, the number of sessions (time steps) we simulate for each dataset is

$$\#Session = \frac{\#Queries \times (\#AverDocs - 5)}{\eta} \tag{3.26}$$

where $\#Queries$ and $\#AverDocs$ are indicated in Table 3.1. The second factor for cold start simulation is that when a ranking algorithm usually is introduced in an LTR system, some documents/items already have collected user feedback in history. To simulate this, for each query, we simulated 20 sessions according to BM25 scores to collect initial user

behaviors for the initial candidate sets. The BM25 scores are already provided in the original datasets' features (see Table 3.1).

3.3.1.3 Baselines

To evaluate our proposed methods, we have the following ranking algorithms to compare,

- **BM25**: The method to collect initial user behaviors.
- **CFTopK**: Train a ranking model with Counterfactual loss that is widely used in existing works [14, 71, 78] and create ranked lists with items sorted by the model scores during ranking service.
- **CFRandomK**: Train a ranking model the same way as CFTopK, but randomly create ranked lists with items during ranking service.
- **CFEpsilon**: Train a ranking model the same way as CFTopK. Uniformly sample an exploration from $[0,1]$ and add to each item's score from the model[14]. Create ranked lists with items sorted by the score after addition during ranking service.
- **DBGD**[23]: A learning to rank method which samples one variation of a ranking model and compares them using online evaluation during ranking service.
- **MGD**[24]: Similar to DBGD but sample multiple variations.
- **PDGD**[26]: A learning to rank method which decides gradient via pairwise comparison during ranking service.
- **PRIOR**[48]: When $x^b = 0$, the method will train a behavior feature prediction model to give a pseudo behavior feature x^b.
- **UCBRank**[14]: uses relevance estimation from x^{nb} when an item is not presented and uses relevance estimation from x^b when an item has been presented. An upper Confidence Bound (UCB) exploration strategy is used as an exploration strategy.
- **EBRank**: Our method shown in Algorithm 3.1.

For those baselines, DBGD, MGD, PDGD, CFTopK, CFRankdomK, CFEpsilon are unbiased learning to rank algorithms that try to learn unbiased LTR models with biased click signals. We will investigate, in this dissertation, whether they can overcome exploitation bias or not. We compare those methods with two feature settings. The first setting is that we only use non-behavior features x^{nb}, referred to as **W/o-Behav**. The

second feature setting is that we use both behavior signals and non-behavior features. The feature setting is referred to as **W/-Behav(Concate)** when behavior-derived features and non-behavior features are concatenated together. **W/-Behav(Non-Concate)** means behavior signals and non-behavior features are combined using a non-concatenation way, such as the EB modeling used by EBRank. Method BM25 only has W/o-Behav results since it uses the non-behavior feature BM25 to rank items. CFTopK, CFRandomK, CFEpsilon, DBGD, MGD, and PDGD have ranking performance with both W/o-Behav and W/-Behav(Concate) feature settings, depending on whether behavior features are used or not. We follow Yang et al. [14] to use relevance's unbiased estimator

$$x^b = \frac{C}{n} \tag{3.27}$$

as the behavior feature, with the default value as 0 when $n = 0$. The default value for x^b has a significant influence on methods DBGD, MGD, PDGD, CFTopK, CFRankdomK, and CFEpsilon since those methods will concatenate x^b and x^{nb}. However, how to set the default x^b for each method is not within the scope of this work, and we leave it for future works. PRIOR only has ranking performance with W/-Behav(Concate) because PRIOR is designed to relieve exploitation bias when behavior features and non-behavior features are concatenated. For UCBRank and EBRank, they only have W/-Behav(Non-Concate) results since they have their own designed non-concatenation way to combine behavior signals and non-behavior features. However, our method EBRank is fundamentally different from UCBRank. UCBRank treats behavior features as independent evidence for relevance and linearly interpolates relevance estimation from x^b and x^{nb}. Besides, UCBRank adopts an Upper Confidence Bound exploration strategy. However, EBRank interpolates relevance estimation from x^b and x^{nb} from a Bayesian perspective and uses the marginal-certainty exploration strategy derived in Eq. 3.9 to guide exploration.

3.3.1.4 Implementation

Linear models are used for all methods except BM25. We retrain and update ranking model parameters periodically 20 times during the simulation. Compared to updating model parameters online, updating ranking logs, including user behaviors, is relatively easy and time-efficient, and we update them after each session. When we train the prior model according to loss in Eq. 3.6, we only train $\alpha = f_\theta(x^{nb})$ and fix $\beta = 5$ for simplicity,

which works well across all experiments.

3.3.1.5 Evaluation

We evaluate ranking baselines with two standard ranking metrics on the test set. The first one is the Cumulative NDCG (referred to as **Cum-NDCG**) (Eq.2.7) with the discounted factor $\gamma = 0.995$ (same γ used in [20, 21]) to evaluate the online performance of presented ranklists. The second metric is the standard **NDCG** (Eq.2.5), where each query's ranked list is generated by sorting scores from the final ranking models (excluding the exploration strategy part of each algorithm). The NDCG evaluates the offline performance of the learned ranking model. NDCG is tested in two situations: with (i.e., **Warm-NDCG**) or without (i.e., **Cold-NDCG**) behavior features collected from click history. In this way, our experiment can effectively evaluate LTR systems in scenarios where we encounter new queries with no user behavior history on any candidate documents (i.e., the *Cold-NDCG*) and the scenarios where user behavior exists due to previous click logs (i.e., the *Warm-NDCG*). For simplicity, we set the rank cutoff to 5 and compute iDCG in both Cum-NDCG and NDCG with all documents in each dataset. Significant tests are conducted with the Fisher randomization test [88] with $p < 0.05$. All evaluations are based on five independent trials and reported on the test partition only.

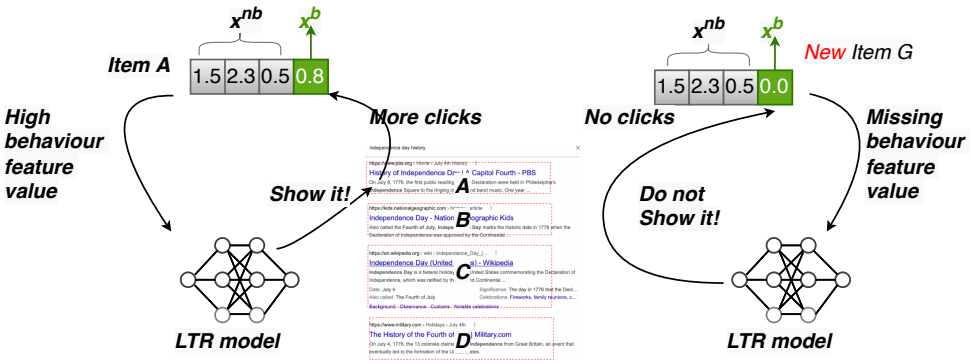


Figure 3.1: Toy example to show the exploitation bias. Old item A and new item G have the same non-behavior (x^{nb}) features, while item G's behavior features (x^{b}) is 0 as it has not been shown to users before. Item G will be discriminated against if the LTR model over-exploits and heavily relies on x^{b}.

Table 3.2: The ranking performance. The best performances within each feature setting are bold. $*$ and † indicate statistical significance over other models in the same or all feature settings, respectively. Cold-NDCG and Warm-NDCG are the same within the W/o-Behav feature setting since behavior signals are not used in both settings.

Feature settings	Online Algorithms	MQ2007			MSLR-10k			MSLR-30k		
		Cold-NDCG	Warm-NDCG	Cum-NDCG	Cold-NDCG	Warm-NDCG	Cum-NDCG	Cold-NDCG	Warm-NDCG	Cum-NDCG
W/o-Behav	BM25	0.474	0.474	94.38	0.449	0.449	90.39	0.451	0.451	89.98
	DBGD	0.557	0.557	110.6	0.488	0.488	98.66	0.498	0.498	99.40
	MGD	0.562	0.562	110.9	0.473	0.473	95.72	0.502	0.502	101.3
	PDGD	**0.599**	**0.599**	115.0	**0.525**	**0.525**	**105.2**	**0.525**	**0.525**	**106.0**
	CFTopK	0.591	0.591	**116.7**	0.510	0.510	102.9	0.506	0.506	101.6
	CFRandomK	0.589	0.589	87.69	0.509	0.509	79.59	0.514	0.514	78.63
	CFEpsilon	0.589	0.589	94.39	0.510	0.510	84.75	0.518	0.518	83.88
W/-Behav (Concate)	DBGD	0.514	0.729	144.7	0.451	0.571	114.3	0.462	0.607	118.8
	MGD	0.523	0.725	142.1	0.461	0.558	109.0	0.444	0.595	122.5
	PDGD	0.574	0.745	147.6	0.466	0.591	117.9	0.480	0.584	116.4
	CFTopK	0.385	0.579	113.5	0.369	0.489	97.53	0.366	0.491	97.65
	CFRandomK	0.377	0.771	87.11	0.403	0.596	79.82	0.404	0.603	78.91
	CFEpsilon	0.387	0.789	143.8	0.355	0.683	116.8	0.354	0.686	116.8
	PRIOR	**0.597**	0.791	158.7	0.507	0.554	110.3	0.503	0.557	111.4
W/-Behav (non-Concate)	UCBRank	0.593	0.799	158.9	**0.514**	0.703	140.1	0.509	0.703	140.5
	EBRank(ours)	0.596	**0.849***†	**171.3***†	0.513	**0.762***†	**151.6***†	**0.513**	**0.762***†	**152.0***†

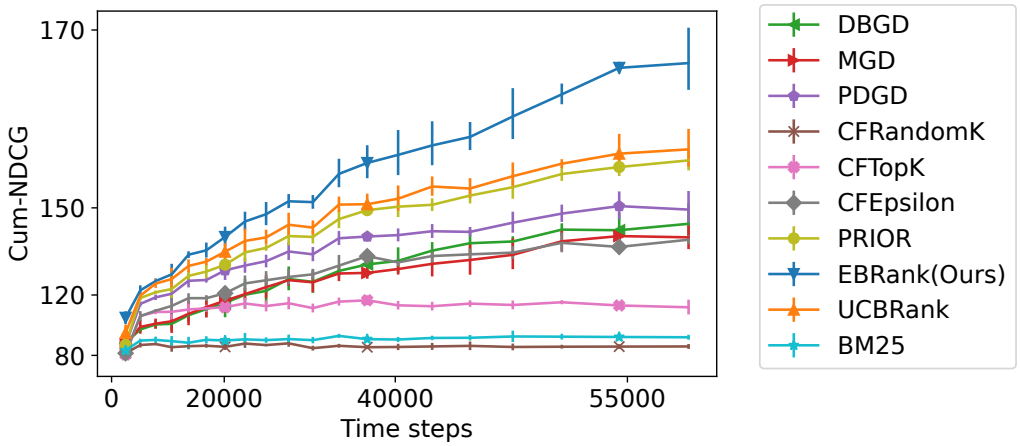


Figure 3.2: Ranking performance (MQ2007, W/-Behav setting).

3.3.2 Result

In this section, we will describe the results of our experiments.

3.3.2.1 How does our method compare with baselines?

In Table 3.2, EBRank significantly outperforms all other methods and feature setting combinations on Warm-NDCG and Cum-NDCG, while its Cold-NDCG is among the best. The discussion in the following sections will give more insights into EBRank's supremacy.

3.3.2.2 Will historical user behavior help ranking algorithms achieve better ranking quality?

As shown in Table 3.2, not all algorithms can benefit from incorporating behavior signals. Particularly, in Table 3.2, we indeed see that both W/-Behav(Concate) and W/-Behav(Non-Concate) feature settings help to boost most ranking algorithms to have better Warm-NDCG and Cum-NDCG. However, such boosting on Warm-NDCG and Cum-NDCG does not apply to all ranking algorithms, and there exist two exceptions. The first one is a trivial exception regarding CFRandomK. CFRandomK always randomly ranks items and shows them to users, so its online performance, i.e., Cum-NDCG, can not be boosted. The second one is a **non-trivial exception** regarding CFTopK. Incorporating user behaviors even makes CFTopK degenerate on Warm-NDCG and Cum-NDCG for all three datasets. Besides CFTopK's degeneration on Warm-NDCG and Cum-NDCG, W/-Behav(Concate) feature setting even causes all ranking algorithms (except PRIOR) to experience a significant drop in Cold-NDCG, when compared to the W/o-Behav feature

setting. Compared to other algorithms, UCBRank and our algorithm EBRank can benefit from behavior signals to have better Warm-NDCG and Cum-NDCG while avoiding drop in Cold-NDCG. In Table 3.2, PRIOR also avoids drop in Cold-NDCG but Prior is not as effective as UCBRank and EBRank in boosting Warm-NDCG and Cum-NDCG with user behavior. Besides the ranking performance in Table 3.2, we additionally show ranking performance as time steps increase in Figure 3.2. As shown in Figure 3.2, EBRank consistently outperforms baselines.

3.3.2.3 How do ranking algorithms suffer from exploitation bias?

In the above section, we found that there exists some exceptions, i.e., degeneration of ranking quality when using the W/-Behav(Concate) feature setting. In this section, we dig into the learned models to investigate the reason for the degeneration. To investigate the learned models, we output behavior and non-behavior features' exploitation ratio for each algorithm in Table 3.3, where the j^{th} feature's exploitation ratio is defined as

$$Ratio_j = \frac{|w_j|}{\sum_i |w_i|} \tag{3.28}$$

w_j is the learned j^{th} weight of a learned linear model. In Table 3.3, the behavior feature's exploitation ratio is significantly greater than the non-behavior features' ratio for all ranking algorithms, which makes behavior features overwhelm other non-behavior features, dominate the output, and discriminate cold-start items. The overwhelming effect will cause exploitation bias. Although the analysis is based on linear models, we believe non-linear models will make exploitation bias even more severe since non-linear models are even better at over-fitting the behavior features.

The exploitation bias is the reason for the ranking quality degeneration mentioned in the last Section (see Sec. 3.3.2.2). Specifically, CFTopK's degeneration in Warm-NDCG and Cum-NDCG is because it fully trusts the exploitation biased ranking model without any exploration, which can discriminate against new items of high quality, making them hard to be discovered. Besides, the drop of Cold-NDCG under the W/-Behav(Concate) feature setting is because the exploitation-biased ranking model can not give an effective ranklist when behavior features are not available in the cold evaluation setting. Although behavior features also dominate in PRIOR, missing behavior feature is filled out by a behavior feature prediction model, which is the reason why its Cold-NDCG does not

Table 3.3: Features' exploitation ratio (Eq. 3.28) in the learnt linear model on dataset MQ2007. x^b Ratio is the behavior features' exploitation ratio. Max x^{nb} Ratio is the maximum exploitation ratio of non-behavior features. Exploitation ratios for UCBRank, EBRank and BM25 are not included since they do not contain x^b in their linear model.

Algorithms	x^b Ratio	Max x^{nb} Ratio
CFEpsilon	$0.604_{(0.098)}$	$0.071_{(0.038)}$
CFRandomK	$0.498_{(0.067)}$	$0.103_{(0.026)}$
CFTopK	$0.591_{(0.087)}$	$0.098_{(0.033)}$
DBGD	$0.109_{(0.021)}$	$0.063_{(0.019)}$
MGD	$0.110_{(0.026)}$	$0.062_{(0.018)}$
PDGD	$0.623_{(0.037)}$	$0.036_{(0.004)}$
PRIOR	$0.572_{(0.105)}$	$0.089_{(0.043)}$

drop. Compared to those algorithms, EBRank is more robust to exploitation bias since it has comparable ranking performance on Cold-NDCG as algorithms using W/o-Behav and has superior performance on Warm-NDCG and Cum-NDCG. In other words, EBRank can take advantage of historical user behavior signals but is also robust to user behavior missing. Besides, EBRank also significantly outperforms UCBRank in Cum-NDCG and Warm-NDCG.

3.3.2.4 EBRank's robustness to entering probability

To investigate the item entering speed (η) 's influence on the ranking performance, we show the experimental results with different η in Figure 3.3. As shown in Figure 3.3, EBRank consistently significantly outperforms all other algorithms under different item entering speeds.

3.3.2.5 Ablation Study

In this section, we do an ablation study to see whether each part of our EBRank is needed. Due to limited space, we only provide analysis on MQ2007 dataset. As shown in Figure 3.4, EBRank significantly outperforms the version only using the Behavior part or the prior model part to rank. Also, from the ablation study, we observe Bayesian modeling can help a ranking model reach good ranking quality and be robust to exploitation bias. The marginal-certainty-aware exploration additionally helps to discover relevant items, which helps to boost ranking performance in the long term.

3.4 Conclusion

In this work, we propose an empirical Bayes-based uncertainty-aware ranking algorithm EBRank with the aim of overcoming the exploitation bias in LTR. With empirical Bayes modeling and a novel exploration strategy, EBRank aims to effectively overcome the exploitation bias and improve ranking quality. Extensive experiments on public datasets demonstrate that EBRank can achieve significantly better ranking performance than state-of-the-art LTR ranking algorithms. In the future, we plan to extend current work further to consider other types of user behaviors beyond user clicks.

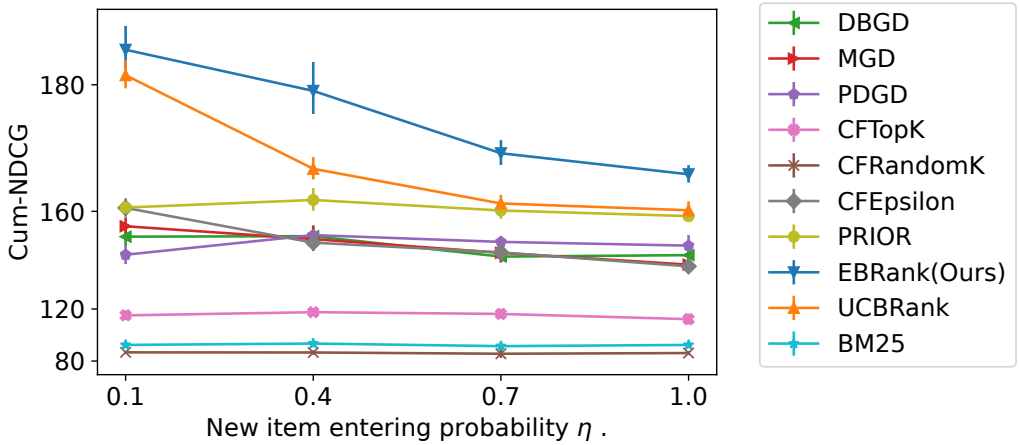


Figure 3.3: Ranking performance with different entering probability η in simulation (see Eq. 3.26) on MQ2007. We only consider using user behavior situations here (except BM25).

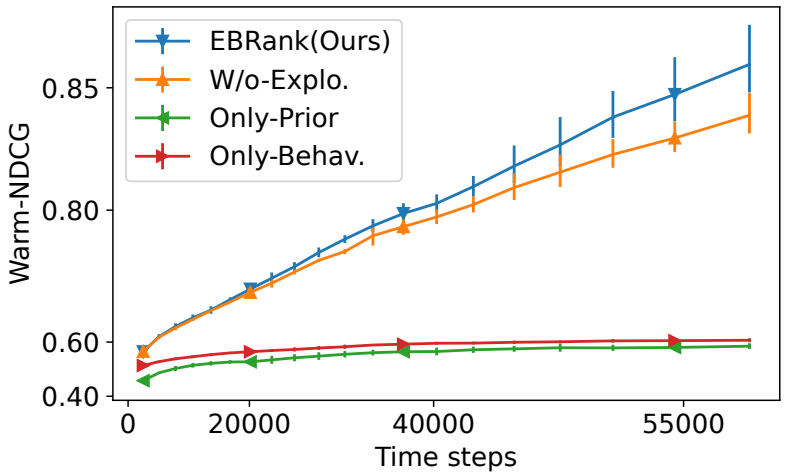


Figure 3.4: Ablation Study of EBRank on dataset MQ2007. W/o-Explo. means excluding $MC(d)$ in Eq. 3.5 when generating ranklists. Only-Prior means only using the prior model part $\frac{\alpha}{\alpha+\beta}$ of $\hat{R}$ (in Eq. 3.16) to rank items. Only-Behav. means only using the behavior part $\frac{C}{n}$ of $\hat{R}$ (in Eq. 3.16) to rank items.

CHAPTER 4

MARGINAL-CERTAINTY-AWARE FAIR RANKING ALGORITHM

Ranking systems are ubiquitous in modern Internet services, including online marketplaces, social media, and search engines. Traditionally, ranking systems only focus on how to get better relevance estimation. When relevance estimation is available, they usually adopt a user-centric optimization strategy where ranked lists are generated by sorting items according to their estimated relevance. However, such user-centric optimization ignores the fact that item providers also draw utility from ranking systems. It has been shown in existing research that such user-centric optimization will cause much unfairness to item providers, followed by unfair opportunities and unfair economic gains for item providers.

To address ranking fairness, many fair ranking methods have been proposed. However, as we show in this dissertation, these methods could be suboptimal as they directly rely on the relevance estimation without being aware of the uncertainty (i.e., variance of the estimated relevance). To address this uncertainty, we propose a novel **M**arginal-**C**ertainty-aware **Fair** algorithm named **MCFair**. MCFair jointly optimizes fairness and user utility, while relevance estimation is constantly updated in an online manner. In MCFair, we first develop a ranking objective that includes uncertainty, fairness, and user utility. Then we directly use the gradient of the ranking objective as the ranking score. We theoretically prove that MCFair based on gradients is optimal for the aforementioned ranking objective. Empirically, we find that on semi-synthesized datasets, MCFair is effective and practical and can deliver superior performance compared to state-of-the-art fair ranking methods. To facilitate reproducibility, we release our code. [1]

[1] https://github.com/Taosheng-ty/WSDM23-MCFair

4.1 Introduction

Advanced ranking techniques have led to improvements in AI-powered information services that significantly changed people's lives. For example, search engines that rank documents according to their utilities to user's queries have helped billions of people better finish their daily work; recommendation systems that rank products/movies/news according to the user's interests have completely changed the way people obtain information every day. Therefore, how to construct and optimize ranking systems is one of the crucial research problems in Information Retrieval (IR) [72].

When considering the quality of result rankings in a ranking system, there are two important criteria: *Ranking Effectiveness* and *Ranking Fairness* [6, 8, 71]. Ranking Effectiveness refers to the ability of a ranking system to effectively put relevant results at the top ranks; by maximizing ranking effectiveness, we can help save users' efforts as they only need to examine the top ranks to satisfy their information needs [16, 86]. However, myopically optimizing ranking effectiveness according to relevance can lead to unfair ranking results. For example, in a hiring website, if a ranking system only considers ranking effectiveness and ranks candidates solely according to relevance, then a small number of top candidates will always be exposed to employers and dominate employers' attention as employers usually only examine the top ranks [16]. In this case, other candidates will be unfairly treated and rarely have the chance to be hired even when they are also highly qualified for the job. Therefore, it is critical to jointly consider ranking fairness and ranking effectiveness. Formally, ranking fairness measures the ranking system's ability to present fairly [6]. In this work, we focus on the important problem of *Exposure Fairness*, as exposure directly influences opinion (e.g., ideological orientation of presented news articles) or economic gain (e.g., revenue from product sales or streaming) for providers of items [71].

Relevance estimation serves as the foundation of the optimization of effectiveness and fairness. Specifically, optimizing effectiveness means putting more relevant items on top ranks, while optimizing fairness means letting items of similar relevance receive similar exposure. To jointly optimize ranking effectiveness and fairness, many fair ranking methods [4, 6, 8] adopt a post-processing setting that assumes that relevance is well estimated prior to the effectiveness-fairness joint optimization. However, such a post-processing setting seldom exists. In a real-world scenario, relevance estimation and ranking opti-

mization are usually dynamically entangled with each other in an online way. Relevance estimation influences how the ranked lists are optimized, and the ranked lists will be later presented to users to collect their feedback, which will, in return, influence relevance estimation. Statistically, relevance estimation usually comes with uncertainty, i.e., variance, and relevance estimations for different items are not equally trustworthy since their uncertainty is usually not the same. Optimizing ranking effectiveness and fairness without considering such differences in uncertainty makes existing post-processing fair methods suboptimal in the online setting. It has come to our notice that although some methods [9, 71] have been proposed to address the online setting, these methods overlook the uncertainty difference in relevance estimation. In this dissertation, we show these uncertainty-oblivious ranking methods are suboptimal in the online setting.

In this dissertation, we propose a **M**arginal-**C**ertainty-aware **Fair** ranking algorithm, or **MCFair** to jointly optimize effectiveness and fairness in an online setting. This algorithm addresses the dynamic nature of online setting where ranking optimization is carried out while the relevance is still being learned. The core of our algorithm is to first formulate a ranking objective that includes effectiveness, fairness, and uncertainty, then take derivatives of the ranking objective with respect to exposure and directly use the gradients as ranking scores. The ranking scores from the gradients automatically include a marginal-certainty-aware exploration strategy to deal with the uncertainty in relevance estimation. We theoretically prove that the ranking scores, i.e., gradients, are optimal for the ranking objective. In addition to the theoretical justification, we provide empirical results with two real-world datasets under both the post-processing setting and the online setting. We find that MCFair outperforms existing state-of-the-art methods significantly. Furthermore, MCFair is efficient, robust, and easy to implement.

4.2 Proposed Method

The challenge of optimizing fairness and effectiveness lies in the fact that the ranking optimization is being carried out based on relevance estimation while relevance estimation is still being learned. Statistically, an estimation such as relevance estimation mostly contains uncertainty, i.e., variance, and possibly some bias, which can make the optimization of fairness and effectiveness suboptimal. For example, an item with under-estimated

relevance and high uncertainty might never get presented to users since both optimizing fairness and optimizing effectiveness will rank irrelevant items to the lower positions of the ranked lists. No presentation will make this item hard to collect feedback to effectively update its relevance estimation. However, this item might be actually relevant and we will know its relevance if more user feedback is collected to reduce the uncertainty in its relevance estimation. Although many existing methods can give an unbiased estimation of relevance [13, 14], the uncertainty (i.e., variance) in this estimation might still make the relevance estimation unreliable, which will make them deliver suboptimal ranking results. To address uncertainty, we propose a Marginal-Certainty-aware Fair ranking algorithm called MCFair, which jointly optimizes effectiveness and fairness.

4.2.1 Gradient-based Optimization Framework

In this section, we introduce a gradient-based ranking optimization framework. As shown in Figure 2.1, at time step t, one user issues a query q and the objective of the framework is to find the optimum ranked list π_t that jointly maximizes effectiveness and fairness:

$$\max_{\pi_t} \quad Obj(q,t) = eff(q,t) + \alpha fair(q,t) \tag{4.1}$$

where α is the coefficient to balance the two utility. According to Eq. 2.5 and Eq. 2.7, we can reorganize the effectiveness as:

$$eff(q,t) = cNDCG@k_s = \frac{1}{DCG@k_s(\pi^*)} \sum_{\tau=1}^{t} \gamma^{t-\tau} DCG@k_s(\pi_\tau) \tag{4.2}$$

In this dissertation, we will adopt k_s (defined in Eq. 2.14) as the cutoff for ranking effectiveness optimization unless explicitly specified. Besides the cutoff, we will set $\gamma = 1$ in Eq. 4.2 for simplicity, and we will relax it to all γ within $[0,1]$ in later discussion. By ignoring the constant $DCG@k_s(\pi^*)$, we can get the *eff* as:

$$
\begin{aligned}
eff(q,t) &= cNDCG@k_s \\
&= \sum_{\tau=1}^{t} DCG@k_s(\pi_\tau) \\
&= \sum_{i=1}^{t}\sum_{j=1}^{k} p_j R(\pi_i[j]) \\
&= \sum_{d\in D(q)} \Big(\sum_{i=1}^{t}\sum_{j=1}^{k} p_j R(d)\mathbb{1}_{\pi_i[j]==d}\Big) \\
&= \sum_{d\in D(q)} R(d)\Big(\sum_{i=1}^{t}\sum_{j=1}^{k} p_j \mathbb{1}_{\pi_i[j]==d}\Big) \\
&= \sum_{d\in D(q)} R(d)E(d)
\end{aligned}
\tag{4.3}
$$

where p_j is the examination probability of j^{th} rank and $\pi_i[j]$ indicates the j^{th} item in user i's ranked list π_i. $E(d)$ is the cumulative exposure defined Eq. 2.10. To maximize $eff(q,t)$, we should allocate more exposure $E(d)$ to items with greater relevance $R(d)$.

At time step t, the objective defined in Eq. 4.1 is equivalent to finding π_t to maximize the marginal objective, denoted as $\Delta Obj(q,t)$:

$$
\begin{aligned}
\max_{\pi_t} Obj(q,t) &\equiv \max_{\pi_t} \quad Obj(q,t) - Obj(q,t-1) \\
&= \max_{\pi_t} \quad \Delta Obj(q,t) \\
&= \max_{\pi_t} \quad \Delta eff(q,t) + \alpha \Delta fair(q,t)
\end{aligned}
\tag{4.4}
$$

where $\Delta Obj(q,t)$ is the increment of objective at time step t. $\equiv$ means equivalence. The equivalence is due to the fact that the ranked list π_t happens time step t and does not change $Obj(t-1)$. To optimize $\Delta Obj(q,t)$, we take the first order approximation of the above marginal objective $\Delta Obj(q,t)$ by considering marginal exposure ΔE's influence:

$$
\begin{aligned}
\Delta Obj(q,t) &\approx \sum_{d\in D(q)} \frac{\partial eff}{\partial E(d)}\Delta E(d) + \alpha \sum_{d\in D(q)} \frac{\partial fair}{\partial E(d)}\Delta E(d) \\
&= \sum_{d\in D(q)} R(d)\Delta E(d) \\
&+\alpha \sum_{d\in D(q)} \underbrace{\frac{4}{n(n-1)}\Big(R(d)\sum_{l} E(l)R(l) - E(d)\sum_{h} R^2(h)\Big)}_{\mathrm{B(d)}}\Delta E(d) \\
&= \sum_{d\in D(q)} \underbrace{(R(d)+\alpha B(d))}_{g(d)}\Delta E(d) \\
&= \langle \overrightarrow{g}\ ,\ \Delta\overrightarrow{E} \rangle
\end{aligned}
\tag{4.5}
$$

where $\overrightarrow{g}$ is $[g(d)\ \forall d \in D(q)]$, the vector form of gradients, $\langle , \rangle$ denotes dot product. The marginal objective at time step t can be approximated by the dot product of gradient $\overrightarrow{g}$ and marginal exposure $\Delta\overrightarrow{E}$ at time step t, i.e. $\langle \overrightarrow{g} , \Delta\overrightarrow{E} \rangle$. Actually, the gradient of effectiveness is $R(d)$, regardless of whether $\gamma = 1$ or $0 \leq \gamma \leq 1$, since γ only affect how we weight the historical DCGs (see Eq. 4.2) in effectiveness and historical DCGs will not affect the current DCG at time step t, i.e., the marginal effectiveness. So, the above derivation still holds when $0 \leq \gamma \leq 1$. The marginal exposure $\Delta\overrightarrow{E}$ is the exposure each item will get at time step t, i.e., p_j in Eq. 4.3. Since $p_j \in (0,1)$ is relatively small, the objective's first-order approximation should approximate $\Delta Obj(q,t)$ well. Furthermore, we can reformulate $\Delta Obj(q,t)$ as:

$$\Delta Obj(q,t) \approx \sum_{d \in D(q)} g(d)\Delta E(d) = \sum_{k=1}^{|D(q)|} g(\pi_t[k])p_k \tag{4.6}$$

where $\pi_t[k]$ is the k^{th} item in the ranklist π_t. To maximize the above $\Delta Obj(q,t)$, we first introduce the Rearrangement Inequality (refer to Section 10.2, Theorem 368 in [89]).

Lemma 1. *Given* $0 \leq x_1 \leq x_2 \leq ... \leq x_n$ *and* $y_1 \leq y_2 \leq ... \leq y_n$*, we have:*

$$\sum_{i=1}^{n} x_{\sigma(i)} y_i \leq \sum_{i=1}^{n} x_i y_i \tag{4.7}$$

where σ *can be any possible permutation.*

According to the above rearrangement inequality, we should let items with greater gradient $g(d)$ get greater examination probability p_k in order to maximize $\Delta Obj(q,t)$. By assuming that p_k drops as rank k increases, **we can maximize $\Delta Obj(q,t)$ by generating a ranked list π_t that arranges items according to their gradients g in descending order**:

$$\pi_t = \arg_{topk}\Big(g(d)|\forall d \in D(q)\Big) \tag{4.8}$$

where k is the length of ranked list π_t.

Aside from optimization, in this dissertation, we also reveal an interesting relation between effectiveness and fairness. When fairness constraint is strictly satisfied and unfairness is reduced to zero, $cNDCG@k_s$ is actually fixed. Then, according to Eq. 2.11, we can reduce the unfairness to zero when items get exposure as:

$$\bar{E}(d) = \frac{R(d)}{\sum_{h \in D} R(h)} E_{sum} \quad \forall d \in D(q) \tag{4.9}$$

where the total exposure $E_{sum} = \sum_{d \in D} E(d)$. By setting $E(d)$ to $\bar{E}(d)$ in Eq. 4.3, we can get the $cNDCG@k_s$ as:

$$\begin{aligned}\overline{cNDCG}@k_s &= \sum_{d\in D(q)} R(d)E(d) \\ &= \frac{\sum_{d\in D(q)} R^2(d)}{\sum_{h\in D} R(h)} E_{sum}\end{aligned} \tag{4.10}$$

where we still assume $\gamma = 1$ and ignore the normalization. From the above derivation, we could know that $cNDCG@k_s$ is fixed as long as the fairness constraint is strictly satisfied, no matter which algorithm we use to reach fairness. Although $cNDCG@k_s$ is fixed, it still leaves us much freedom to improve the top ranks' effectiveness $cNDCG@k_c$ ($k_c < k_s$) as well as to improve effectiveness when some tolerance of fairness is allowed.

4.2.2 Uncertainty-aware Ranking Optimization

In this section, we will extend the above gradient-based ranking optimization framework to be uncertainty-aware. In the above ranking optimization, we optimize ranking in **a post-processing setting** where relevance R is already well-estimated prior to ranking optimization, and we can calculate g in Eq. 4.5 as the ranking score based on the pre-estimated relevance. Such an assumption means we optimize ranking in a post-processing manner. However, in real-world applications, relevance estimation and ranking optimization are often entangled in **an online setting**, where ranking optimization takes place while relevance estimation is still being learned. The online setting brings us a problem relevance estimation is often not perfect and contains uncertainty (variance). In this section, we focus on the scenario where relevance is estimated online with uncertainty.

To analyze uncertainty, we first accumulate the uncertainty of all candidate items:

$$\hat{uncert}(q,t) = \sum_{d\in D(q)} Variance(\hat{R}(d)) \tag{4.11}$$

where $\hat{R}(d)$ is item d's relevance estimation[2] and $Variance(\hat{R}(d))$ is the variance of $\hat{R}(d)$; we leave co-variance to future work. As for how to estimate relevance, we will discuss it in **S**4.2.3. Being uncertainty-aware, we formulate the ranking objective as:

$$\max_{\pi_t} \quad Obj(q,t) = \hat{eff}(q,t) + \alpha \hat{fair}(q,t) - \beta \hat{uncert}(q,t) \tag{4.12}$$

where $\hat{eff}$ and $\hat{fair}$ are the estimated effectiveness and fairness, calculated by substituting R with $\hat{R}$ in Eq. 4.3 and Eq. 2.11.

Although our goal is to jointly maximize effectiveness and fairness, we still include the

[2]Notation with a ˆ is used to denote something is an estimation.

negative $\hat{uncert}$ as a third goal to decrease the uncertainty when optimizing π_t. We follow the assumption that decreasing uncertainty to get a more certain relevance estimation for candidate items can help better optimize effectiveness and fairness, which is later verified by our experimental results. To optimize the above ranking objective, we follow Eq. 4.4 and Eq. 4.5 to take the first order approximation of the marginal objective by considering marginal exposure $\Delta E(d)$,

$$\begin{aligned}\Delta Obj(q,t) &\approx \sum_{d\in D(q)} \left(\frac{\partial \hat{eff}}{\partial E(d)} + \alpha \frac{\partial \hat{fair}}{\partial E(d)} + \beta \frac{-\partial \hat{uncert}}{\partial E(d)} \right) \Delta E(d) \\ &= \sum_{d\in D(q)} \underbrace{\left(\hat{R}(d) + \alpha \hat{B}(d) + \beta \hat{MC}(d) \right)}_{\hat{ug}(d)} \Delta E(d) \\ &= \langle \overrightarrow{\hat{ug}} \, , \, \Delta \overrightarrow{E} \rangle \end{aligned} \tag{4.13}$$

where $\hat{MC}$ denotes Marginal Certainty. Recall that in Eq. 4.8, directly using $\hat{ug}(d)$ as ranking scores to generate the ranklist π_t can help optimize the ranking objective. By optimizing such ranking objective, we automatically get a marginal-certainty-aware exploration strategy. With this strategy, items bringing greater marginal certainty $\hat{MC}$ will be boosted in $\hat{ug}$ to increase their exposure. We refer to the marginal certainty based fairness optimization method as **MCFair**. We also notice a recent related work UCBRank [14], which directly boosts items' ranking scores with uncertainty for exploration. Our marginal-certainty-aware exploration strategy is different from UCBRank, as we consider marginal (un)certainty instead of the uncertainty itself. And we believe that marginal (un)certainty is more effective in terms of exploration. For example, if there exit some items of high uncertainty and such uncertainty cannot be reduced with more user interaction, i.e., low marginal certainty, we shouldn't boost their scores because boosting them cannot reduce the uncertainty of the relevance estimation. Therefore we adopt marginal (un)certainty because we think it could better guide the exploration than the uncertainty itself.

4.2.3 Unbiased Relevance Estimator

The aforementioned gradient-based ranking optimization framework does not depend on the specific choice of relevance estimator. Here we introduce unbiased relevance estimator we adopt in this work. As user feedback could be biased relevance indicator, we follow previous works [13, 14] and use an unbiased estimator of the relevance:

$$\hat{R}(d) = \frac{cumC(d)}{E(d)} \tag{4.14}$$

where $cumC(d)$ is the cumulative clicks computed by:

$$cumC(d) = \sum_{i=1}^{t}\sum_{j=1}^{k} c_{i,j}\mathbb{1}_{\pi_i[j]==d} \tag{4.15}$$

The proof of unbiasedness can be found in [13, 14]. And we can also compute $\hat{R}(d)$'s variance by:

$$Variance[\hat{R}(d)] = \frac{\sum_{i=1}^{t}\sum_{j=1}^{k} Var[c_{i,j}]\mathbb{1}_{\pi_i[j]==d}}{E(d)^2} \tag{4.16a}$$

$$= \frac{\sum_{i=1}^{t}\sum_{j=1}^{k} (\mathbb{E}_c[c_{i,j}^2] - \mathbb{E}_c^2[c_{i,j}])\mathbb{1}_{\pi_i[j]==d}}{E(d)^2} \tag{4.16b}$$

$$= \frac{\sum_{i=1}^{t}\sum_{j=1}^{k} (p(c_{i,j}=1) - p(c_{i,j}=1)^2)\mathbb{1}_{\pi_i[j]==d}}{E(d)^2} \tag{4.16c}$$

$$= \frac{\sum_{i=1}^{t}\sum_{j=1}^{k} (p_j R - p_j^2 R^2)\mathbb{1}_{\pi_i[j]==d}}{E(d)^2} \tag{4.16d}$$

$$< \frac{\sum_{i=1}^{t}\sum_{j=1}^{k} p_j R\mathbb{1}_{\pi_i[j]==d}}{E(d)^2} \tag{4.16e}$$

$$= \frac{RE(d)}{E(d)^2} \tag{4.16f}$$

$$< \frac{1}{E(d)} \tag{4.16g}$$

For simplicity, we use above upper bound as the variance. In Eq. 4.16a, we treat $\hat{R}(d)$ as a linear combination of $c_{i,j}$ to get the variance. In Eq. 4.16c, $c_{i,j} = c_{i,j}^2$ and $\mathbb{E}_c[c_{i,j}] = p(c_{i,j} = 1)$ since $c_{i,j}$ is binary random variable. With above variance, we could get the $\hat{MC}(d)$ as:

$$\hat{MC}(d) = \frac{1}{E(d)^2} \tag{4.17}$$

By substituting $\hat{MC}(d)$ to Eq. 4.13, we will get ranking score $\hat{ug}(d)$. The ranked list is generated by $\pi_t = \arg_{topk}\left(\hat{ug}(d)|\forall d \in D(q)\right)$. In this dissertation, we only use non-parameterized relevance estimator in Eq. 4.14. We leave the analysis of parameterized relevance estimators and its marginal uncertainty to future work.

4.3 Experiments

4.3.1 Experimental Setup

To evaluate our methods, we will conduct semi-synthetic experiments. We cover the experimental settings in this section.

4.3.1.1 Datasets

In the experiment, we use two publicly available datasets, i.e., MQ2008 [90] and Istella-S [91]. MQ2008 has three-level relevance judgments (from 0 to 2), and Istella-S has five-level relevance judgments (from 0 to 4). MQ2008 has about 800 queries and about 20 candidate documents for each query. Istella-S has about 33018 queries and about 103 candidate documents for each query. Queries in both datasets are divided into training, validation, and test partitions.

4.3.1.2 Baselines

To evaluate the proposed method, we compare the following baselines,

- **TopK**. Sort items according to relevance $\hat{R}(d)$, i.e., the first part of $\hat{ug}(d)$ in Eq. 4.13.
- **FairK**. Sort items according to the gradient of fairness $\hat{B}(d)$, i.e., the second part of $\hat{ug}(d)$ in Eq. 4.13.
- **ExploreK**. Sort items according to marginal certainty $\hat{MC}(d)$, i.e., the third part of $\hat{ug}(d)$ in Eq. 4.13.
- **FairCo** [71]. Fair ranking algorithm based on a proportional controller. $\alpha \in [0.0, 1000.0]$
- **ILP** [8]. Fair ranking algorithm based on Integer Linear Programming (ILP).$\alpha \in [0.0, 1.0]$
- **LP** [6]. Fair ranking algorithm based on Linear Programming (LP).$\alpha \in [0.0, 1000.0]$
- **MMF** [9]. Similar to FairCo but focus on top ranks fairness. $\alpha \in [0.0, 1.0]$
- **PLFair** [11]. Fair ranking algorithm based on Placket-Luce optimization. $\alpha \in [0.0, 1.0]$
- **MCFair**. Our method. Sort items according to the gradient of fairness $\hat{ug}(d)$ in Eq. 4.13. $\alpha \in [0.0, 1000.0]$

Among the above ranking algorithm, TopK and ExploreK are unfair algorithms, while the others are fair algorithms. Among the fair algorithms, FairK directly uses fairness's gradient to rank items and can be viewed as a reduced and degenerated version of MCFair when MCFair's α is set to a large number. Please note that FairK is also our proposed method which is derived with the gradient-based optimization framework proposed in Sec 4.2.1. Except FairK, all other fair ranking algorithms have trade-off parameters to balance effectiveness and fairness, referred to as α. Given a greater tradeoff parameter α, the fair algorithms including FairCo, ILP, LP and MCFair care more about fairness, i.e.,

less tolerance for unfairness, which usually sacrificing effectiveness. For example, MCFair can increase α in Eq. 4.12 to give a higher weight to fairness during optimization. For different fair algorithms, α may lie in different range. For FairCo, LP, and MCFair, α are within $[0.0, +\infty]$, and we adopt $\alpha \in [0.0, 1000.0]$ in our experiment. For ILP, $\alpha \in [0.0, 1.0]$. In this dissertation, we do not tune and select one α for each baseline when comparing each baseline. The reason is that different ranking applications can have different fairness requirements, and one α for each baseline is not enough for covering different fairness requirements. To make a comprehensive comparison, we compare baselines for all α within each baseline's respective ranges instead of one particular tuned α within its range. Detailed comparison method can be found at Sec 4.3.2.2. Besides, we limit our discussion within the scope of ranking fairness. UCBRank [14] is not chosen as a baseline since it does not address the ranking fairness problem.

During implementation, we notice that methods LP and ILP are highly time-consuming as their decision variable quadratically increases with the number of candidates. Considering the time cost ((see Table. 4.1)), we filtered out queries with more than 20 documents for MQ2008 and we didn't evaluate ILP and LP on the larger dataset, i.e., the Istella-S dataset.

4.3.1.3 Ranking Service Simulation

Following the workflow in Figure 2.1, at each time step, a simulated user will issue a query by randomly sampling a query from the training, validation, or test partition. Then a ranking algorithm will construct a ranklist π of candidate items and present it to the simulated user. To simulate user's click on the ranklist π, we need to simulate the relevance and examination (details in **S**2.3). Following [74], the relevance probability of each document-query pair (d,q) is simulated according to its relevance judgements y as $P(r = 1|d,q,\pi) = \epsilon + (1-\epsilon)\frac{2^y-1}{2^{y_{max}}-1}$ where y_{max} is the maximum value of relevance judgement y. y_{max} can be 2 or 4 depending on the datasets. Aside from relevance, following [13, 71], we also simulate user's examination probability on π as, $P(e = 1|d,\pi) = \frac{1}{\log_2(rank(d|\pi)+1)}$. We only simulate users' examination behavior on top ranks, and we set k_s to 5 throughout the experiments. For simplicity, we follow existing works [9, 13, 14, 71] to assume that users' examination $P(e = 1|d,\pi)$ is known in experiment as many existing works [74, 86, 87, 92]

have been proposed for it. With simulated relevance and examination behavior, we sample and collect clicks with Eq. 2.13.

Aside from the simulated users' behavior, we notice that LP and ILP methods were originally proposed with the assumption that relevance was already well-estimated prior to ranking optimization. However, in most real-world systems, ranking optimization and relevance learning is carried out at the same time. To give a comprehensive comparison of different methods, we will compare two settings. The first setting is the **post-processing setting** where true relevance R is already given. The second one is the **online setting**, where ranking optimization happens while relevance estimation is still being learned. In the post-processing setting, all the ranking methods in Section 4.3.1.2 are based on true relevance R, and we assume true relevance R is known in advance. Our method MCFair will set β as 0. In the online setting, all baselines are based on the relevance estimation $\hat{R}$ in Eq. 4.14 to perform ranking optimization, and we assume true relevance R is not known at all. Our method MCFair will set β to 100 unless otherwise explicitly specified, as 100 works well across all of our experiments. For MQ2008, we simulate 10^4 and 10^5 steps for post-processing and online settings, respectively. The online setting requires more iterations to learn relevance. For Istella-S, we simulate 10^6 and 10^7 steps for post-processing and online settings, respectively.

4.3.1.4 Evaluation

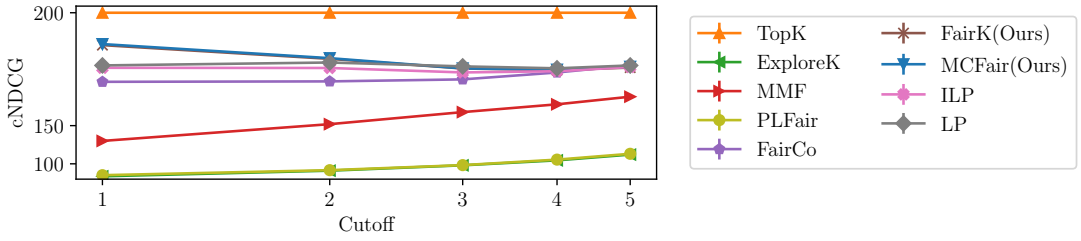
We use the cumulative NDCG (cNDCG) in Eq. 2.7 with $\gamma = 0.995$ to evaluate the effectiveness at different cutoffs, $1 \leq k_c \leq 5$. Aside from effectiveness, unfairness defined in Eq. 2.11 is used for unfairness measurement. We run each experiment five times and report the average evaluation performance on the test partition of each dataset. Note that the test partition was used only for evaluation and not for optimization or validation. As we mentioned in Sec 4.3.1.2, we do not tune and select parameters (e.g., α) in this dissertation . Significant tests are conducted with the Fisher randomization test [88] with $p < 0.05$.

4.3.2 Results in the Post-processing Setting.

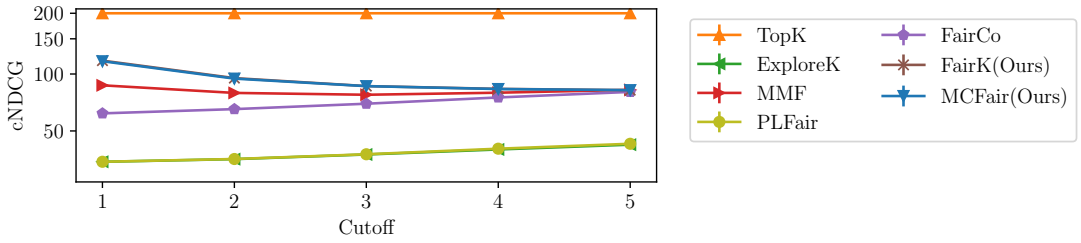
In this section, we show the results in the post-processing setting.

Table 4.1: Unfairness and average time for generating 1k ranklists during simulation in the post-processing setting, where α, if available, are set to the maximum value. The standard deviation is in the parentheses. By setting α to the maximum value, we compare algorithms' fairness capacity to mitigate unfairness. Time costs for ILP and LP on Istella-S are estimated by only running 1k steps instead of the total simulation steps indicated in Sec. 4.3.1.3. Due to the time cost, unfairness performances of ILP and LP on the larger Istella-S dataset are NA in Table.

Methods	Datasets			
	MQ2008	Istella-S	MQ2008	Istella-S
Unfair algorithm	unfairness		Time (sec)	
TopK	$214.4_{(3.884)}$	$19.45_{(0.047)}$	$0.543_{(0.159)}$	$0.572_{(0.131)}$
ExploreK	$261.3_{(7.128)}$	$3.452_{(0.040)}$	$0.577_{(0.192)}$	$0.700_{(0.053)}$
Fair algorithm				
FairCo [71]	$23.69_{(0.740)}$	$0.038_{(0.005)}$	$0.691_{(0.176)}$	$0.607_{(0.015)}$
LP [8]	$25.44_{(0.747)}$	NA	$4.036_{(0.157)}$	≥10 days
ILP [6]	$47.55_{(1.718)}$	NA	$17.24_{(0.479)}$	$1508.9_{(80.83)}$
MMF [9]	$53.36_{(1.334)}$	$0.154_{(0.007)}$	$2.133_{(0.205)}$	$6.876_{(0.475)}$
PLFair [11]	$256.4_{(5.988)}$	$3.700_{(0.062)}$	$4.283_{(0.309)}$	$4.366_{(0.080)}$
FairK(Ours)	$23.16_{(0.742)}$	$0.030_{(0.005)}$	$0.627_{(0.201)}$	$0.770_{(0.099)}$
MCFair(Ours)	$\mathbf{22.68}_{(0.735)}$	$\mathbf{0.029}_{(0.005)}$	$0.631_{(0.195)}$	$0.645_{(0.011)}$



(a) MQ2008 dataset

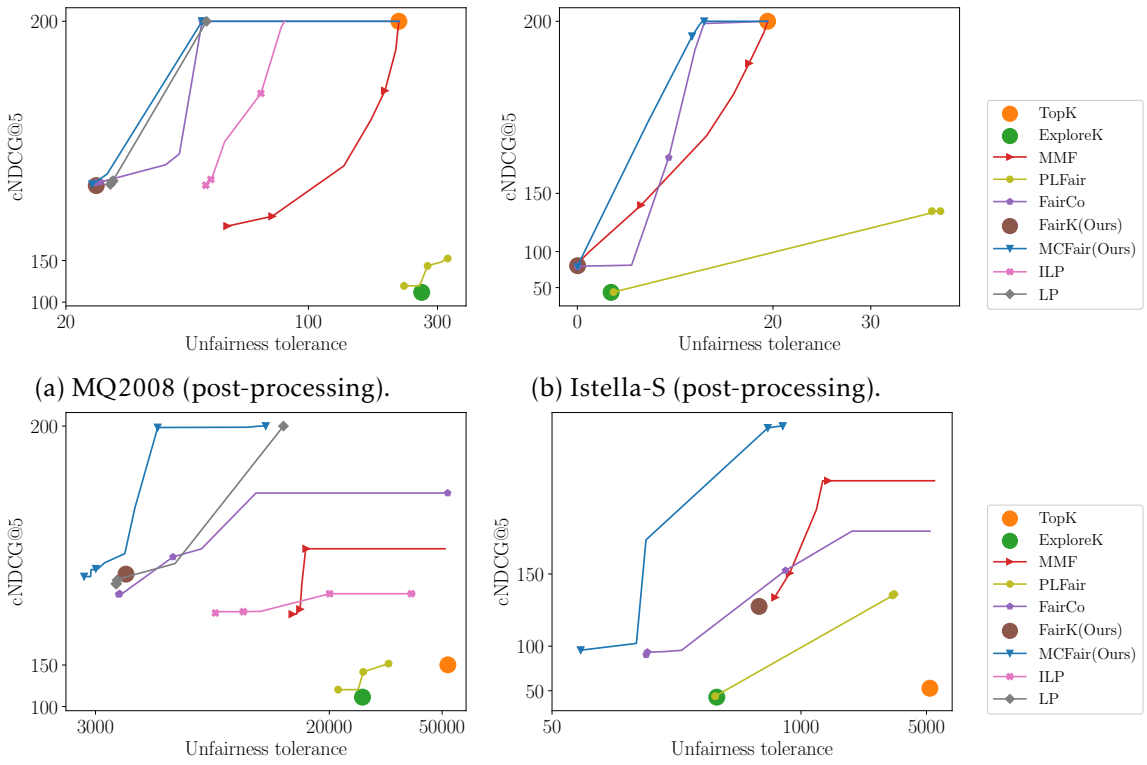


(b) Istella-S dataset

Figure 4.1: cNDCG@$cutoff$ in the post-processing setting. FairK and MCFair overlap. PLFair and ExploreK overlap.

4.3.2.1 Can MCFair reach fairness in the post-processing setting?

In Table 4.1, we compare different methods' capacity to reach fairness, where we prioritize fairness by setting α, if available, as the its maximum value. As shown in Table 4.1,



(a) MQ2008 (post-processing). (b) Istella-S (post-processing).

(c) MQ2008 (online). (d) Istella-S (online).

Figure 4.2: Effectiveness vs. unfairness tolerance in the post-processing setting (a,b) and the online setting (c,d). Given the same unfairness, the higher curves or points lie, the better their performances are.

fair ranking algorithms FairCo, LP, MCFair and FairK can significantly outperform unfair ranking algorithms in term of unfairness mitigation. The success of MCFair and FairK validates our assumption that fairness's gradient can be directly used as ranking scores to optimize fairness. Besides, ILP and MMF show a slightly inferior fairness capacity and PLFair can not mitigate unfairness. More detail discussion and possible reason for their poor performance is in the next section.

Aside from unfairness, we also show the cNDCG performance of different prefixes in Figure 4.1 where α is set to the maximum for each method. In Table 4.1, unfair ranking method TopK can get the highest cNDCG performance as TopK only cares about effectiveness and sacrifices fairness. ExploreK only explores items and thus does not optimize fairness or effectiveness. All the fair algorithms except MMF and PLFair have very similar cNDCG@5 on both datasets which empirically shows that cNDCG@k_s (here $k_s = 5$) is fixed as we derived in Eq. 4.10. Although we use $\gamma = 1$ for derivation in Eq. 4.10 while using $\gamma = 0.995$ to calculate cNDCG@k_s, similar cNDCG@k_s still holds. Despite

similar cNDCG@5 and fairness, MCFair and FairK still significantly outperform other fair algorithms at the top ranks' effectiveness. And we believe higher performance at top ranks' is more important as users usually pay more attention (i.e., higher examining probability) to them.

Besides fairness capacity and effectiveness, we also empirically compare the time efficiency. In Table 4.1, ILP and LP are really time-consuming, while PLFair and MMF also need more time than other algorithms. While the other algorithms have similar time costs.

4.3.2.2 Can MCFair reach a better balance between fairness and effectiveness?

In the previous sections, we compare algorithms when they only care about fairness. However, such a comparison is not sufficient since different ranking systems may have different fairness and effectiveness requirements. To give a comprehensive comparison of different ranking methods, we compare ranking methods' effectiveness-fairness balance given different fairness requirements.

In Figure 4.2a and Figure 4.2b, we show the balance between fairness and effectiveness in the post-processing setting. To generate the balance curves, we incrementally sample α from the minimum value to the maximum value within α's ranges indicated in Section 4.3.1.2. After sampling, we carry out five independent experiments for each α to get the effectiveness and fairness pair based on the average performance of the five independent experiments. Then we connect different α's effectiveness-fairness pair to form a curve in Figure 4.2. The curves start from the right to the left as α increases, and we care more about fairness. Since TopK, ExploreK and FairK don't have trade-off parameters, each one of them only have one single pair of effectiveness and fairness and their performances are shown as single points in Figure 4.2. Besides, the left bottom part of curves means caring fairness only, which corresponds to the results in Table 4.1 and Figure 4.1. All curves show a tradeoff between effectiveness and fairness, which means that mitigating more unfairness usually sacrifices effectiveness. The reason behind this trade-off is that achieving fairness will bring constraints on optimizing effectiveness. Among all fair methods, our method MCFair significantly outperforms all other fair methods where MCFair reaches the best cNDCG given the same unfairness, i.e., MCFair's curve

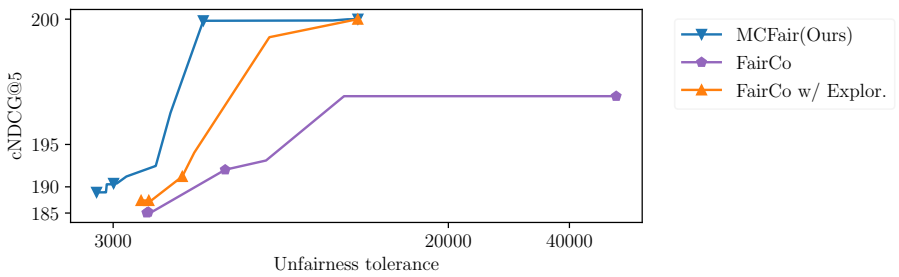


Figure 4.3: FairCo boosted by exploration with marginal certainty based exploration. (MQ2008)

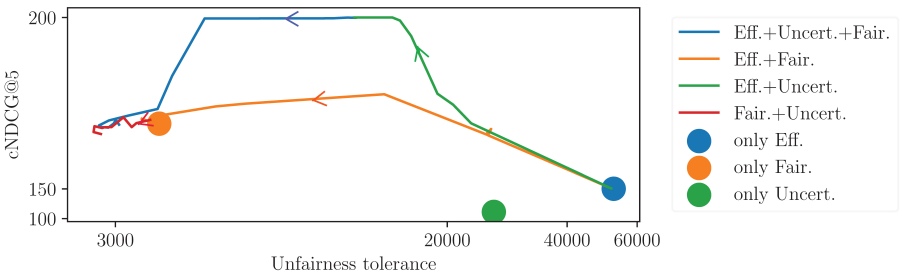


Figure 4.4: Ablation study of MCFair with different combinations of the three parts in Eq. 4.12 on MQ2008. Only considering one part is shown as a scatter point. When considering more than one part, we will get a curve that is generated by increasing the weight of the last part. For example, for Eff.+Uncert.+Fair. and Eff.+Fair., we increase the weights of Fair. i.e., the fairness part when optimizing a ranked list. For Eff.+Uncert. and Fair.+Uncert., we increase the weights of uncertainty. Arrows show how curves develop when increasing the weight of the last part.

lies higher. ILP, MMF, and PLFair can also show the tradeoff between effectiveness and fairness, although they have poor fairness capacities (discussed in Sec 4.3.2.1). As for the possible reason, the integer linear programming method used by ILP may not be effective in optimizing fairness. MMF actually follows a slightly different definition of fairness which require fairness at any cutoff should be fair. As for PLFair, PLFair tries to learn the ranking score that optimizes fairness based on the feature representation (the original setting in [11]). However, the feature representation is originally designed for relevance which makes PLFair suboptimal.

4.3.3 Results in the Online Setting.

In this section, we analyze the results in the online setting.

4.3.3.1 Can MCFair work in the online setting?

To study this problem, we show the balance between the effectiveness and fairness of the online setting in Fig. 4.2c and Fig. 4.2d. Similar to the post-processing setting, our method MCFair still outperforms all other baselines in the online setting. Since most of the results are similar to the post-processing setting in **S**4.3.2.2, we only discuss the difference. In Fig. 4.2c and Fig. 4.2d, TopK cannot reach the highest effectiveness and FairK also can not reach the lowest unfairness in the online setting, which is different from the post-processing setting. We think the reason for the difference is that they naively trust the uncertain relevance estimation, which makes effectiveness optimization and fairness optimization fail (more discussion in **S**4.3.3.3).

4.3.3.2 Can marginal certainty help boost existing fair methods' performance?

In this section, we investigate whether the marginal certainty-based exploration can boost existing fair methods. We mainly focus on how to boost FairCo and leave how to boost other fair methods for future study. Specifically, we directly add $\hat{MC}(d)$ (see Eq. 4.17), the marginal certainty, to FairCo's ranking score, referred to as *FairCo w/ Explor.* As shown in Figure 4.3, FairCo w/ Explor. outperforms FairCo as it reaches better cNDCG given the same unfairness. FairCo w/ Explor.'s better performance shows marginal certainty can effectively boost FairCo. Our method MCFair still significantly outperforms FairCo w/ Explor.

4.3.3.3 Ablation study.

In this section, we conduct an ablation study to evaluate the significance of each part of MCFair. Since the ranking score of MCFair $\hat{ufg}(d)$ in Eq. 4.13 has three parts, corresponding to effectiveness, fairness, and uncertainty respectively, there are a total of seven ($\binom{3}{1} + \binom{3}{2} + \binom{3}{3} = 7$) combinations that need to be evaluated. The ablation results of each combination are shown in Figure 4.4. For the ablation results, considering more than two parts are shown as balance curves while considering only one part is shown as single points. In Figure 4.4, there is a very interesting cycle formed by the curves. In the cycle, considering all three parts, i.e., Eff.+Fair.+Uncer., outperform all other combinations since Eff.+Fair.+Uncer. can reach better effectiveness given the same unfairness.

4.4 Conclusions

In this work, we study the critical problem of relevance-fairness balance in online ranking settings. We propose a novel Marginal-Certainty-aware Fair Ranking algorithm named MCFair. MCFair jointly optimizes fairness and user utility while relevance estimation is constantly updated in an online manner. With extensive experiments on semi-synthesized datasets, MCFair shows its superior performance compared to other fair ranking algorithms.

CHAPTER 5

FARA: FUTURE-AWARE RANKING ALGORITHM FOR FAIRNESS OPTIMIZATION

Ranking systems are the key components of modern Information Retrieval (IR) applications, such as search engines and recommender systems. Besides the ranking relevance to users, the exposure fairness to item providers has also been considered an important factor in ranking optimization. Many fair ranking algorithms have been proposed to jointly optimize both ranking relevance and fairness. However, we find that most existing fair ranking methods adopt greedy algorithms that only optimize rankings for the next immediate session or request. As shown in this dissertation, such a myopic paradigm could limit the upper bound of ranking optimization and lead to suboptimal performance in the long term.

To this end, we propose **FARA**, a novel **F**uture-**A**ware **R**anking **A**lgorithm for ranking relevance and fairness optimization. Instead of greedily optimizing rankings for the next immediate session, FARA plans ahead by jointly optimizing multiple ranklists together and saving them for future sessions. Specifically, FARA first uses the Taylor expansion of the fairness objective to investigate how future ranklists will influence the overall fairness of the system. Then, based on the analysis of the Taylor expansion, FARA adopts a two-phase optimization algorithm where we first solve an optimal future exposure planning problem and then construct the optimal ranklists according to the optimal future exposure planning. Theoretically, we show that FARA is optimal for ranking relevance and fairness joint optimization. Empirically, our extensive experiments on three semi-synthesized datasets show that FARA is efficient, effective, and can deliver significantly better ranking performance compared to state-of-the-art fair ranking methods. We make our implementation public at https://github.com/Taosheng-ty/QP_fairness/.

5.1 Introduction

Ranking systems are one of the important cornerstones of information retrieval (IR). Existing ranking systems are usually constructed to optimize ranking relevance with the Probability Ranking Principle (PRP) [93] where items of greater likely relevance should be ranked higher. The PRP is a user-centered ranking strategy that helps save users energy and time since users could satisfy their needs with the top-ranked items [16]. However, recent research has shown that, besides users, item providers also draw utility from ranking systems, and the PRP could result in severe unfairness for item providers [6, 8]. Particularly, the PRP always assigns a few top items with high-rank positions. Those top items usually get the majority of exposure while other items rarely get exposure, although other items might still be relevant [1, 94, 95, 96]. The unbalanced exposure leads to unfair opportunities and unfair economic gains for item providers. Such unfairness will eventually force unfairly treated providers to leave the system, and fewer options will be left for users [4]. Therefore, IR researchers have argued that ranking relevance and fairness are both important for modern ranking systems [6, 10]. Many fair ranking algorithms have been proposed to optimize both of them jointly [94, 97].

However, existing fair algorithms are mostly greedy algorithms and could only deliver suboptimal ranking performance in the long run. In particular, existing fair ranking algorithms [6, 8, 11, 71, 98] usually behave greedily to sequentially produce the locally optimal ranklist for the next immediate session without being aware of the influence of future sessions[1] The unawareness could lead to unmitigated ranking conflict between relevance and fairness optimization. For example, imagine a case that there are in total 3 items in consideration, item A, item B, and item C, where item A is the most relevant one and item C is the least relevant one. Ranklist $[A, B, C]$ is the ranklist to maximize ranking relevance. We now consider a scenario where item C is severely unfairly treated in history. To optimize exposure fairness, we need to allocate item C more exposure by boosting item C to a higher position. However, If we try to greedily boost item C within the next immediate session, it is highly likely that item C will be boosted to the first rank to get the maximum exposure and the resulting ranklist is $[C, A, B]$. However, Ranklist

[1] In this dissertation, we define a session as a query issued by a user.

$[C, A, B]$ is of poor ranking relevance due to the ranking conflict that the least relevant item (item C) is put on the most important rank (the first rank).

Intuitively, the ranking conflict can be smoothed if we plan ahead and jointly optimize multiple future sessions' ranklists together instead of greedily optimizing the next immediate session. For example, the multiple ranklists after joint optimization can be $[[A, C, B], [A, C, B], ...]$, where item C is smoothly boosted in multiple ranklists and the most relevant item, i.e., item A, is still ranked the highest. Based on the above idea, we propose **FARA**, a novel **F**uture-**A**ware **R**anking **A**lgorithm for relevance and fairness optimization. Briefly, FARA precomputes and jointly optimizes multiple ranklists together and saves them for future use. Particularly, to be able to plan for the future, FARA first uses the Taylor expansion of the fairness objective to investigate how future ranklists will influence fairness. Then, based on the influence, FARA uses a two-phase optimization to jointly optimizes multiple ranklists together for future use. In phase 1, we solve an exposure planning problem and get the optimal future planning for item exposure. In phase 2, we construct the optimal ranklists according to the optimal future planning for item exposure. We prove FARA's optimum in terms of ranking relevance and fairness joint optimization in **S** 5.3. Extensive experiments on three semi-synthesized datasets also demonstrate FARA's effectiveness and efficiency compared to existing fair ranking algorithms (**S** 5.4).

5.2 Proposed Method

Most existing fair algorithms are greedy algorithms, i.e., they sequentially construct the locally optimal ranklist for the next immediate session. Therefore they usually fail to optimize the construction procedure if we expect multiple sessions will come for the same query in the future. To mitigate this gap and reach a global optimal for a query, we propose to (i) **plan ahead and precompute multiple ranklists for future use (S 5.2.1)** and (ii) **jointly optimize those ranklists together to maximize both fairness and ranking relevance (S 5.2.2 & S 5.2.3)**. We hypothesize that jointly optimizing multiple ranklists can construct better ranklists compared to sequentially greedily optimizing one single ranklist at each time step. Such hypothesis is verified by both the theoretical analysis in **S** 5.3 and the empirical results in **S** 5.4.2.

5.2.1 Future-aware Ranking Objective

We first propose a ranking fairness objective to plan and optimize the future ΔT ranklists for a query q. Specifically, when we are at time step $t+1$, the objective is to pre-compute the optimal ranklists $\mathcal{B}^* = [\pi_{t+1}, ..., \pi_{t+\Delta T}]$ that can maximize the marginal fairness $\Delta\hat{fair}(q, t, t+\Delta T)$,

$$\mathcal{B}^* = \underset{\mathcal{B}=[\pi_{t+1},...,\pi_{t+\Delta T}]}{\operatorname{argmax}} \quad \Delta\hat{fair}(q, t, t+\Delta T) \tag{5.1}$$

$\Delta\hat{fair}(q, t, t+\Delta T) = \hat{fair}(q, t+\Delta T) - \hat{fair}(q, t)$, and $\hat{fair}$ is the estimated fairness. $\hat{fair}$ is calculated with Eq. 2.11 by substituting true relevance R with the estimated relevance, denoted as $\hat{R}$, since true relevance is mostly not available during optimization. Here maximizing the marginal fairness $\Delta\hat{fair}(q, t, t+\Delta T)$ is equivalent to maximizing final fairness $\hat{fair}(q, t+\Delta T)$ at time step $t+\Delta T$. The reason for the equivalence is that ranklists $\mathcal{B} = [\pi_{t+1}, ..., \pi_{t+\Delta T}]$ can only influence the marginal fairness from timestep $t+1$ to timestep $t+\Delta T$ rather than fairness before timestep t. Here, fairness evaluation in Eq. 2.11 is a direct objective in our method.

To the best of our knowledge, there is no trivial algorithm to get the optimal ranklists $\mathcal{B}^*$ due to the discontinuity of ranking problem [72]. One example of discontinuity is that increasing an item's ranking score may not change the output ranklist, and fairness stays the same unless the increased score can surpass another item's score, and fairness will experience a sudden change. To alleviate the discontinuity of $\mathcal{B}^* \leftarrow \operatorname{argmax}_B \Delta\hat{fair}$, we propose a novel two-phase solution path by introducing a continuous variable, ΔE,

$$\mathcal{B}^* \xleftarrow[phase2]{\text{Vertical Allocation}} \Delta E^*(d)\ \forall d \in D(q) \xleftarrow[phase1]{\operatorname{argmax}_{\Delta E}} \Delta\hat{fair} \tag{5.2}$$

where $\Delta E(d)$, also referred to as the planning exposure, is the marginal (or incremental) exposure we plan to assign to item d within the next ΔT timesteps. $\Delta E^*(d)$ is the optimal marginal exposure. We found that introducing ΔE helps to effectively maximize $\Delta\hat{fair}$ in phase 1 of our solution. In phase 2, we construct the optimal ranklists $\mathcal{B}^*$ by allocating the optimal exposure ΔE^* to each item with a vertical allocation method (more details are in **S**5.2.3).

To get ΔE^*, we carry out a Taylor series expansion to investigate how future exposure will influence the fairness objective,

$$
\begin{aligned}
\Delta\hat{fair}(q,t,t+\Delta T) &= \sum_{d\in D(q)} \frac{\partial \hat{fair}}{\partial E(d)} \Delta E(d) \\
&+ \frac{1}{2} \sum_{d_x\in D(q)} \sum_{d_y\in D(q)} \frac{\partial^2 \hat{fair}}{\partial E(d_x)\partial E(d_y)} \Delta E(d_x)\Delta E(d_y) \\
&= \sum_{d\in D(q)} \hat{G}(d)\Delta E(d) \\
&- \frac{1}{2} \sum_{d_x\in D(q)} \sum_{d_y\in D(q)} \hat{H}(d_x,d_y)\Delta E(d_x)\Delta E(d_y) \\
&= \overrightarrow{\hat{G}} \cdot \Delta\overrightarrow{E} - \frac{1}{2}\Delta\overrightarrow{E}^T \cdot \hat{H} \cdot \Delta\overrightarrow{E}
\end{aligned}
\tag{5.3}
$$

where $\Delta E(d) = E^{t+\Delta T}(d) - E^t(d)$, the exposure increments. $\hat{G}$ and $\hat{H}$ are the first and the second order derivative, i.e., the gradient vector and the Hessian matrix, respectively,

$$
\begin{aligned}
\hat{G}(d) &= \frac{4}{n(n-1)}\left(\hat{R}(d)\sum_l E^t(l)\hat{R}(l) - E^t(d)\sum_h \hat{R}^2(h)\right) \\
\hat{H}(d_x,d_y) &= \frac{4}{n(n-1)}\left((\sum_{d\in D(q)} \hat{R}^2(d))\mathbb{1}_{x=y} - \hat{R}(d_x)\hat{R}(d_y)\right)
\end{aligned}
\tag{5.4}
$$

By observing Eq. 2.11, we could derive two facts about the above second-order expansion in Eq. 5.3. (i) The above second-order expansion is not an approximation, but equality since (un)fairness in Eq. 2.11 is defined as a polynomial of E with a degree of two, and its derivative of order higher than two is zero. (ii) Since (un)fairness in Eq. 2.11 is defined as a sum of squares, the second order derivative $\hat{H}$ is semi-definite. Being equality, Eq. 5.3 allows us to correctly estimate future fairness given marginal exposure ΔE even when we consider a long-term future (large ΔT and ΔE). Based on the correct future fairness estimation, it is possible to find the optimal marginal exposure planning, denoted as ΔE^*, that can maximize future fairness. Since $\hat{H}$ is semi-definite, Quadratic Programming-based (QP) optimization is valid to find ΔE^*. We give the specific QP problem formulation to find ΔE^* in **S** 5.2.2 and leave constructing optimal ranklists $\mathcal{B}^*$ from ΔE^* in **S** 5.2.3.

5.2.2 *Phase 1*: Future Exposure Planning

When giving the QP problem formulation, we noticed that existing ranking fairness optimization usually considers two settings: (i) **the post-processing setting** [6, 8, 10] where relevance is assumed to be known or well estimated in advance; and (ii) **the online setting** [9, 71] where fairness is optimized while relevance is still being learned. To consider both settings, we first illustrate the QP problem formulation in the post-processing

setting in **S** 5.2.2.1 and then extend it to work in the online setting in **S** 5.2.2.2.

5.2.2.1 The post-processing setting

To get the optimal exposure planning ΔE^*, we propose the following QP formulation with $\Delta E(d)\ \forall d \in D(q)$ as decision variables,

$$\max_{\Delta E} \quad \Delta \hat{fair}(q, t, t + \Delta T) \tag{5.5a}$$

$$s.t. \quad \sum_{d \in D(q)} \Delta E(d) = \sum_{i=1}^{\Delta T} \sum_{j=1}^{k_s} P_j \tag{5.5b}$$

$$\sum_{d \in D(q)} \Delta E(d) \hat{R}(d) \geq (1 - \alpha) \sum_{i=1}^{\Delta T} \sum_{j=1}^{k_s} P_j \hat{R}(d_{order_j}) \tag{5.5c}$$

$$\Delta E(d) \geq 0, \forall d \in D(q) \tag{5.5d}$$

$$\Delta E(d) \leq \sum_{i=1}^{\Delta T} P_1, \forall d \in D(q) \tag{5.5e}$$

where k_s is the length of ranklists, P_j is the examining probability at rank j. Eq.5.5b indicates that the sum of items' marginal exposure should equal the sum of the ΔT ranklists' exposure. In Eq.5.5c, we introduce the *NDCG* constraint, where $\hat{R}(d_{order_j})$ means the j^{th} largest estimated relevance. According to Eq.(2.3&2.5), $\sum_{d \in D(q)} \Delta E(d) \hat{R}(d)$ indicates the *DCG* and $\sum_{i=1}^{\Delta T} \sum_{j=1}^{k_s} P_j \hat{R}(d_{order_j})$ indicates *IDCG*. Then, it is straightforward that $(1 - \alpha)$ indicates the minimum *NDCG* requirement we want to guarantee. In the post-processing setting, relevance is assumed to be given or already well-estimated prior to ranking optimization. Eq.5.5d indicates that an item's marginal exposure $\Delta E(d)$ should be no less than 0. In Eq.5.5e, $\Delta E(d)$ should be more than the accumulation of the first rank's exposure in the ΔT ranklists because an item should be unique in a ranklist and there are ΔT ranklists under consideration. Here we assume that the first rank's exposure is the largest and exposure drops from the top to the lower ranks.

5.2.2.2 The online setting.

In the online setting, ranklists are optimized while relevance is still being learned. How to actively explore items and get more accurate relevance for ranking optimization is critical. Yang et al. [14] show that a more accurate relevance estimation for an item can be achieved by exposing an item more because more exposure leads to more interaction with users. Based on this, we do explorations by setting a minimum exposure requirement for

items and propose the following QP formulation,

$$\max_{\Delta E} \quad \Delta \hat{fair}(q,t,t+\Delta T) - \beta \sum_{d \in D(q)} s(d) \tag{5.6a}$$

$$s.t. \qquad Eq.\ (5.5b, 5.5c, 5.5d, 5.5e) \tag{5.6b}$$

$$s(d) \geq 0, \forall d \in D(q) \tag{5.6c}$$

$$s(d) + \Delta E(d) + E^t(d) \geq E_{min}, \forall d \in D(q) \tag{5.6d}$$

where β indicates the importance of exploration, $s(d)\ \forall d \in D(q)$ are slack variables to encourage exploration. In other words, both $s(d)$ and $\Delta E(d)\ \forall d \in D(q)$ are decision variables in the setting. In Eq. 5.6d, E_{min} is the minimum exposure requirement, $E^t(d)$ is item d's exposure accumulated till time step t, and $\Delta E(d)$ is the marginal exposure we plan to allocate to item d within the next ΔT steps. $s(d)$ can be interpreted as the additional exposure still needed to satisfy the minimum exposure requirement after the next ΔT steps. When $E^t(d) \geq E_{min}$ for item d, i.e., minimum exposure requirement is already satisfied for item d, it is straightforward that $s(d)$ will be 0 and will not contribute to the ranking objective in Eq. 5.6a. When $E^t(d) < E_{min}$, i.e. item d does not meet the minimum exposure requirement, the objective in Eq. 5.6a will try to minimize $s(d)$. In other words, $\Delta E(d)$ will be boosted in order to satisfy the constraint in Eq. 5.6d. With more exposure, item d will be explored more. In this dissertation, we refer to the introduction of s as **Exploration**. And we treat β in Eq. 5.6a and E_{min} as hyper-parameters to control the degree of exploration.

As quadratic programming has been well studied, there are many available existing solvers. In this dissertation, we use quadratic programming library qpsolvers[2] within python to solve Equation 5.5 and Equation 5.6 to get the optimal exposure planning ΔE^*.

5.2.3 *Phase 2*: Ranklists Construction

Following the solution path in Eq. 5.2, the next step is to construct the optimal ranklists $\mathcal{B}^*$ according to ΔE^* as ΔE^* has been solved in Phase 1. Here, we should allocate each item exactly its optimal exposure ΔE^* within $\mathcal{B}^*$. However, we find that the allocation solution of $\mathcal{B}$ is not unique, and they share the same aver-NDCG@k_s (see Theorem 4). Therefore, we additionally aim to find the optimal $\mathcal{B}^*$ that can optimize **all top ranks' effectiveness**,

[2]https://pypi.org/project/qpsolvers/

Algorithm 5.1: Vertical Allocation

Input: The optimal exposure planning ΔE^*, the number of planning sessions to consider ΔT, the ranked list length k_s, relevance estimation $\hat{R}$.;
Initialize: ranking lists $\mathcal{B}^*(i) \leftarrow [\varnothing]$ for $i \in range(\Delta T)$, set $\tilde{E}(d) \leftarrow 0 \quad \forall d \in D$;
for $rnk \in [1, 2, ..., k_s]$ **do**
  for $sess \in [1, 2, ..., \Delta T]$ **do**
    $Set1 \leftarrow \{d | \Delta E^*(d) - \tilde{E}(d) \geq P_{rnk}\}$;
    $Set2 \leftarrow \{d | d \notin \mathcal{B}^*(sess))\}$;
    if $Set1 \cap Set2 = \varnothing$ **then**
      $Candidates \leftarrow Set2$
    else
      $Candidates \leftarrow Set1 \cap Set2$
    $d^* \leftarrow \underset{d \in Candidates}{\text{argmax}} \ \hat{R}(d)$;
    $\mathcal{B}^*(sess).append(d^*)$;
    $\tilde{E}(d^*) \leftarrow \tilde{E}(d^*) + P_{rnk}$;
Output: $\mathcal{B}^*$;

i.e., aver-NDCG@k_c, $\forall k_c \leq k_s$. Optimizing top ranks' effectiveness is important since users usually pay more attention to top ranks.

Inspired by [4], we propose a vertical exposure allocation method in Algorithm 5.1 to construct the optimal $\mathcal{B}^*$ based on ΔE^*. The difference between a vertical allocation and a horizontal allocation is the ranklist construction order. As shown in Fig. 5.1, a horizontal allocation prioritizes earlier ranklists and first fills out all ranks of the i^{th} ranklist π_i before filling out π_{i+1}. However, a vertical allocation prioritizes top ranks and fills out the i^{th} ranked items of all ranklists before filling out any $(i+1)^{th}$ ranked item. Since top ranks are usually more important, our proposed Algorithm 5.1 adopts a vertical allocation to fill out $\mathcal{B}^*$. In our proposed Algorithm 5.1, $\mathcal{B}^*$ is the generated ΔT ranklists and $\mathcal{B}^*(sess, rnk)$ denote the rnk^{th} rank of the $sess^{th}$ ranklist. To fill out $\mathcal{B}^*(sess, rnk)$, we first construct two item sets, items that have not been selected for this session ($d \notin B(sess)$), i.e., $Set1$, and items that still have planned exposure left ($\Delta E(d) - \tilde{E}(d) \geq P_{rnk}$), i.e., $Set2$. Here, examination probability P_{rnk} serves as the margin, and $\tilde{E}(d)$ stores the actual exposure item d receives. If $Set1 \cap Set2 = \varnothing$, it means that planned exposure has been fulfilled and we only need to avoid repeated items in the same session. So we construct item candidates from $Set2$ only. If $Set1 \cap Set2 \neq \varnothing$, we construct item candidates from their intersect, i.e., items that have not been selected for this session but still have planned exposure

Algorithm 5.2: FARA: Future-aware Ranking Algorithm

Input: The number of planning sessions to consider ΔT, fairness-effectiveness tradeoff parameter α. And in the online setting, we need to additionally give the exploration parameters β and E_{min} (see Eq. 5.6) ;
Initialize time step $t \leftarrow 0$, initialize an empty dictionary $\mathbb{B} \leftarrow \{\}$ to store ranked lists, items' exposure $E \leftarrow 0$ and cumulative click $cumC \leftarrow 0$;
while *True* **do**
    $t \leftarrow t + 1$;
    A user issues a query q_t;
    if $q_t \notin \mathbb{B}$ **then**
        $\mathbb{B}[q_t] = []$, i.e., add an empty list
    if $\mathbb{B}[q_t]$ *is empty* **then**
        Get ΔE^* by solving Eq. 5.5 (post-processing) or Eq. 5.6 (online);
        Get $\mathcal{B}^*$ with Algorithm 5.1;
        Randomly shuffle $\mathcal{B}^*$;
        $\mathbb{B}[q_t] \leftarrow \mathcal{B}^*$;
    Pop out a ranking list from $\mathbb{B}[q_t]$ and present it;
    Update cumulative click $cumC$ and items' exposure E;
    Update the relevance estimation via Eq. 5.7;

left. Within $Candidates$, our algorithm selects the most relevant item from the candidate set to fill out $\mathcal{B}^*(sess, rnk)$. Algorithm 5.1 is theoretically justified to accurately allocate exposure ΔE^* (see Theorem 2) and can construct optimal $\mathcal{B}^*$ for aver-NDCG@k_c $\forall k_c \leq k_s$ (see Theorem in Sec.5.3).

Although inspired by the vertical method in [4], the proposed vertical allocation is different from it. Yang et al. [4] focus on a certain share of exposure to be guaranteed and have a complicated 3-step procedure, i.e., allocation, appending, and resorting, which cannot be used to allocate ΔE^* for our problem. The proposed allocation algorithm in this dissertation uses up all ΔE^*, and the allocation procedure is less complicated and more straightforward than those introduced by Yang et al. [4].

5.2.4 FARA: Future-Aware Ranking Algorithm

Combining *Phase 1* and *Phase 2*, we propose a future-aware ranking algorithm for fairness optimization, **FARA**, detailed in Algorithm 5.2. FARA serves users in an online manner where we pre-compute $\mathcal{B}^*$, the next ΔT ranklists, and randomly pop out one ranklist from $\mathcal{B}^*$ when needed. When $\mathcal{B}^*$ is used up and empty, We will re-compute $\mathcal{B}^*$ for future ΔT timesteps. $\mathbb{B}[q]$ is used to store $\mathcal{B}^*$ for query q. Besides, FARA does not depend

on any specific relevance estimation model, therefore, can be seamlessly integrated into most existing ranking applications. In this dissertation, we follow works by [14] to use the following unbiased estimator of relevance,

$$\hat{R}(d) = \frac{cumC^t(d)}{E^t(d)} \tag{5.7}$$

where

$$cumC^t(d) = \sum_{i=1}^{t}\sum_{j=1}^{k_s} C_{i,j}\mathbb{1}_{\pi_i[j]==d} \tag{5.8}$$

is the cumulative clicks[3]. Moreover, it is worth noting that the relevance estimator can be replaced with other relevance estimators as well.

5.3 Theoretical Analysis

Theorem 2. *Algorithm 5.1 can theoretically guarantee* $\Delta E(d) - \tilde{E}(d) \leq P_{k_s}$ *for at least* $|D| - k_s$ *items.*

Proof. Here we discuss the exposure allocation error bounds in Phase 2 of FARA, i.e., $|\tilde{E}(d) - \Delta E(d)|$, in Algorithm 5.1. We noticed that there are two possible scenarios of lines 6-12 in Algorithm 5.1:

Scenario 1: There exists a $(rnk^*, sess^*)$ *pair where* $Set1 \cap Set2 = \varnothing$. If this happens, $(k_s, sess^*)$ will also have $Set1 \cap Set2 = \varnothing$ since the size of $Set1$ and $Set2$ monotonically decrease for lower rank of the same session. As $Set2$ is the set of unselected items for a session, we know that there are at least $|D| - k_s$ items in $Set2$, i.e., $|Set2| \geq |D| - k_s$. If $Set1 \cap Set2 = \varnothing$, those $|D| - k_s$ items are not in $Set1$. In other words, there are at least $|D| - k_s$ items that satisfy $\Delta E(d) - \tilde{E}(d) < P_{k_s}$. Since $\Delta E(d) >> P_{k_s}$ and $|D| >> k_s$ when using FARA, we still claim that Algorithm 5.1 correctly allocates exposure in this scenario.

Scenario 2: $Set1 \cap Set2 \neq \varnothing \; \forall (rnk, sess)$ *pair*. Due to the margin P_{rnk} in line 5 of Algorithm 5.1, $\Delta E(d) \geq \tilde{E}(d) \; \forall d \in D$ should always hold if $Set1 \cap Set2 = \varnothing$ never happens. By considering $\Delta E(d) \geq \tilde{E}(d) \; \forall d \in D$ and the identity $\sum_{d \in D(q)} \Delta E(d) \equiv \sum_{d \in D(q)} \tilde{E}(d)$, we would know that $\Delta E(d) = \tilde{E}(d) \; \forall d \in D$, which means exposures are perfectly allocated according to $\Delta E(d)$.

Combing the two scenarios, the vertical allocation in Algorithm 5.1 can theoretically guarantee $\Delta E(d) - \tilde{E}(d) \leq P_{k_s}$ for at least $|D| - k_s$ items.

---

[3] We skip the proof of unbiasedness for the above estimator and refer interested readers to [14]

□

Theorem 3. *FARA can reach the optimal NDCG with the given exposure planning.*

Proof. Here we provide theoretical proof that vertical allocation, i.e., phase 2, can optimize effectiveness (*aver-NDCG*) when exposure planning ΔE is given. Specifically, maximizing *aver-NDCG@*k_c in Eq. 2.8 is equivalent to

$$\max \sum_{d \in D(q)} R(d) E@k_c(d) \tag{5.9a}$$

$$s.t. \quad \sum_{d \in D(q)} E@k_c(d) = Const. \tag{5.9b}$$

$$0 \leq E@k_c(d) \leq \Delta E(d) \ \ \forall d \in D \tag{5.9c}$$

where normalization is ignored in Eq. 5.9a, the sum of top ranks exposure should be a constant in Eq. 5.9b, and the top ranks exposure should be less than the total exposure planning in Eq. 5.9c. According to Rearrangement Inequality [89], it is straightforward to know that *aver-NDCG@*k_c, i.e., Eq. 5.9a, can be optimized by letting item of greater relevance R get more exposure at top ranks, i.e., greater $E@k_c$. In other words, we should prioritize letting items of greater relevance R fulfill their exposure planning ΔE at top k_c ranks since $E@k_c$ is bounded in $[0, \Delta E]$. By assuming that *aver-NDCG* at higher ranks is more important [93], we should maximize *aver-NDCG@*k_c before maximizing *aver-NDCG@*$(k_c + 1)$, $\forall\, 1 \leq k_c < k_s$. As we maximize *aver-NDCG* from top to lower ranks, it is straightforward that the optimal way is to follow a greedy selection strategy to let an item of greater relevance R fulfill its exposure planning ΔE at its highest possible ranks. In Algorithm 5.1, the proposed vertical allocation exactly follows the above greedy selection strategy to let item of greater relevance R (line 11 in Algorithm 5.1) fulfill its exposure planning ΔE at the highest possible ranks (setting rank loops as the outer loop in 3 and line 4 of Algorithm 5.1). So it can reach optimal effectiveness at the top ranks. □

Theorem 4. *Effectiveness and fairness are fixed when ΔE is fixed.*

Proof. Given the same exposure planning ΔE, effectiveness (aver-NDCG@k_s) and fairness are fixed since we can substitute exposure planning $\Delta E(d)$ for $E^t@k_c(d)$ in Eq. 2.9 and substituting ΔE for E in Eq. 2.11, respectively. In other words, for any ranklist $\mathcal{B}^*$, as long as exposure planning ΔE can be accurately allocated in $\mathcal{B}^*$, the effectiveness and fairness

Table 5.1: Datasets statistics. For each dataset, the table below shows the number of queries, the average number of docs for each query, and the relevance annotation y's range.

Datasets	#Queries	#Aver. Docs per Query	y's range
MQ2008	800	20	0 – 2
MSLR-10k	10k	122	0 – 4
Istella-S	33k	103	0 – 4

are fixed. □

5.4 Experiments

5.4.1 Experimental setup

5.4.1.1 Datasets

In this work, we use three public Learning-to-Rank (LTR) datasets: MQ2008 [90], MSLR10k[4] and Istella-S [91]. Datasets' statistics are shown in Table 5.1. MQ2008 has a three-level relevance judgment (from 0 to 2). MSLR10k and Istella-S have a five-level relevance judgment (from 0 to 4). Queries in each dataset are already divided into training, validation, and test partitions according to a 60%-20%-20% scheme. In this work, we mainly focus on comparison within the LTR tasks. However, the proposed method can be adapted to recommendation tasks, which we leave for future studies.

5.4.1.2 Baselines

In this dissertation, we compare the following methods:

- TopK: Sort items according to $\hat{R}(d)$
- RandomK: Randomly rank items.
- FairCo [71]: Fair ranking algorithm based on a proportional controller. $\alpha \in [0.0, 1000.0]$
- MCFair [2]: Fair ranking algorithm directly uses gradient as the ranking score. $\alpha \in [0.0, 1000.0]$
- ILP [8]: Fair ranking algorithm based on Integer Linear Programming (ILP).$\alpha \in [0.0, 1.0]$
- LP [6]: Fair ranking algorithm based on Linear Programming (LP).$\alpha \in [0.0, 1000.0]$
- MMF [9]. Similar to FairCo but focus on top ranks fairness. $\alpha \in [0.0, 1.0]$

[4] https://www.microsoft.com/en-us/research/project/mslr/

- PLFair [11]. A fair ranking algorithm based on Placket-Luce optimization. $\alpha \in [0.0, 1.0]$
- FARA-Horiz. (ours): A variant of FARA. Compared to FARA, we switch line 3 and line 4 in Algorithm 5.1 to first iterate the sessions and then iterate the ranks. We refer to the iterations as the **horizontal allocation** paradigm. $\alpha \in [0.0, 1.0]$
- FARA (ours). The proposed fair ranking algorithm. $\alpha \in [0.0, 1.0]$.

Among the above ranking algorithm, TopK and RandomK are unfair algorithms, while the others are fair algorithms. While all the fair ranking algorithms aim to maximize effectiveness and fairness, FARA and FARA-Horiz. differ from others by taking a joint optimization across multiple ranklists rather than a traditional greedy optimization approach. For fair ranking algorithms, there exists a tradeoff parameter α, similar to α in Eq. 5.5, to balance effectiveness and fairness. For fair algorithms, the greater α is, the more we care about fairness while potentially sacrificing more effectiveness. For example, when increasing α in Eq. 5.5c, FARA can maximize fairness with less effectiveness constraint. For different fair algorithms, α lies in different ranges. For FairCo, MCFair, LP, α are originally within $[0.0, +\infty]$, and we adopt $\alpha \in [0.0, 1000.0]$ which is enough according to our experiments. For ILP, MMF, PLFair, FARA-Horiz. and FARA, $\alpha \in [0.0, 1.0]$. Although the vertical allocation in Algorithm 5.1 was inspired by [4], [4] cannot be used as a baseline because [4] works with offline ranking services where all user queries are known in advance. However, in this dissertation, we consider the online services depicted in Algorithm 5.2.

5.4.1.3 Ranking Service Simulation

Following the workflow in Algorithm 5.2, at each time step, a simulated user will issue a query q, which is randomly sampled from the training, validation, or test partition. Corresponding to the query q, a ranking algorithm will construct a ranklist π of candidate items and present it to the simulated user. To collect users' feedback for the ranked list π, we need to simulate relevance and examination (see Eq.2.13). Same as [74], the relevance probabilities of each document-query pair (d, q) are simulated with their relevance judgement y as

$$P(r = 1|d, q) = \epsilon + (1 - \epsilon)\frac{2^y - 1}{2^{y_{max}} - 1}$$

where y_{max} is the maximum value of relevance judgement y, i.e., 2 or 4 depending on the datasets. Besides relevance, following [13, 71], we simulate users' examination probability as,

$$P(e=1|d,\pi) = \begin{cases} \frac{1}{\log_2(rank(d|\pi)+1)}, & \text{if} \quad rank(d|\pi) \leq k_s \\ 0, & \text{otherwise} \end{cases}$$

For simplicity, we only simulate users' examination behavior on top ranks, and we set k_s to 5 throughout the experiments (refer to Eq. 2.14 for more details of k_s). With $P(r=1|d,q,\pi)$ and $P(e=1|d,\pi)$, we sample clicks with Equation 2.13. The advantage of the simulation is that it allows us to do online experiments on a large scale while still being easy to reproduce by researchers without access to live ranking systems [13]. For simplicity, same as existing works [9, 13, 14, 71], we assume that users' examination $P(e=1|d,\pi)$ is known in experiment since many existing works [74, 86, 87, 92] have been proposed to estimate it. Due to different data sizes, we simulate 200k steps for MQ2008 and 4M steps for MSLR10k and Istella-S.

5.4.1.4 Experiment Settings

We noticed that LP and ILP methods are proposed in **the post-processing setting**, where relevance is already known or well estimated in advance. However, in most real-world settings, ranking optimization and relevance learning are carried out at the same time, which we refer to as **the online setting**. To give a comprehensive comparison, we evaluate ranking methods in both settings. In the post-processing setting, all the ranking methods in Section 5.4.1.2 are based on true relevance R, and FARA will set β as 0. In the online setting, all the ranking methods in Section 5.4.1.2 are based on the relevance estimation $\hat{R}$ in Eq. 5.7 to perform ranking optimization. FARA set β to 1 and $E_{min} = 10$ unless otherwise explicitly specified, as they work well across all our experiments.

5.4.1.5 Evaluation

We use the cum-NDCG (cNDCG) in Eq. 2.7 with $\gamma = 0.995$ (same γ adopted in [20, 21]) to evaluate the effectiveness at different cutoffs, $1 \leq k_c \leq 5$. Aside from effectiveness, unfairness defined in Eq. 2.11 is used for unfairness measurement. We run each experiment five times and report the average evaluation performance on the test partition. We use the Fisher randomization test [88] with $p < 0.05$ to do significant tests. Due to the time cost

(see Table 5.2), we do not run ILP and LP on the larger datasets, MSLR10k and Istella-S, and the performances are not available (NA).

5.4.2 Results and Analysis

In this section, we first compare the ranking relevance performance given different degrees of fairness requirements. Then we dive deep into our method to offer more insights into FARA's supremacy.

5.4.2.1 Can FARA reach a better balance between effectiveness and fairness?

In Figure 5.2, we compare ranking methods' effectiveness-fairness balance given different fairness requirements. To generate the balance curves in Figure 5.2, we incrementally sample α from the minimum value to the maximum value within α's ranges indicated in Section 5.4.1.2. For each method, twenty α are sampled with the step size as $(\alpha_{max} - \alpha_{min})/20$. After sampling, we perform ranking simulation experiments for each α to get a (cNDCG, unfairness) pair. Then we connect different α's (cNDCG), unfairness) pair to form a curve for each method respectively in Figure 5.2. All the curves start from the top right to the bottom left as α increases, which means there exists a tradeoff between fairness and effectiveness (cNDCG). The reason behind this tradeoff is that requiring more fairness will bring more constraints on optimizing effectiveness. Since TopK and RandomK do not have trade-off parameters, both of them only have one single pair of (cNDCG, unfairness), and their performances are shown as single points in Figure 5.2.

In Figure 5.2, our methods FARA and FARA-Horiz. outperform all other fair methods since our methods reach the best cNDCG given the same unfairness tolerance. And FARA's supremacy is consistent in both post-processing and online settings. All fair ranking algorithms are effective fair ranking algorithms since they all show the tradeoff, i.e., higher cNDCG when increasing the unfairness tolerance. For unfair algorithms, TopK performs differently in post-processing and online settings. In the post-processing setting, TopK reaches the highest cNDCG since relevance is known, and ranking relevance is the only consideration. However, in the online setting, TopK can not reach the highest cNDCG. We think the drop in cNDCG is that TopK naively trusts the relevance estimation without any exploration when optimizing ranking relevance. However, fair algorithms are shown

to be robust to the online setting since they mostly can reach better cNDCG than Topk when increasing unfairness tolerance. We think the reason for the robustness is that fair algorithms usually rerank items for different sessions to optimize fairness, and such reranking brings explorations.

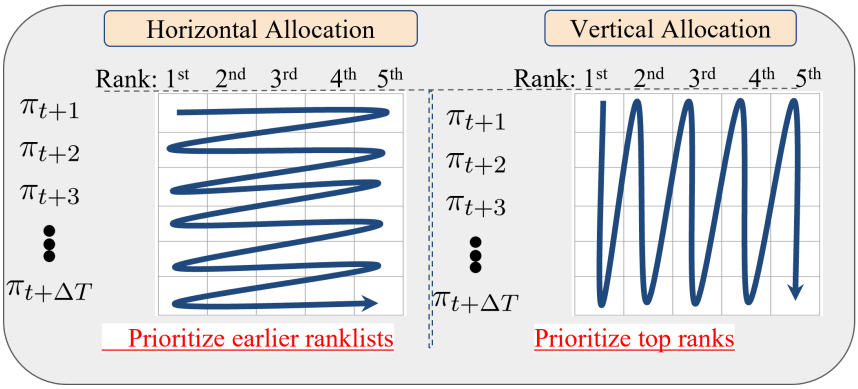


Figure 5.1: The ranklist construction order of the horizontal allocation and vertical allocation.

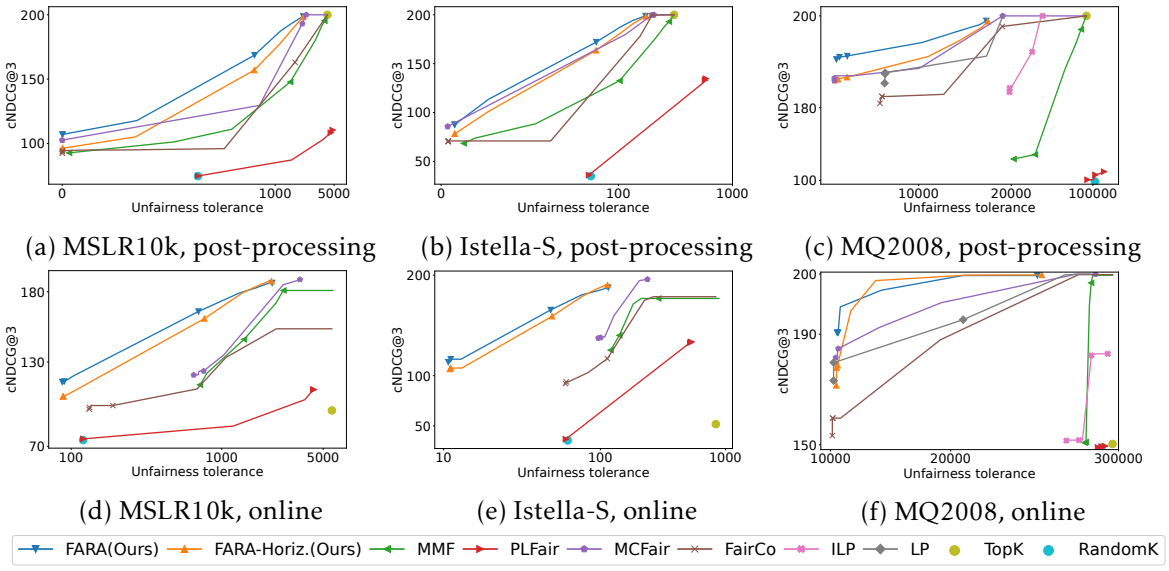


Figure 5.2: c-NDCG vs. unfairness tolerance in the post-processing setting and the online setting. Given the same unfairness, the higher curves or points lie, the better their performances are. Our methods FARA and FARA-Horiz. lie higher than all fair baselines in all figures. ILP and LP are unavailable for MSLR10k and Istella-S due to time costs (refer to Table 5.3).

Table 5.2: Comparison of cNDCG@(1,3,5) and unfairness tolerance in the post-processing setting. Significant improvements or degradations with respect to FairCo are indicated with +/-. Within fair algorithms, the best performance with statistical significance is bolded and underlined. Here, α is set to the maximum value (see Sec. 4.3.1.3 for α's range) for each fair algorithm respectively, which means that all algorithms are trying their best to optimize ranking fairness and the numbers in the table represents their unfairness lower bound. Results are rounded to one decimal place.

Methods	MSLR-10k				Istella-S				MQ2008			
	cNDCG@1	cNDCG@3	cNDCG@5	unfair.	cN@1	cN@3	cN@5	unfair.	cN@1	cN@3	cN@5	unfair.
TopK	200.0^{+}	200.0^{+}	200.0^{+}	4165.0^{-}	200.0^{+}	200.0^{+}	200.0^{+}	310.1^{-}	200.0^{+}	200.0^{+}	200.0^{+}	86001.1^{-}
Randomk	68.0^{-}	74.7^{-}	79.7^{-}	119.0^{-}	30.2^{-}	35.7^{-}	41.1^{-}	56.7^{-}	74.0^{-}	95.7^{-}	114.6^{-}	104632.2^{-}
PLFair	68.2^{-}	74.8^{-}	79.9^{-}	119.6^{-}	31.8^{-}	36.2^{-}	41.5^{-}	54.6^{-}	79.2^{-}	99.3^{-}	117.2^{-}	101245.1^{-}
MMF	84.4	92.8	99.9	8.0^{-}	62.2	68.8	80.1	6.6^{-}	132.8^{-}	162.3^{-}	172.5^{-}	20688.7^{-}
ILP	NA	NA	NA	NA	NA	NA	NA	NA	185.6^{+}	183.8	186.7	19916.3^{-}
LP	NA	NA	NA	NA	NA	NA	NA	NA	188.4^{+}	187.4^{+}	187.9	9425.7
MCFair	114.8^{+}	102.7^{+}	101.0	0.0	113.7^{+}	85.25^{+}	81.3	0.4	193.5^{+}	186.0^{+}	186.6	9113.7
FairCo	85.5	93.7	100.8	0.0	63.3	69.9	80.4	0.5	179.0	182.0	187.4	9382.0
FARA-Horiz.(Ours)	90.7^{+}	96.1^{+}	100.4	0.0	78.5^{+}	79.8^{+}	82.6	0.9	187.3^{+}	186.1^{+}	187.0	9125.9
FARA(Ours)	$\underline{\mathbf{129.0}}^{+}$	$\underline{\mathbf{107.0}}^{+}$	99.7	0.0	$\underline{\mathbf{135.5}}^{+}$	$\underline{\mathbf{89.1}}^{+}$	82.6	0.9	$\underline{\mathbf{196.3}}^{+}$	$\underline{\mathbf{190.9}}^{+}$	186.8	9129.9

5.4.2.2 What is the fairness upper bound that FARA can reach?

In Table 5.2, lower unfairness means higher fairness capacity and fairness upper bound, i.e., the maximum possible fairness one algorithm can reach. Fair effective ranking algorithms, including FairCo, LP, FARA-Horiz. and FARA, have similar fairness capacity and outperform unfair ranking algorithms in terms of unfairness. The success of FARA-Horiz. and FARA validates the proposed quadratic programming formulation can optimize fairness. Similar cNDCG@5 and unfairness for those effective algorithms are expected according to Theorem 4. For other fair ranking algorithms, ILP and MMF, and PLFair show inferior fairness capacity. As for the possible reason, ILP uses the integer linear programming method, which may not be effective in optimizing fairness. MMF actually follows a slightly different definition of fairness which requires fairness at any cutoff should be fair, which is more strict than the definition we use in this dissertation. As for PLFair, PLFair tries to learn the ranking score that optimizes fairness based on the feature representation (the exact setting in original paper [11]). However, the feature representation is initially designed for relevance which makes PLFair suboptimal for fairness optimization. In Table 5.2, ILP and LP are NA for MSLR10k and Istella-S due to time costs (refer to Table 5.3). Besides, we show the ranking performance of the online setting in Fig. (5.2).

5.4.2.3 How is FARA's effectiveness at different cutoffs?

In Table 5.2, we show cNDCG at different cutoffs. Although FairCo, LP, FARA-Horiz. and FARA have similar fairness capacities, FARA significantly outperforms those fair algorithms for cNDCG@1 and cNDCG@3 on all three datasets. Compared to FARA-Horiz., shown in Table 5.2, FARA still significantly outperforms FARA-Horiz. at top ranks, which shows the necessity of vertical allocation.

5.4.2.4 How is FARA's time efficiency?

Besides fairness and effectiveness optimization, we also empirically compare the time efficiency. In Table 5.3, ILP and LP are really time-consuming, especially on large datasets, MSLR10k and Istella-S. Compared to ILP and LP, FARA is more than 1000× time efficient on MSLR10k and Istella-S, although all three of them are programming-based methods. There are two reasons behind FARA's time efficiency. The first one is that FARA has a much

Table 5.3: The average time (seconds per 1k ranklists) cost with standard deviations in parentheses. Since ILP and LP are time-consuming on large datasets, the time costs on MSLR-10k and Istella-S are estimated by only running 1k steps instead of the total simulation steps indicated in Sec. 4.3.1.3.

Algorithms	Datasets		
	MSLR10k	Istella	MQ2008
TopK	$0.65_{(0.14)}$	$0.50_{(0.00)}$	$0.55_{(0.10)}$
Randomk	$0.63_{(0.12)}$	$0.57_{(0.04)}$	$0.59_{(0.14)}$
PLFair	$2.24_{(0.04)}$	$3.11_{(0.07)}$	$1.77_{(0.04)}$
MMF	$8.01_{(0.39)}$	$6.57_{(0.23)}$	$1.82_{(0.28)}$
ILP	$1208.90_{(85.80)}$	$1102.30_{(75.20)}$	$19.70_{(1.29)}$
LP	≥10 days	≥10 days	$2.09_{(0.48)}$
MCFair	$0.724_{(0.016)}$	$0.660_{(0.025)}$	$0.567_{(0.035)}$
FairCo	$0.73_{(0.04)}$	$0.71_{(0.03)}$	$0.70_{(0.12)}$
FARA-Horiz.(Ours)	$1.00_{(0.17)}$	$0.86_{(0.07)}$	$0.97_{(0.22)}$
FARA(Ours)	$0.91_{(0.07)}$	$0.91_{(0.00)}$	$0.97_{(0.30)}$

fewer number of decision variables since FARA only has $O(n)$ decision variables, while ILP and LP have $O(n^2)$ decision variables. The second one is that FARA does not need to solve quadratic programming for every time step. By solving quadratic programming once, we can get ΔT ranklists used for future ΔT sessions. FARA can reach comparable time efficiency with non-programming-based algorithms like TopK, RandomK, and FairCo. Compared with those non-programming-based algorithms, the slightly additional time cost of FARA is acceptable given FARA' superior ranking performance in Tab. 5.2 and in Fig. 5.2.

5.4.2.5 How does ΔT influence FARA?

In Figure 5.3, we show the results of cNDCG and unfairness by varying the value of ΔT. With greater ΔT, we see a clear boost of *cNDCG* for FARA, while such a boost does not happen for FARA-Horiz. We know that the proposed vertical allocation is the key reason to have better-ranking relevance when we get the optimal exposure planning ΔE^*, and we theoretically analyze the reason in **S** 3. Besides, as we increase ΔT, unfairness does not vary much, and its value stays close to the minimum unfairness we can achieve in Table 5.2. We think the reason for the steady value of unfairness is that FARA already reaches the upper limit of fairness when ΔT is small, and it is hard to improve when we increase ΔT.

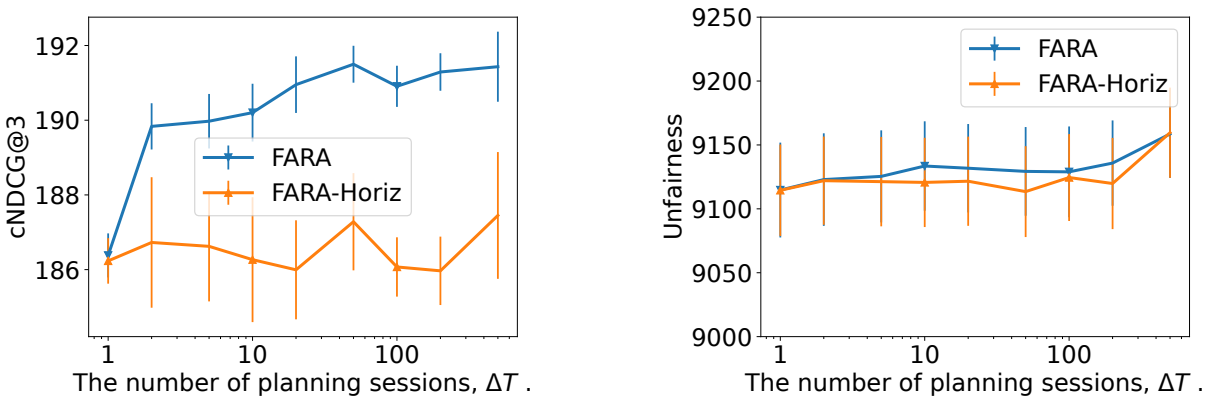
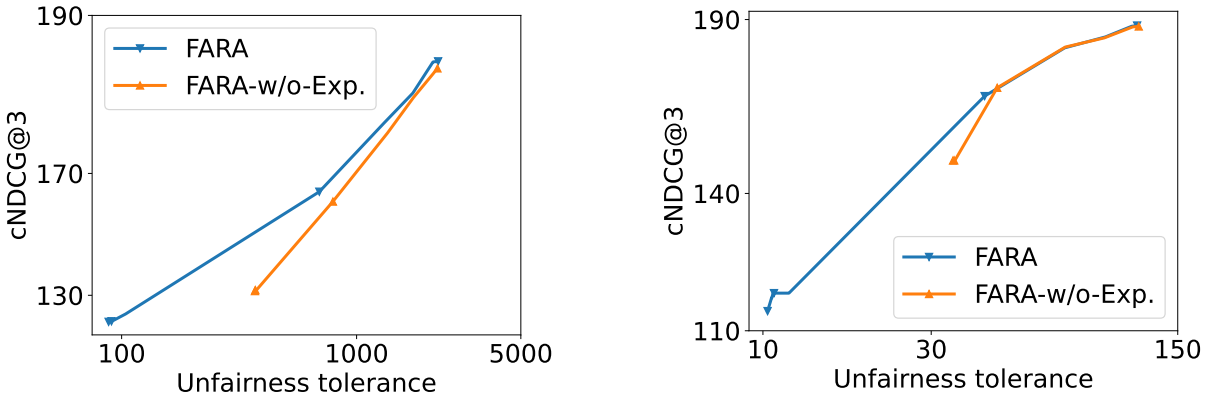


Figure 5.3: The numbers of planning session ΔT's influence on FARA in the post-processing setting on MQ2008. α is set as 1.



(a) MSLR10k.

(b) Istella-S

Figure 5.4: Ablation study of exploration in the online setting. The higher curves lie, the better their performances are.

5.4.2.6 How does exploration influence FARA?

To study how the exploration part (the slack variables s in Eq. 5.6) influences FARA, we did an ablation study for FARA with or without exploration. For simplicity, we only show the ablation results on the larger dataset, i.e., MSLR10k and Istella-S, in Figure 5.4. The advantage of exploration is two-folded based on Figure 5.4. Firstly, FARA lies higher than FARA-w/o-Exp. in the figure, which suggests exploration leads to a better effectiveness-fairness balance. Secondly, FARA has a smaller lower bound of unfairness tolerance, which implies exploration enables FARA to have a higher fairness capacity and can meet a more strict fairness requirement.

5.5 Conclusions

In this work, we found that existing fair ranking algorithms are usually greedy algorithms that sequentially produce a locally optimal ranklist for each session. To reach a global optimum, we propose FARA, a future-aware ranking algorithm for fairness, which optimizes multiple future sessions' ranklists together. With extensive experiments, FARA achieves better performance compared to existing fair ranking algorithms.

CHAPTER 6

SUMMARY AND FUTURE WORK

To summarize, in the first part of this dissertation, we proposed an uncertainty-aware empirical Bayes based ranking algorithm., which can overcome exploitation bias brought by behavior features in ranking models. We then developed a marginal-certainty-aware Fair ranking algorithm, that can jointly optimize effectiveness and fairness in an online setting. Finally, we proposed a novel future-aware ranking algorithm that can plan ahead by jointly optimizing multiple ranklists together and saving them for future sessions, instead of greedily optimizing rankings for the next immediate session.

There are several directions we can continue to explore.

Initially, in the realm of user behavior features, there is an opportunity to broaden the scope of our research by considering additional types of user behavior features beyond just user clicks. In contemporary Information Retrieval (IR), an array of user behaviors has become prominent, including user dwell time, purchase history, items added to the cart, user ratings, and more. These diverse forms of user behavior hold significant importance in accurately predicting ranking relevance. The central challenge in modern IR revolves around the effective utilization of these various user behavior signals while simultaneously safeguarding against the potential pitfalls of exploitation bias. This issue represents a crucial and complex problem that demands comprehensive exploration and resolution.

Second, In the previous works, I followed the well-defined amortized fairness principle [6, 8]. This principle requires items' cumulative exposure to be proportional to their relevance. It is simple but also ignores a very important fact that different items come to a ranking system at different times, i.e., starting to get exposure at different times. For example, two items, A and B, of the same relevance, but A came to the system before B. It is unfair to require A and B to have the same amount of exposure since A has a longer time to be exposed than B. Considering this, in this project, I plan to propose a time-aware

fairness metric and design optimization algorithms to optimize it based on my previous works.

REFERENCES

[1] T. Yang, C. Han, C. Luo, P. Gupta, J. M. Phillips, and Q. Ai, "Mitigating exploitation bias in learning to rank with an uncertainty-aware empirical bayes approach," *arXiv preprint arXiv:2305.16606*, 2023.

[2] T. Yang, Z. Xu, Z. Wang, A. Tran, and Q. Ai, "Marginal-certainty-aware fair ranking algorithm," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 24–32.

[3] T. Yang, Z. Xu, Z. Wang, and Q. Ai, "Fara: Future-aware ranking algorithm for fairness optimization," in *Proceedings of the 32nd ACM International Conference on Information & Knowledge Management (CIKM '23)*, 2023.

[4] T. Yang, Z. Xu, and Q. Ai, "Vertical allocation-based fair exposure amortizing in ranking," *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region (SIGIR-AP '23)*, 2023.

[5] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.

[6] A. Singh and T. Joachims, "Fairness of exposure in rankings," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2219–2228.

[7] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke, "Lerot: An online learning to rank framework," in *Proceedings of the 2013 workshop on Living labs for information retrieval evaluation*, 2013, pp. 23–26.

[8] A. J. Biega, K. P. Gummadi, and G. Weikum, "Equity of attention: Amortizing individual fairness in rankings," in *The 41st international acm sigir conference on research & development in information retrieval*, 2018, pp. 405–414.

[9] T. Yang and Q. Ai, "Maximizing marginal fairness for dynamic learning to rank," in *Proceedings of the Web Conference 2021*, 2021, pp. 137–145.

[10] A. Singh and T. Joachims, "Policy learning for fairness in ranking," in *Advances in Neural Information Processing Systems*, 2019, pp. 5426–5436.

[11] H. Oosterhuis, "Computationally efficient optimization of plackett-luce ranking models for relevance and fairness," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1023–1032.

[12] A. Chuklin, I. Markov, and M. d. Rijke, "Click models for web search," *Synthesis lectures on information concepts, retrieval, and services*, vol. 7, no. 3, pp. 1–115, 2015.

[13] H. Oosterhuis and M. de Rijke, "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*, 2021, pp. 463–471.

[14] T. Yang, C. Luo, H. Lu, P. Gupta, B. Yin, and Q. Ai, "Can clicks be both labels and features? unbiased behavior feature collection and uncertainty-aware learning to rank," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 6–17.

[15] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey, "An experimental comparison of click position-bias models," in *Proceedings of the 2008 international conference on web search and data mining*, 2008, pp. 87–94.

[16] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay, "Accurately interpreting clickthrough data as implicit feedback," in *ACM SIGIR Forum*, vol. 51, no. 1. Acm New York, NY, USA, 2017, pp. 4–11.

[17] H. Oosterhuis and M. de Rijke, "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*, 2020, pp. 489–498.

[18] Q. Ai, T. Yang, H. Wang, and J. Mao, "Unbiased learning to rank: Online or offline?" *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 2, pp. 1–29, 2021.

[19] K. Hofmann, S. Whiteson, and M. de Rijke, "Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval," *Information Retrieval*, vol. 16, no. 1, pp. 63–90, 2013.

[20] H. Wang, S. Kim, E. McCord-Snook, Q. Wu, and H. Wang, "Variance reduction in gradient exploration for online learning to rank," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 835–844.

[21] H. Wang, R. Langley, S. Kim, E. McCord-Snook, and H. Wang, "Efficient exploration of gradient space for online learning to rank," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 145–154.

[22] H. Wang, Y. Jia, and H. Wang, "Interactive information retrieval with bandit feedback," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2658–2661.

[23] Y. Yue and T. Joachims, "Interactively optimizing information retrieval systems as a dueling bandits problem," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 1201–1208.

[24] A. Schuth, H. Oosterhuis, S. Whiteson, and M. de Rijke, "Multileave gradient descent for fast online learning to rank," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 457–466.

[25] A. Schuth, R.-J. Bruintjes, F. Buüttner, J. van Doorn, C. Groenland, H. Oosterhuis,

C.-N. Tran, B. Veeling, J. van der Velde, R. Wechsler *et al.*, "Probabilistic multileave for online retrieval evaluation," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015, pp. 955–958.

[26] H. Oosterhuis and M. de Rijke, "Differentiable unbiased online learning to rank," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1293–1302.

[27] T. Joachims, A. Swaminathan, and T. Schnabel, "Unbiased learning-to-rank with biased feedback," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 781–789.

[28] Q. Ai, K. Bi, J. Guo, and W. B. Croft, "Learning a deep listwise context model for ranking refinement," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 135–144.

[29] T. Yang, S. Fang, S. Li, Y. Wang, and Q. Ai, "Analysis of multivariate scoring functions for automatic unbiased learning to rank," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2277–2280.

[30] A. Agarwal, K. Takatsu, I. Zaitsev, and T. Joachims, "A general framework for counterfactual learning-to-rank," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 5–14.

[31] D. Draper, "Assessment and propagation of model uncertainty," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 57, no. 1, pp. 45–70, 1995.

[32] M. Clyde and E. I. George, "Model uncertainty," *Statistical science*, vol. 19, no. 1, pp. 81–94, 2004.

[33] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[34] J. Zhu, J. Wang, M. Taylor, and I. J. Cox, "Risk-aware information retrieval," in *European Conference on Information Retrieval*. Springer, 2009, pp. 17–28.

[35] H. Roitman, S. Erera, and B. Weiner, "Robust standard deviation estimation for query performance prediction," in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, 2017, pp. 245–248.

[36] A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits, "Predicting query performance by query-drift estimation," *ACM Transactions on Information Systems (TOIS)*, vol. 30, no. 2, pp. 1–35, 2012.

[37] Y.-C. Lien, D. Cohen, and W. B. Croft, "An assumption-free approach to the dynamic truncation of ranked lists," in *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, 2019, pp. 79–82.

[38] J. S. Culpepper, C. L. Clarke, and J. Lin, "Dynamic cutoff prediction in multi-stage retrieval systems," in *Proceedings of the 21st Australasian Document Computing Sym-*

posium, 2016, pp. 17–24.

[39] D. Cohen, B. Mitra, O. Lesota, N. Rekabsaz, and C. Eickhoff, "Not all relevance scores are equal: Efficient uncertainty and calibration modeling for deep retrieval models," *arXiv preprint arXiv:2105.04651*, 2021.

[40] G. Penha and C. Hauff, "On the calibration and uncertainty of neural learning to rank models for conversational search," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 160–170.

[41] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[42] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "Letor: A benchmark collection for research on learning to rank for information retrieval," *Information Retrieval*, vol. 13, no. 4, pp. 346–374, 2010.

[43] E. Agichtein, E. Brill, and S. Dumais, "Improving web search ranking by incorporating user behavior information," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 19–26.

[44] C. Macdonald, R. L. Santos, and I. Ounis, "On the usefulness of query features for learning to rank," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 2559–2562.

[45] H. Oosterhuis and M. d. de Rijke, "Robust generalization and safe query-specializationin counterfactual learning to rank," in *Proceedings of the Web Conference 2021*, 2021, pp. 158–170.

[46] C. Li, B. Kveton, T. Lattimore, I. Markov, M. de Rijke, C. Szepesvári, and M. Zoghi, "Bubblerank: Safe online learning to re-rank via implicit click feedback," in *Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 196–206.

[47] B. Kveton, O. Meshi, M. Zoghi, and Z. Qin, "On the value of prior in online learning to rank," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 6880–6892.

[48] P. Gupta, T. Dreossi, J. Bakus, Y.-H. Lin, and V. Salaka, "Treating cold start in product search by priors," in *Companion Proceedings of the Web Conference 2020*, 2020, pp. 77–78.

[49] C. Han, P. Castells, P. Gupta, X. Xu, and V. Salaka, "Addressing cold start in product search via empirical bayes," in *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, ser. CIKM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3141–3151. [Online]. Available: https://doi.org/10.1145/3511808.3557066

[50] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[51] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 259–268.

[52] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Fairness-aware classifier with prejudice remover regularizer," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2012, pp. 35–50.

[53] W. Zhang and E. Ntoutsi, "Faht: an adaptive fairness-aware decision tree classifier," *arXiv preprint arXiv:1907.07237*, 2019.

[54] M. Kearns, S. Neel, A. Roth, and Z. S. Wu, "Preventing fairness gerrymandering: Auditing and learning for subgroup fairness," in *International Conference on Machine Learning*, 2018, pp. 2564–2572.

[55] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 98–108.

[56] A. Lambrecht and C. Tucker, "Algorithmic bias? an empirical study of apparent gender-based discrimination in the display of stem career ads," *Management Science*, vol. 65, no. 7, pp. 2966–2981, 2019.

[57] B. Edelman, M. Luca, and D. Svirsky, "Racial discrimination in the sharing economy: Evidence from a field experiment," *American Economic Journal: Applied Economics*, vol. 9, no. 2, pp. 1–22, 2017.

[58] D. Serbos, S. Qi, N. Mamoulis, E. Pitoura, and P. Tsaparas, "Fairness in package-to-group recommendations," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 371–379.

[59] Z. Zhu, X. Hu, and J. Caverlee, "Fairness-aware tensor-based recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1153–1162.

[60] A. Beutel, J. Chen, T. Doshi, H. Qian, L. Wei, Y. Wu, L. Heldt, Z. Zhao, L. Hong, E. H. Chi *et al.*, "Fairness in recommendation ranking through pairwise comparisons," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2212–2220.

[61] H. Abdollahpouri, R. Burke, and B. Mobasher, "Controlling popularity bias in learning-to-rank recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 42–46.

[62] R. Borges and K. Stefanidis, "On mitigating popularity bias in recommendations via variational autoencoders," in *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 2021, pp. 1383–1389.

[63] H. Abdollahpouri, G. Adomavicius, R. Burke, I. Guy, D. Jannach, T. Kamishima, J. Krasnodebski, and L. Pizzato, "Multistakeholder recommendation: Survey and research directions," *User Modeling and User-Adapted Interaction*, vol. 30, no. 1, pp.

127–158, 2020.

[64] E. Pitoura, K. Stefanidis, and G. Koutrika, "Fairness in rankings and recommendations: An overview," *arXiv preprint arXiv:2104.05994*, 2021.

[65] K. Yang and J. Stoyanovich, "Measuring fairness in ranked outputs," in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017, pp. 1–6.

[66] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates, "Fa* ir: A fair top-k ranking algorithm," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1569–1578.

[67] L. E. Celis, D. Straszak, and N. K. Vishnoi, "Ranking with fairness constraints," *arXiv preprint arXiv:1704.06840*, 2017.

[68] G. K. Patro, A. Biswas, N. Ganguly, K. P. Gummadi, and A. Chakraborty, "Fairrec: Two-sided fairness for personalized recommendations in two-sided platforms," in *Proceedings of The Web Conference 2020*, 2020, pp. 1194–1204.

[69] F. Guo, C. Liu, and Y. M. Wang, "Efficient multiple-click models in web search," in *Proceedings of the second acm international conference on web search and data mining*, 2009, pp. 124–131.

[70] G. E. Dupret and B. Piwowarski, "A user browsing model to predict search engine click data from past observations." in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 331–338.

[71] M. Morik, A. Singh, J. Hong, and T. Joachims, "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*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 429–438. [Online]. Available: https://doi.org/10.1145/3397271.3401100

[72] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[73] X. Wang, M. Bendersky, D. Metzler, and M. Najork, "Learning to rank with selection bias in personal search," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 115–124.

[74] Q. Ai, K. Bi, C. Luo, J. Guo, and W. B. Croft, "Unbiased learning to rank with unbiased propensity estimation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 385–394.

[75] O. Chapelle and Y. Chang, "Yahoo! learning to rank challenge overview," in *Proceedings of the learning to rank challenge*. PMLR, 2011, pp. 1–24.

[76] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie, "Smoothing clickthrough data for web search ranking," in *Proceedings of the 32nd international ACM SIGIR conference on*

Research and development in information retrieval, 2009, pp. 355–362.

[77] T. Yang, Z. Xu, Z. Wang, A. Tran, and Q. Ai, "Marginal-certainty-aware fair ranking algorithm," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 24–32.

[78] Y. Saito, S. Yaginuma, Y. Nishino, H. Sakata, and K. Nakata, "Unbiased recommender learning from missing-not-at-random implicit feedback," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 501–509.

[79] J. Bekker, P. Robberechts, and J. Davis, "Beyond the selected completely at random assumption for learning from positive and unlabeled data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 71–85.

[80] H. Raiffa, R. Schlaifer *et al.*, "Applied statistical decision theory," 1961.

[81] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[82] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. US Government printing office, 1964, vol. 55.

[83] D. Dato, C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, N. Tonellotto, and R. Venturini, "Fast ranking with additive ensembles of oblivious and non-oblivious regression trees," *ACM Transactions on Information Systems (TOIS)*, vol. 35, no. 2, pp. 1–31, 2016.

[84] A. Vardasbi, H. Oosterhuis, and M. de Rijke, "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*, 2020, pp. 1475–1484.

[85] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 621–630.

[86] X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork, "Position bias estimation for unbiased learning to rank in personal search," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 610–618.

[87] A. Agarwal, I. Zaitsev, X. Wang, C. Li, M. Najork, and T. Joachims, "Estimating position bias without intrusive interventions," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 474–482.

[88] M. D. Smucker, J. Allan, and B. Carterette, "A comparison of statistical significance tests for information retrieval evaluation," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 623–632.

[89] G. H. Hardy, J. E. Littlewood, G. Pólya, G. Pólya *et al.*, *Inequalities*. Cambridge university press, 1952.

[90] T. Qin and T.-Y. Liu, "Introducing letor 4.0 datasets," *arXiv preprint arXiv:1306.2597*, 2013.

[91] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, F. Silvestri, and S. Trani, "Post-learning optimization of tree ensembles for efficient ranking," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 949–952.

[92] F. Radlinski and T. Joachims, "Minimally invasive randomization for collecting unbiased preferences from clickthrough logs," in *Proceedings of the national conference on artificial intelligence*, vol. 21, no. 2. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 1406.

[93] S. E. Robertson, "The probability ranking principle in ir," *Journal of documentation*, 1977.

[94] G. K. Patro, L. Porcaro, L. Mitchell, Q. Zhang, M. Zehlike, and N. Garg, "Fair ranking: a critical review, challenges, and future directions," *arXiv preprint arXiv:2201.12662*, 2022.

[95] J. Kotary, F. Fioretto, P. Van Hentenryck, and Z. Zhu, "End-to-end learning for fair ranking systems," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 3520–3530.

[96] A. Tran, T. Yang, and Q. Ai, "Ultra: An unbiased learning to rank algorithm toolbox," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4613–4622.

[97] M. Zehlike, K. Yang, and J. Stoyanovich, "Fairness in ranking: A survey," *arXiv preprint arXiv:2103.14000*, 2021.

[98] Y. Wu, J. Cao, G. Xu, and Y. Tan, "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*, 2021, pp. 1013–1022.