End-to-End Learning for Fair Ranking Systems

James Kotary
Syracuse University
Syracuse, NY, USA
jkotary@syr.edu

Ferdinando Fioretto
Syracuse University
Syracuse, NY, USA
ffiorett@syr.edu

Pascal Van Hentenryck
Georgia Institute of Technology
Atlanta, GA, USA
pvh@isye.gatech.edu

Ziwei Zhu
Texas A&M University
College Station, TX, USA
zhuziwei@tamu.edu

ABSTRACT

The learning-to-rank problem aims at ranking items to maximize exposure of those most relevant to a user query. A desirable property of such ranking systems is to guarantee some notion of fairness among specified item groups. While fairness has recently been considered in the context of learning-to-rank systems, current methods cannot provide guarantees on the fairness of the predicted rankings. This paper addresses this gap and introduces *Smart Predict and Optimize for Fair Ranking (SPOFR)*, an integrated optimization and learning framework for fairness-constrained learning to rank. The end-to-end SPOFR framework includes a constrained optimization sub-model and produces ranking policies that are guaranteed to satisfy fairness constraints, while allowing for fine control of the fairness-utility tradeoff. SPOFR is shown to significantly improve on current state-of-the-art fair learning-to-rank systems with respect to established performance metrics.

CCS CONCEPTS

• **Computing methodologies → Machine learning algorithms**; • **Information systems → Learning to rank**; • **Applied computing** → *Multi-criterion optimization and decision-making*.

KEYWORDS

Learning to Rank; Fairness; Optimization; Decision Focused Learning; Smart Predict and Optimize

ACM Reference Format:
James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Ziwei Zhu. 2022. End-to-End Learning for Fair Ranking Systems. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512247

1 INTRODUCTION

Ranking systems are a pervasive aspect of our everyday lives: They are an essential component of online web searches, job searches, property renting, streaming content, and even potential friendships. In these systems, the items to be ranked are products, job candidates, or other entities associated with societal and economic benefits, and the relevance of each item is measured by implicit feedback from users (click data, dwell time, etc.). It has been widely recognized that the position of an item in the ranking has a strong influence on its exposure, selection, and, ultimately economic success.

The algorithms used to learn these rankings are typically oblivious to their potential disparate impact on the exposure of different groups of items. For example, it has been shown that in a job candidate ranking system, a small difference in relevance can incur a large difference in exposure for candidates from a minority group [7]. Similarly, in an image search engine, a disproportionate number of males may be shown in response to the query 'CEO' [16].

Ranking systems that ignore fairness considerations, or are unable to bound these effects, are prone to the "rich-get-richer" dynamics that exacerbate the disparate impacts. The resulting biased rankings can be detrimental to users, ranked items, and ultimately society. *There is thus a pressing need to design learning-to-rank (LRT) systems that can deliver accurate ranking outcomes while controlling disparate impacts.*

Current approaches to fairness in learning-to-rank systems rely on using a loss function representing a weighted combination of expected task performance and fairness. This strategy is effective in improving the fairness of predicted rankings on average, but has three key shortcomings: **(1)** The resulting rankings, even when fair in expectation across all queries, can admit large fairness disparities for some queries. This aspect may contribute to exacerbate the rich-get-richer dynamics, while giving a false sense of controlling the system's disparate impacts. **(2)** While a tradeoff between fairness and ranking utility is usually desired, these models cannot be directly controlled through the specification of an allowed magnitude for the violation of fairness. **(3)** A large hyperparameter search is required to find the weights of the loss function that deliver the desired performance tradeoff. Furthermore, each of these issues becomes worse as the number of protected groups increases.

This paper addresses these issues and proposes the first fair learning to rank system–named Smart Predict and Optimize for Fair Ranking (SPOFR)–that *guarantees* satisfaction of fairness in the resulting rankings. The proposed framework uses a unique

Symbol	Semantic
N	Size of the training dataset
n	Number of items to rank, for each query
$\boldsymbol{x}_q = (x_q^i)_{i=1}^n$	List of items to rank for query q
$\boldsymbol{a}_q = (a_q^i)_{i=1}^n$	protected groups associated with items x_q^i
$\boldsymbol{y}_q = (y_q^i)_{i=1}^n$	relevance scores (1-hot labels)
σ	Permutation of the list $[n]$ (individual rankings)
Π	Ranking policy
$\boldsymbol{v} = (v_i)_{i=1}^n$	Position bias vector
$\boldsymbol{w} = (w_i)_{i=1}^n$	Position discount vector

Table 1: Common symbols

integration of a constrained optimization model within a deep learning pipeline, which is trained end-to-end to produce *optimal* fair ranking policies with respect to empirical relevance scores.

Contributions The paper makes the following contributions: **(1)** It proposes *SPOFR*, a Fair LTR system that predicts and optimizes through an end-to-end composition of differentiable functions, guaranteeing the satisfaction of user-specified group fairness constraints. **(2)** Due to their discrete structure, imposing fairness constraints over ranking challenges the computation and back-propagation of gradients. To overcome this challenge, *SPOFR* introduces a novel training scheme which allows direct optimization of empirical utility metrics on predicted rankings using efficient back-propagation through constrained optimization programs. **(3)** The model ensures uniform fairness guarantees over all queries, a directly controllable fairness-utility tradeoff, and guarantees for multi-group fairness. **(4)** These unique aspects are demonstrated on two LTR datasets in the partial information setting. Additionally, SPOFR is shown to significantly improve on current state-of-the-art fair LTR systems with respect to established performance metrics.

2 RELATED WORK

The imposition of fairness constraints over discrete rankings can require nontrivial optimizations. To address this challenge, multiple notions of fairness in ranking have been developed. Celis et al. [6] propose to directly require fair representation between groups within each prefix of a ranking, by specifying a mixed integer programming problem to solve for rankings of the desired form. Zehlike et al. [21] design a greedy randomized algorithm to produce rankings which satisfy fairness up to a threshold of statistical significance. The approach taken by Singh and Joachims [16] also constructs a randomized ranking policy by formalizing the ranking policy as a solution to a linear optimization problem with constraints ensuring fair exposure between groups in expectation.

Fairness in learning-to-rank is studied by Zehlike and Castillo [22], which adopts the LTR approach of Cao et al. [5] and introduces a penalty term to the loss function to account for the violation of group fairness in the top ranking position. Stronger fairness results are reported by Yadav et al. [20] and Singh and Joachims [17], which apply a policy gradient method to learn fair ranking policies. The notion of fairness is enforced by a penalty to its violation in the loss function, forming a weighted combination of terms representing fairness violation and ranking utility over rankings sampled from the learned polices using a REINFORCE algorithm [19].

3 SETTINGS AND GOALS

The LTR task consists in learning a mapping between a list of n *items* and a permutation σ of the list $[n]$, which defines the order in which the items should be ranked in response to a user query. The LTR setting considers a training dataset $\boldsymbol{D} = (\boldsymbol{x}_q, \boldsymbol{a}_q, \boldsymbol{y}_q)_{q=1}^N$ where the $\boldsymbol{x}_q \in \mathcal{X}$ describe lists $(x_q^i)_{i=1}^n$ of n items to rank, with each item x_q^i defined by a feature vector of size k. The $\boldsymbol{a}_q = (a_q^i)_{i=1}^n$ elements describe protected group attributes in some domain $\mathcal{G}$ for each item x_q^i. The $\boldsymbol{y}_q \in \mathcal{Y}$ are supervision labels $(y_q^i)_{i=1}^n$ that associate a non-negative value, called *relevance scores*, with each item. Each sample $\boldsymbol{x}_q$ and its corresponding label $\boldsymbol{y}_q$ in $\boldsymbol{D}$ corresponds to a unique *query* denoted q. For example, on a image web-search context, a query q denotes the search keywords, e.g., "nurse", the feature vectors x_q^i in $\boldsymbol{x}_q$ encode representations of the items relative to q, the associated protected group attribute a_q^i may denote gender or race, and the label y_q^i describes the relevance of item i to query q.

The goal of learning to rank is to predict, for any query q, a distribution of rankings Π, called a *ranking policy*, from which individual rankings can be sampled. The utility U of a ranking policy Π for query q is defined as

$$U(\Pi, q) = \mathbb{E}_{\sigma\sim\Pi}\left[\Delta(\sigma, \boldsymbol{y}_q)\right], \tag{1}$$

where Δ measures the utility of a given ranking with respect to given relevance scores $\boldsymbol{y}_q$.

Let $\mathcal{M}_\theta$ be a machine learning model, with parameters θ, which takes as input a query and returns a ranking policy. The LTR goal is to find parameters θ^* that maximize the empirical risk:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ \frac{1}{N}\sum_{q=1}^{N} U(\mathcal{M}_\theta(\boldsymbol{x}_q), \boldsymbol{y}_q). \tag{P}$$

This description refers to the *Full-Information* setting [11], in which all target relevance scores are assumed to be known. While this setting is described to ease notation, the methods proposed in this work are not limited to this setting and Section 7 and Appendix B.1 discuss the *Partial-Information* setting.

Fairness. This paper aims at learning ranking policies that satisfy group fairness. It considers a predictor $\mathcal{M}$ satisfying some group fairness notion on the learned ranking policies with respect to protected attributes $\boldsymbol{a}_q$. A desired property of fair LTR models is to ensure that, for a given query, items associated with different groups receive equal exposure over the ranked list of items. The exposure $\mathcal{E}(i, \sigma)$ of item i within some ranking σ is a function of only its position, with higher positions receiving more exposure than lower ones. Thus, similar to [17], this exposure is quantified by $\mathcal{E}(i, \sigma) = v_{\sigma_i}$, where the *position bias* vector $\boldsymbol{v}$ is defined with elements $v_j = 1/(1+j)^p$, for $j \in [n]$ and with $p > 0$ being an arbitrary power.

For ranking policy $\mathcal{M}_\theta(\boldsymbol{x}_q)$ and query q, fairness of exposure requires that, for every group indicator $g \in \mathcal{G}$, $\mathcal{M}$'s rankings are statistically independent of the protected attribute g:

$$\mathbb{E}_{\substack{\sigma\sim\mathcal{M}_\theta(\boldsymbol{x}_q)\\ i\sim[n]}}\left[\mathcal{E}(i, \sigma)\,|a_q^i = g\right] = \mathbb{E}_{\substack{\sigma\sim\mathcal{M}_\theta(\boldsymbol{x}_q)\\ i\sim[n]}}\left[\mathcal{E}(i, \sigma)\right]. \tag{2}$$

This paper considers bounds on fairness defined as the difference between the group and population level terms, i.e.,

$$\nu(\mathcal{M}_\theta(\boldsymbol{x}_q), g) = \mathbb{E}\left[\mathcal{E}(i,\sigma) | a_q^i = g\right] - \mathbb{E}\left[\mathcal{E}(i,\sigma)\right]. \quad (3)$$

Definition 1 (δ-fairness). *A model $\mathcal{M}_\theta$ is δ-fair, with respect to exposure, if for any query $q \in [N]$ and group $g \in \mathcal{G}$:*

$$\left|\nu(\mathcal{M}_\theta(\boldsymbol{x}_q), g)\right| \leq \delta.$$

In other words, the fairness violation on the resulting ranking policy is upper bounded by $\delta \geq 0$.

The goal of this paper is to design accurate LTR models that guarantee δ-fairness for any prescribed fairness level $\delta \geq 0$. As noted by Agarwal et al. [1] and Fioretto et al. [10], several fairness notions, including those considered in this paper, can be viewed as linear constraints between the properties of each group with respect to the population. While the above description focuses on exposure, the methods discussed here can handle any fairness notion that can be formalized as a (set of) linear constraints, including *merit weighted fairness*, introduced in Section 5.2. A summary of the common adopted symbols is provided in Table 1.

4 LEARNING FAIR RANKINGS: CHALLENGES

When interpreted as constraints of the form (3), fairness properties can be explicitly imposed to problem (P), resulting in a constrained empirical risk problem, formalized as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \; \frac{1}{N} \sum_{q=1}^{N} U(\mathcal{M}_\theta(\boldsymbol{x}_q), \boldsymbol{y}_q) \quad (4a)$$

$$\texttt{s.t.} \quad \left|\nu(\mathcal{M}_\theta(\boldsymbol{x}_q), g)\right| \leq \delta \;\; \forall q \in [N], g \in \mathcal{G}. \quad (4b)$$

Solving this new problem, however, becomes challenging due to the presence of constraints. Rather than enforcing constraints (4b) exactly, state of the art approaches in fair LTR (e.g., [16, 20]) rely on augmenting the loss function (4a) with a term that penalizes the constraint violations ν weighted by a multiplier λ. This approach, however, has several undesirable properties:

1. Because the constraint violation term is applied at the level of the loss function, it applies only on average over the samples encountered during training. Because the sign ($\pm$) of a fairness violation depends on which group is favored, disparities in favor of one group can cancel out those in favor of another group for different queries. This can lead to models which predict individual policies that are far from satisfying fairness in expectation, as desired. These effects will be shown in Section 7.
2. The multiplier λ must be treated as a hyperparameter, increasing the computational effort required to find desirable solutions. This is already challenging for binary groups and the task becomes (exponentially) more demanding with the increasing of the number of protected groups.
3. When a tradeoff between fairness and utility is desired, it cannot be controlled by specifying an allowable magnitude for fairness violation. This is due to the lack of a reliable relationship between the hyperparameter λ and the resulting constraint violations. In particular, choosing λ to satisfy Definition 1 for a given δ is near-impossible due to the sensitivity and unreliability of the relationship between these two values.

The approach proposed in this paper avoids these difficulties by providing an end-to-end integration of predictions and optimization into a single machine-learning pipeline, where (1) fair policies are obtained by an optimization model using the predicted relevance scores and (2) the utility metrics are back-propagated from the loss function to the inputs, through the optimization model and the predictive models. This also ensures that the fairness constraints are satisfied on *each* predicted ranking policy.

5 SPOFR

Overview. The underlying idea behind SPOFR relies on the realization that constructing an optimal ranking policy Π_q associated with a query q can be cast as a linear program (as detailed in the next section) which relies only on the relevance scores $\boldsymbol{y}_q$. The cost vector of the objective function of this program is however not observed, but can be predicted from the feature vectors x_q^i ($i \in [n]$) associated with the item list $\boldsymbol{x}_q$ to rank. The resulting framework thus operates into three steps:

1. First, for a given query q and its associated item list $\boldsymbol{x}_q$, a neural network model $\mathcal{M}_\theta$ is used to predict relevance scores $\hat{\boldsymbol{y}}_q = (\hat{y}_q^1, \ldots, \hat{y}_q^n)$;
2. Next, the predicted relevance scores are used to specify the objective function of a linear program whose solution will result in a *fair* optimal (with the respect to the predicted scores) ranking policy $\Pi^*(\hat{\boldsymbol{y}}_q)$;
3. Finally, a regret function, which measures the loss of optimality relative to the true optimal policy $\Pi^*(\boldsymbol{y}_q)$ is computed, and gradients are back-propagated along each step, including in the argmax operator adopted by the linear program, creating an end-to-end framework.

The overall scheme is illustrated in Figure 1. It is important to note that, rather than minimizing a standard error (such as a mean square loss) between the predicted quantities $\hat{\boldsymbol{y}}_q$ and the target scores $\boldsymbol{y}_q$, SPOFR minimizes directly a loss in optimality of the predicted ranking with respect to the optimal ones. Minimizing this loss is however challenging as ranking are discrete structures, which requires to back-propagate gradients through a linear program. These steps are examined in detail in the rest of this section.

While the proposed framework is general and can be applied to any linear utility metric U for rankings (see Problem (1)), this section grounds the presentation on a widely adopted utility metric, the *Discounted Cumulative Gain (DCG)*:

$$DCG(\sigma, \boldsymbol{y}_q) = \sum_{i=1}^{n} y_q^i w_{\sigma_i}, \quad (5)$$

where σ is a permutation over $[n]$, $\boldsymbol{y}_q$ are the true relevance scores, and $\boldsymbol{w}$ is an arbitrary weighting vector over ranking positions, capturing the concept of *position discount.* Commonly, and throughout this paper, $w_i = 1/\log_2(1+i)$. Note that following [17, 20], these discount factors are considered distinct from the position bias factors $\boldsymbol{v}$ used in the calculation of group exposure.

5.1 Predict: Relevance Scores

Given a query q with a list of items $\boldsymbol{x}_q = (x_q^1, \ldots, x_q^n)$ to be ranked, the *predict* step uses a single fully connected ReLU neural network $\mathcal{M}_\theta$ acting on each individual item x_q^i to predict a score $\hat{y}_q^i$ ($i =$

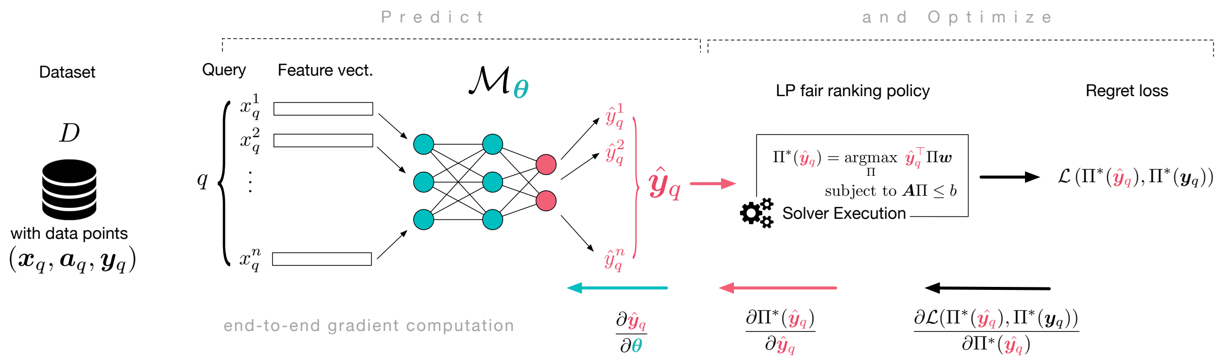
Predict
and Optimize
Dataset
D
with data points
$(\boldsymbol{x}_q, \boldsymbol{a}_q, \boldsymbol{y}_q)$
Query
Feature vect.
q
x_q^1
x_q^2
x_q^n
$\mathcal{M}_{\boldsymbol{\theta}}$
$\hat{y}_q^1$
$\hat{y}_q^2$
$\hat{y}_q^n$
$\hat{\boldsymbol{y}}_q$
LP fair ranking policy
$\Pi^*(\hat{y}_q) = \text{argmax}_{\Pi}\ \hat{y}_q^\top \Pi \boldsymbol{w}$ subject to $\boldsymbol{A}\Pi \le b$
Solver Execution
Regret loss
$\mathcal{L}(\Pi^*(\hat{y}_q), \Pi^*(\boldsymbol{y}_q))$
end-to-end gradient computation
$\frac{\partial \hat{y}_q}{\partial \boldsymbol{\theta}}$
$\frac{\partial \Pi^*(\hat{y}_q)}{\partial \hat{y}_q}$
$\frac{\partial \mathcal{L}(\Pi^*(\hat{y}_q), \Pi^*(\boldsymbol{y}_q))}{\partial \Pi^*(\hat{y}_q)}$

Figure 1: SPOFR. A single neural network learns to predict item scores from individual feature vectors, which are used to construct a linear objective function for the constrained program that produces a ranking policy.

Model 1 LP Computing the Fair Ranking Policy

$$\Pi^*(\hat{\boldsymbol{y}}_q) = \text{argmax}_{\Pi}\ \hat{\boldsymbol{y}}_q^\top \Pi \boldsymbol{w} \tag{6a}$$
$$\text{subject to: } \sum_j \Pi_{ij} = 1 \quad \forall i \in [n] \tag{6b}$$
$$\sum_i \Pi_{ij} = 1 \quad \forall j \in [n] \tag{6c}$$
$$0 \le \Pi_{ij} \le 1 \quad \forall i, j \in [n] \tag{6d}$$
$$|\nu(\Pi, g)| \le \delta \quad \forall g \in \mathcal{G} \tag{6e}$$

$1, \ldots, n$). Combined, the predicted scores for query q are denoted with $\hat{\boldsymbol{y}}_q$ and serve as the cost vector associated with the optimization problem solved in the next phase.

5.2 Optimize: Fair Ranking Policies

The predicted relevance scores $\hat{\boldsymbol{y}}_q$, combined with the constant position discount values $\boldsymbol{w}$, can be used to form a linear function that estimates the utility metric (*DCG*) of a ranking policy. Expressing the utility metric as a linear function makes it possible to form the LTR model as an end-to-end continuous function.

Linearity of the Utility Function. The following description omits subscripts "$_q$" for readability. The references below to ranking policy Π and relevance scores $\boldsymbol{y}$ are to be interpreted in relation to an underlying query q.

Using the Birkhoff–von Neumann decomposition [4], any $n \times n$ doubly stochastic matrix Π[1] can be decomposed into a convex combination of at most $(n-1)^2 + 1$ permutation matrices $P^{(i)}$, each associated with a coefficient $\mu_i \le 0$, which can then represent rankings $\sigma^{(i)}$ under the interpretation $\boldsymbol{w}_{\sigma^{(i)}} = P^{(i)}\boldsymbol{w}$. A ranking policy is inferred from the set of resulting convex coefficients μ_i, which sum to one, forming a discrete probability distribution: each permutation has likelihood equal to its respective coefficient

$$\Pi = \sum_{i=1}^{(n-1)^2+1} \mu_i P^{(i)}. \tag{7}$$

[1] A slight abuse of notation is used to refer to Π as a matrix of marginal probabilities encoding the homonyms ranking policy.

Next, note that any linear function on rankings can be formulated as a linear function on their permutation matrices, which can then be applied to any square matrix. In particular, applying the DCG operator to a doubly stochastic matrix Π results in the expected DCG over rankings sampled from its inferred policy. Given item relevance scores $\boldsymbol{y}$:

$$\mathbb{E}_{\sigma \sim \Pi} DCG(\sigma, \boldsymbol{y}) = \sum_{i=1}^{(n-1)^2+1} \mu_i\, \boldsymbol{y}^\top P^{(i)}\, \boldsymbol{w}$$
$$= \boldsymbol{y}^\top \left(\sum_{i=1}^{(n-1)^2+1} \mu_i\, P^{(i)} \right) \boldsymbol{w} = \boldsymbol{y}^\top \Pi \boldsymbol{w}. \quad \text{(by Eq. (7))}$$

The expected DCG of a ranking sampled from a ranking policy Π can thus be represented as a linear function on Π, which serves as the objective function for Model 1 (see Equation (6a)). This analytical evaluation of expected utility is key to optimizing fairness-constrained ranking policies in an end-to-end manner.

Importantly, and in contrast to state-of-the art methods, this approach does not require sampling from ranking policies during training in order to evaluate ranking utilities. Sampling is only required during deployment of the ranking model.

Ranking Policy Constraints. Note that, with respect to *any* such linear objective function, the optimal fair ranking policy Π^* can be found by solving a linear program (LP). The linear programming model for optimizing fair ranking *DCG* functions is presented in Model 1, which follows the formulations presented in [16].

The ranking policy predicted by the SPOFR model takes the form of a doubly stochastic $n \times n$ matrix Π, in which Π_{ij} represents the marginal probability that item i takes position j within the ranking. The doubly stochastic form is enforced by equality constraints which require each row and column of Π to sum to 1. With respect to row i, these constraints express that the likelihood of item i taking any of n possible positions must be equal to 1 (Constraints (6b)). Likewise, the constraint on column j says that the total probability of some item occupying position j must also be 1 (Constraints (6c)).

For the policy implied by Π to be fair, additional *fairness constraints* must be introduced.

Fairness Constraints. Enforcing fairness requires only one additional set of constraints, which ensures that the exposures are allocated fairly among the distinct groups. The expected exposure of item i in rankings σ derived from a policy matrix Π can be expressed in terms of position bias factors $\boldsymbol{v}$ as $\mathbb{E}_{\sigma\sim\Pi}\mathcal{E}(i,\sigma) = \sum_{j=1}^{n}\Pi_{ij}v_j$. The δ-fairness of exposure constraints associated with predicted ranking policy Π and group $g \in \mathcal{G}$ becomes:

$$\left|\left(\frac{1}{|G_q^g|}\mathbb{1}_g - \frac{1}{n}\mathbb{1}\right)^{\top}\Pi\,\boldsymbol{v}\right| \leq \delta, \tag{8}$$

where, $G_q^g = \{i : (a_q^i) = g\}$, $\mathbb{1}$ is the vector of all ones, and $\mathbb{1}_g$ is a vector whose values equal to 1 if the corresponding item to be ranked is in G_q^g, and 0 otherwise. This definition is consistent with that of Equation (2). It is also natural to consider a notion of weighted fairness of exposure:

$$\left|\left(\frac{\mu}{|G_q^g|}\mathbb{1}_g - \frac{\mu_g}{n}\mathbb{1}\right)^{\top}\Pi\,\boldsymbol{v}\right| \leq \delta, \tag{9}$$

which specifies that group g receive exposure in proportion to the weight μ_g. In this paper, where applicable and for a notion of *merit-weighted* fairness of exposure, μ_g is chosen to be the average relevance score of items in group g, while μ is the average over all items. Note that, while the above are natural choices for fairness in ranking systems, any linear constraint can be used instead.

5.3 Regret Loss and SPO Training

The training of the end-to-end fair ranking model uses a loss function that minimizes the *regret* between the exact and approximate policies, i.e.,

$$\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \boldsymbol{y}^{\top}\Pi^{*}(\boldsymbol{y})\,\boldsymbol{w} - \boldsymbol{y}^{\top}\Pi^{*}(\hat{\boldsymbol{y}})\,\boldsymbol{w}. \tag{10}$$

To train the model with stochastic gradient descent, the main challenge is the back-propagation through Model (1), i.e., the final operation of our learnable ranking function. It is well-known that a parametric linear program with fixed constraints is a nonsmooth mapping from objective coefficients to optimal solutions. Nonetheless, effective approximations for the gradients of this mapping can be found [8] (see also [12] for a review on the topic). Consider the optimal solution to a linear programming problem with fixed constraints, as a function of its cost vector $\hat{\boldsymbol{y}}$:

$$\Pi^{*}(\hat{\boldsymbol{y}}) = \underset{\Pi}{\operatorname{argmax}}\ \hat{\boldsymbol{y}}^{\top}\Pi \quad \texttt{s.t.}\ \ A\Pi \leq b,$$

with A and b being an arbitrary matrix and vector, respectively. Given candidate costs $\hat{\boldsymbol{y}}$, the resulting optimal solution $\Pi^{*}(\hat{\boldsymbol{y}})$ can be evaluated relative to a known cost vector $\boldsymbol{y}$. Further, the resulting objective value can be compared to that of the optimal objective under the known cost vector using the regret metric $\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}})$.

The regret measures the loss in objective value, relative to the true cost function, induced by the predicted cost. It is used as a loss function by which the predicted linear program costs vectors can be supervised by ground-truth values. However, the regret function is discontinuous with respect to $\hat{\boldsymbol{y}}$ for fixed $\boldsymbol{y}$. Following the approach pioneered in [8], this paper uses a convex surrogate loss function, called the SPO+ loss, which forms a convex upper-bounding function

Algorithm 1: *Training the Fair Ranking Function*

```
input: $D, \alpha, \boldsymbol{w}$ : Training Data, Learning Rate, Position Discount.
1 for epoch $k = 0, 1, \ldots$ do
2     foreach $(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}) \leftarrow D$ do
3         $\hat{\boldsymbol{y}} \leftarrow \mathcal{M}_{\theta}(\boldsymbol{x})$
4         $\Pi_1 \leftarrow \Pi^{*}(\boldsymbol{y}^{\top}\boldsymbol{w})$ by Model 1
5         $\Pi_2 \leftarrow \Pi^{*}(2\hat{\boldsymbol{y}}^{\top}\boldsymbol{w} - \boldsymbol{y}^{\top}\boldsymbol{w})$ by Model 1
6         $\nabla\mathcal{L}(\boldsymbol{y}^{\top}\boldsymbol{w}, \hat{\boldsymbol{y}}^{\top}\boldsymbol{w}) \leftarrow \Pi_2 - \Pi_1$
7         $\theta \leftarrow \theta - \alpha\nabla\mathcal{L}(\boldsymbol{y}^{\top}\boldsymbol{w}, \hat{\boldsymbol{y}}^{\top}\boldsymbol{y})\frac{\partial\hat{\boldsymbol{y}}^{\top}\boldsymbol{w}}{\partial\theta}$
```

over $\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}})$. Its gradient is computed as follows:

$$\frac{\partial}{\partial\boldsymbol{y}}\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}}) \approx \frac{\partial}{\partial\boldsymbol{y}}\mathcal{L}_{\mathsf{SPO+}}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \Pi^{*}(2\hat{\boldsymbol{y}} - \boldsymbol{y}) - \Pi^{*}(\boldsymbol{y}). \tag{11}$$

Remarkably, risk bounds about the SPO+ loss relative to the SPO loss can be derived [13], and the empirical minimizer of the SPO+ loss is shown to achieve low excess true risk with high probability. Note that, by definition, $\boldsymbol{y}^{\top}\Pi^{*}(\boldsymbol{y}) \geq \boldsymbol{y}^{\top}\Pi^{*}(\hat{\boldsymbol{y}})$ and therefore $\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}}) \geq 0$. Hence, finding the $\hat{\boldsymbol{y}}$ minimizing $\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}})$ is equivalent to finding the $\hat{\boldsymbol{y}}$ maximizing $\boldsymbol{y}^{\top}\Pi^{*}(\hat{\boldsymbol{y}})$, since $\boldsymbol{y}^{\top}\Pi^{*}(\boldsymbol{y})$ is a constant value.

In the context of fair learning to rank, the goal is to predict the cost coefficients $\hat{\boldsymbol{y}}$ for Model 1 which maximize the empirical DCG, equal to $\boldsymbol{y}^{\top}\,\Pi^{*}(\hat{\boldsymbol{y}})\,\boldsymbol{w}$ for ground-truth relevance scores $\boldsymbol{y}$. A vectorized form can be written:

$$\boldsymbol{y}^{\top}\Pi\boldsymbol{w} = \overrightarrow{(\boldsymbol{y}^{\top}\boldsymbol{w})}\cdot\overrightarrow{\Pi}, \tag{12}$$

where $\overrightarrow{A}$ represents the row-major-order vectorization of a matrix A. Hence, the regret induced by prediction of cost coefficients $\hat{\boldsymbol{y}}$ is

$$\mathcal{L}(\boldsymbol{y},\hat{\boldsymbol{y}}) = \overrightarrow{(\boldsymbol{y}^{\top}\boldsymbol{w})}\cdot\overrightarrow{\Pi^{*}(\boldsymbol{y})} - \overrightarrow{(\boldsymbol{y}^{\top}\boldsymbol{w})}\cdot\overrightarrow{\Pi^{*}(\hat{\boldsymbol{y}})}. \tag{13}$$

Note that while the cost coefficients $\boldsymbol{y}$ can be predicted generically (i.e., predicting an n^2-sized matrix), the modeling approach taken in this paper is to predict item scores independently from individual feature vectors (resulting in an n-sized vector). These values combine naturally with the known position bias values $\boldsymbol{v}$, to estimate DCG in the absence of true item scores. This simplification allows for learning independently over individual feature vectors, and was found in practice to outperform frameworks which use larger networks which take as input the entire feature vector lists.

Algorithm 1 maximizes the expected DCG of a learned ranking function by minimizing this regret. Its gradient is approximated as

$$\overrightarrow{\Pi^{*}(2\hat{\boldsymbol{y}}^{\top}\boldsymbol{w} - \boldsymbol{y}^{\top}\boldsymbol{w})} - \overrightarrow{\Pi^{*}(\boldsymbol{y}^{\top}\boldsymbol{w})}, \tag{14}$$

with $\hat{\boldsymbol{y}}$ predicted as described in Section 5.1. To complete the calculation of gradients for the fair ranking model, the remaining chain rule factor of line 7 is completed using the typical automatic differentiation.

6 MULTIGROUP FAIRNESS

SPOFR generalizes naturally to more than two groups. In contrast, multi-group fairness raises challenges for existing approaches that rely on penalty terms in the loss function [17, 20, 22]. Reference [20] proposes to formulate multi-group fairness using the single constraint

$$\sum_{g \neq g'} \left| \left(\frac{1}{|G_q^g|} \mathbb{1}_{G_q^g} - \frac{1}{|G_q^{g'}|} \mathbb{1}_{G_q^{g'}} \right)^\top \Pi \, \boldsymbol{v} \right| \leq \delta \tag{15}$$

where g and g' are groups indicators in $\mathcal{G}$, so that the average pairwise disparity between groups is constrained. However, this formulation suffers when $\delta \geq 0$, because the allowed fairness gap can be occupied by disparities associated with a single group in the worst case. Multiple constraints are required to provide true multi-group fairness guarantees and allow a controllable tradeoff between muti-group fairness and utility. Furthermore, the constraints (8) ensure satisfaction of (15) for appropriately chosen δ and are thus a generalization of (15). If unequal group disparities are desired, δ may naturally be chosen differently for each group in the equations (8).

7 EXPERIMENTS

This section evaluates the performance of SPOFR against the prior approaches of [20] and [22], the current state-of-the-art methods for fair learning rank, which are denoted by FULTR and DELTR respectively. The experimental evaluation follows the more realistic *Partial Information setting* described in [20]. A formal description of this setting is deferred to Appendix B.1.

Datasets. Two full-information datasets were used in [20] to generate partial-information counterparts using click simulation:

- *German Credit Data* is a dataset commonly used for studying algorithmic fairness. It contains information about 1000 loan applicants, each described by a set of attributes and labeled as creditworthy or non-creditworthy. Two groups are defined by the binary feature A43, indicating the purpose of the loan applicant. The ratio of applicants between the two groups is around 8 : 2.
- *Microsoft Learn to Rank (MSLR)* is a standard benchmark dataset for LTR, containing a large number of queries from Bing with manually-judged relevance labels for retrieved web pages. The QualityScore attribute (feature id 133) is used to define binary protected groups using the 40th percentile as threshold as in [20]. For evaluating fairness between $k > 2$ groups (multi-group fairness), k evenly-spaced quantiles define the thresholds.

Following the experimental settings of [17, 20], all datasets are constructed to contain item lists of size 20 for each query. The German Credit and MSLR training sets consist of 100k and 120k sample queries, respectively while full-information test sets consist of 1500 and 4000 samples. Additionally, as reported in Appendix B, much smaller training sets result in analogous performance.

The reader is referred to [20] for details of the click simulation used to produce the training and validation sets.

Models and Hyperparameters. The prediction of item scores is the same for each model, with a single neural network which acts at the level of individual feature vectors as described in Section 5.1. The size of each layer is half that of the previous, and the output is a scalar value representing an item score. Hyperparameters were selected as the best-performing on average among those listed in Table 3, Appendix B.2). Final hyperparameters for each model are as stated also in Table 3, and Adam optimizer is used throughout.

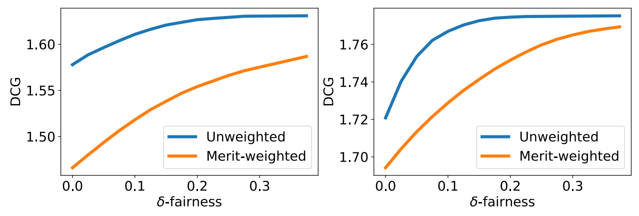

Figure 2: Fairness-Utility tradeoff for unweighted and merit-weighted fairness on credit (left) and MSLR (right) datasets.

The special *fairness parameters*, while also hyperparameters, are treated differently. Recall that fair LTR systems often aim to offer a tradeoff between utility and group fairness, so that fairness can be reduced by an acceptable tolerance in exchange for increased utility. For the baseline methods FULTR and DELTR, this tradeoff is controlled indirectly through the constraint violation penalty term denoted λ, as described in Section 5. Higher values of λ correspond to a preference for stronger adherence to fairness. In order to achieve δ-fairness for some specified δ, many values of λ must be searched until a trained model satisfying δ-fairness is found. As described in Section 5, this approach is unwieldly. In the case of SPOFR, the acceptable violation magnitude δ can be directly specified as in Definition (1).

The performance of each method is reported on the full-information test set for which all relevance labels are known. Ranking utility and fairness are measured with average DCG (Equation (1)) and fairness violation (Equation (3)), where each metric is computed on average over the entire dataset. The position bias power $p = 1$, so that $v_j = 1/(1+j)$ when computing fairness disparity.

Fairness-Utility Tradeoff for Two Groups. The analysis first focuses on experiments involving two protected groups. Figure 2 shows the average test DCG attained by SPOFR on both the German Credit and MSLR datasets, for each level of both unweighted and merit-weighted δ-fairness as input to the model. Each result comes from a model trained with final hyperparameters as shown in Table 3. Recall that each value of δ (defined as in Definition 1) on the x-axis is guaranteed to bound the ranking policy's expected fairness violation in response to *each* query. Note the clear trend showing an increase in utility with the relaxation of the fairness bound δ, for all metrics and datasets. Note also that, in the datasets studied here, average merit favors the majority group. Merit-weighted group fairness can thus constrain the ranking positions of the minority group items further down than in unweighted fairness, regardless of their individual relevance scores, leading to more restricted (thus with lower utility) policies than in the unweighted case.

Fairness Parameter Search. Figure 3 shows the average DCG vs average fairness disparity over the *test set* due to SPOFR, and compares it with those attained by FULTR and DELTR. Each point represents the performance of a single trained model, taken from a grid-search over *fairness parameters* δ (for SPOFR) and λ (for FULTR and DELTR) between the minimum and maximum values in Table 3. Darker colors represent more restrictive fairness parameters in each case. Non-fairness hyperparameters take on the final values

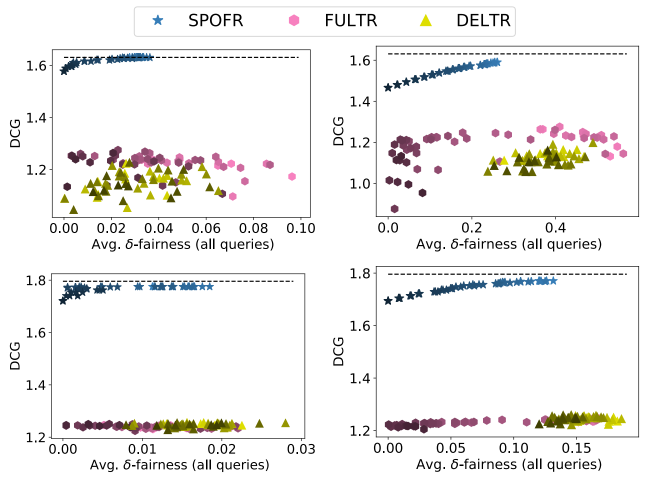

Figure 3: Fairness-Utility tradeoff for unweighted (left) and merit-weighted (right) fairness on credit (top) and MSLR (bottom) datasets.

shown also in Table 3, and each model specification is repeated with 3 random seeds. Note that points on the grid which are lower on the x-axis and higher on the y-axis represent results which are strictly superior relative to others, as they represent a larger utility for smaller fairness violations. Dashed lines represent the maximum utility attainable in each case, computed by averaging over the test set the maximum possible DCG associated to each relevance vector.

Observe that the expected fairness violations due to SPOFR are much lower than the fairness levels guaranteed by the listed fairness parameters δ. This is because δ is a bound on the worst-case violation of fairness associated with any query, but actual resulting fairness disparities are typically much lower on average.

Second, the figure shows that SPOFR attains a substantial improvement in utility over the baselines, while exhibiting more consistent results across independently trained models. Note the dashed line represents the theoretical maximum attainable utility; remarkably, as the fairness parameter is relaxed, the DCG attained by SPOFR converges very close to this value. Section 8 provides theoretical motivation to explain these marked improvements.

Finally, notice that for FULTR and DELTR, large λ values (darker colors) should be associated with smaller fairness violations, compared to models trained with smaller λ values (lighter colors). However, this trend is not consistently observable: These results show the challenge to attain a meaningful relationship between the fairness penalizers λ and the fairness violations in these fair LTR methods. A similar observation also pertains to utility; It is expected that more permissive models in terms of fairness would attain larger utilities; this trend is not consistent in the FULTR and DELTR models. In contrast, the ability of the models learned by SPOFR to *guarantee* satisfying the desired fairness violation equips the resulting LTR models with much more interpretable and consistent outcomes.

Query-level Guarantees. As discussed in Section 5, current fair LTR methods apply a fairness violation term on average over all training samples. Thus, disparities in favor of one group can cancel out those in favor of another group leading to *individual* policies that may not satisfy a desired fairness level. This section illustrates on these behaviors and analyzes the fairness guarantees attained by each model compared at the level of each individual query.

The results are summarized in Figure 7 which compares, SPOFR with FULTR (top) and SPOFR with DELTR (bottom). Each bar represents the maximum expected DCG attained while guaranteeing δ-fairness at the query level for δ as shown on the x-axis. Since neither baseline method can satisfy δ-fairness for every query, confidence levels are shown which correspond to the percentage of queries within the test set that resulted in ranking policies that satisfy *delta*-fairness. If no bar is shown at some fairness level, it was satisfied by no model at the given confidence level.

Notably, SPOFR satisfies δ-fairness with 100 percent confidence while also surpassing the baseline methods in terms of expected utility. This is remarkable and is due partly to the fact that the baseline methods can only be specified to optimize for fairness *on average* over all queries, which accomodates large query-level fairness disparities when they are balanced in opposite directions; i.e., in favor of opposite groups. In contrast SPOFR guarantees the specified fairness violation to be attained for ranking policies associated with each individual query.

Multi-group Fairness. Finally, Figure 5 shows the fairness-utility tradeoff curves attained by SPOFR for each number of groups between 2 and 7 on the MSLR dataset. Note the decrease in expected DCG as the number of groups increases. This is not necessarily due to a degradation in predictive capability from SPOFR; the expected utility of any ranking policy necessarily decreases as fairness constraints are added. In fact, the expected utility converges for each multi-group model as the allowed fairness gap increases. Because this strict notion of multi-group fairness in LTR is uniquely possible using SPOFR, no results from prior approaches are available for direct comparison.

8 DISCUSSION

Theoretical Remarks. This section provides theoretical intuitions to explain the strong performance of SPOFR. As direct outputs of a linear programming solver, the ranking policy returned by SPOFR are subject to the properties of LP optimal solutions. This allows for certain insights on the representative capacity of the ranking model and on the properties of its resulting ranking policies. Let predicted scores be said to be *regret-optimal* if their resulting policy induces zero regret, i.e, $\boldsymbol{y}^\top \Pi^*(\boldsymbol{y})\, \boldsymbol{v} - \boldsymbol{y}^\top \Pi^*(\hat{\boldsymbol{y}})\, \boldsymbol{w} = 0$. That is equivalent to the maximization of the empirical utility.

Theorem 1 (Optimal Policy Prediction). *For any given ground-truth relevance scores* $\boldsymbol{y}$*, there exist predicted item scores which maximize the empirical utility relative to* $\boldsymbol{y}$.

Proof. It suffices to predict $\hat{\boldsymbol{y}} = \boldsymbol{y}$. These scores are regret-optimal by definition, thus maximizing empirical utility. □

Note that the above property is due to the alignment between the structured policy prediction of SPOFR and the evaluation metrics, and is not shared by prior fair learning to rank frameworks.

Next, recall that any linear programming problem has a finite number of feasible solutions whose objective values are distinct

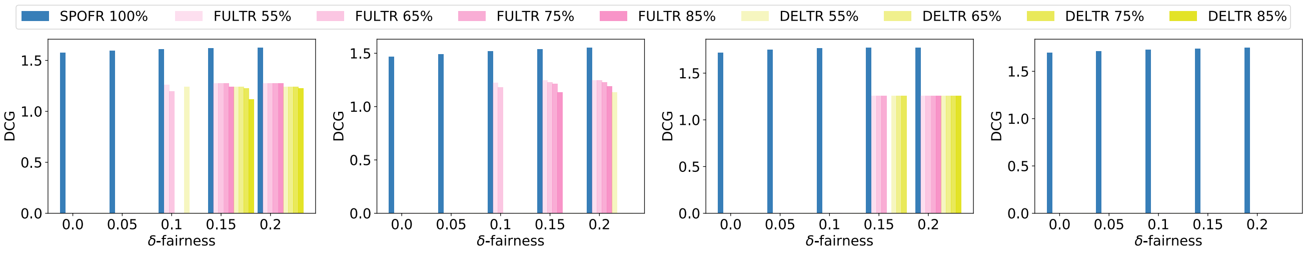

Figure 4: Query level guarantees: German credit unweighted (1st column) and merit-weighted fairness (2nd column); MSLR unweighted (3rd column) and merit-weighted fairness (4th column).

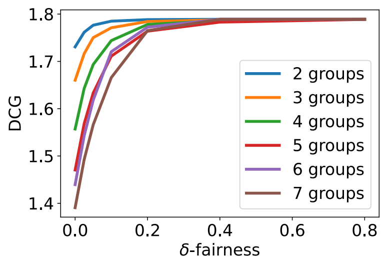

Figure 5: Multigroup Fairness on MSLR 120k.

[3]. *There is thus not only a single point but a region of* $\hat{\boldsymbol{y}}$ *which are regret-optimal under* $\boldsymbol{y}$*, with respect to any instance of Model 1.* This important property eases the difficulty in finding model parameters which maximize the empirical utility for any input sample, as item scores *do not* need to be predicted precisely in order to do so.

Finally, the set of $\hat{\boldsymbol{y}}$ which minimize the regret with respect to any instance of Model 1 overlaps (has nonempty intersection) with the set of $\hat{\boldsymbol{y}}$ which minimize the regret with respect to any other instance of the model, regardless of fairness constraints, under the same ground-truth $\boldsymbol{y}$. To show this, it suffices to exhibit a value of $\hat{\boldsymbol{y}}$ which minimizes the respective regret in every possible instance of Model 1, namely $\boldsymbol{y}$:

$$\boldsymbol{y}^\top \Pi^*_{f_1}(\boldsymbol{y})\boldsymbol{w} - \boldsymbol{y}^\top \Pi^*_{f_1}(\boldsymbol{y})\boldsymbol{w} = 0 = \boldsymbol{y}^\top \Pi^*_{f_2}(\boldsymbol{y})\boldsymbol{w} - \boldsymbol{y}^\top \Pi^*_{f_2}(\boldsymbol{y})\boldsymbol{w}, \quad (16)$$

where $\Pi^*_{f_1}(y)$ and $\Pi^*_{f_2}(y)$ are the optimal policies subject to distinct fairness constraints f_1 and f_2. This implies that a model which learns to rank fairly under this framework need not account for the group composition of item lists in order to maximize empirical utility; *It suffices to learn item scores from independent feature vectors, rather than learn the higher-level semantics of feature vector lists required to enforce fairness.* This is because group fairness is ensured automatically by the embedded optimization model.

The empirical results presented, additionally, show that the utility attained by SPOFR is close to optimal on the test cases analyzed. Together with the theoretical observations above, this suggests that, on the test cases analyzed, fairness does not change drastically the objective of the optimal ranking policy. This may be an artifact of the LTR tasks, obtained from [20], being relatively easy predict given a sufficiently powerful model. These observation may signal a need for the study and curation of more challenging benchmark datasets for future research on fairness in LTR.

SPOFR Limitations. The primary disadvantage of SPOFR is that it cannot be expected to learn to rank lists of arbitrary size, as runtime increases with the size of the lists to be ranked. In contrast to penalty-based methods, which require a single linear pass to the neural network to derive a ranking policy for a given query, SPOFR requires solving a linear programming problem to attain an optimal ranking policy. While this is inevitably computationally more expensive, solving the LP of Model 1 requires low degree polynomial time in the number of items to rank [18], due to the sparsity of its constraints. Fortunately, this issue can be vastly alleviated with the application of hot-starting schemes [14], since the SPO framework relies on iteratively updating a stored solution to each LP instance for slightly different objective coefficients as model weights are updated. *Thus, each instance of Model 1 need not be solved from scratch.* Appendix A reports a detailed discussion on the steps taken in this work to render the proposed model both computationally and memory efficient.

9 CONCLUSIONS

This paper has described SPOFR, a framework for learning fair ranking functions by integrating constrained optimization with deep learning. By enforcing fairness constraints on its ranking policies at the level of each prediction, this approach provides direct control over the allowed disparity between groups, which is guaranteed to hold for every user query. Since the framework naturally accommodates the imposition of many constraints, it generalizes to multigroup fair LTR settings without substantial degradation in performance, while allowing for stronger notions of multigroup fairness than previously possible. Further, it has been shown to outperform previous approaches in terms of both the expected fairness and utility of its learned ranking policies. By integrating constrained optimization algorithms into its fair ranking function, SPOFR allows for analytical representation of expected utility metrics and end-to-end training for their optimization, along with theoretical insights into properties of its learned representations. These advantages may highlight the integration of constrained optimization and machine learning techniques as a promising avenue to address further modeling challenges in future research on learning to rank.

ACKNOWLEDGEMENTS

This research is partially supported by NSF grants 2007164, 2133169, and 2112533, and by CUSE grant II-37-2021. Its views and conclusions are those of the authors only.

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. 2018. A Reductions Approach to Fair Classification. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[2] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating Position Bias without Intrusive Interventions. In *WSDM*. 474–482.

[3] Mokhtar S Bazaraa, John J Jarvis, and Hanis D Sherali. 2008. *Linear programming and network flows*. John Wiley & Sons.

[4] Garrett Birkhoff. 1940. *Lattice theory*. Vol. 25. American Mathematical Soc.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[6] L Elisa Celis, Damian Straszak, and Nisheeth K Vishnoi. 2017. Ranking with fairness constraints. *arXiv preprint arXiv:1704.06840* (2017).

[7] Shady Elbassuoni, Sihem Amer-Yahia, Ahmad Ghizzawi, and Christine Atie. 2019. Exploring fairness of ranking in online job marketplaces. In *22nd International Conference on Extending Database Technology (EDBT)*.

[8] Adam N Elmachtoub and Paul Grigas. 2021. Smart "predict, then optimize". *Management Science* (2021).

[9] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. 2019. Intervention Harvesting for Context-Dependent Examination-Bias Estimation. In *SIGIR*. 825–834.

[10] Ferdinando Fioretto, Pascal Van Hentenryck, Terrence W. K. Mak, Cuong Tran, Federico Baldo, and Michele Lombardi. 2020. Lagrangian Duality for Constrained Deep Learning. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD*, Vol. 12461. Springer, 118–135.

[11] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*. 781–789.

[12] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. 2021. End-to-End Constrained Optimization Learning: A Survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. 4475–4482. https://doi.org/10.24963/ijcai.2021/610

[13] Heyuan Liu and Paul Grigas. 2021. Risk bounds and calibration for a smart predict-then-optimize method. *arXiv preprint arXiv:2108.08887* (2021).

[14] Jayanta Mandi, Peter J Stuckey, Tias Guns, et al. 2020. Smart predict-and-optimize for hard combinatorial optimization problems. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 34. 1603–1610.

[15]]ortools Laurent Perron and Vincent Furnon. [n. d.]. *OR-Tools*. Google. https://developers.google.com/optimization/

[16] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2219–2228.

[17] Ashudeep Singh and Thorsten Joachims. 2019. Policy learning for fairness in ranking. *arXiv preprint arXiv:1902.04056* (2019).

[18] Jan van den Brand. 2020. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 259–278.

[19] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3 (1992), 229–256.

[20] Himank Yadav, Zhengxiao Du, and Thorsten Joachims. 2021. Policy-Gradient Training of Fair and Unbiased Ranking Functions. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1044–1053.

[21] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. 2017. Fa*ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1569–1578.

[22] Meike Zehlike and Carlos Castillo. 2020. Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of The Web Conference 2020*. 2849–2855.

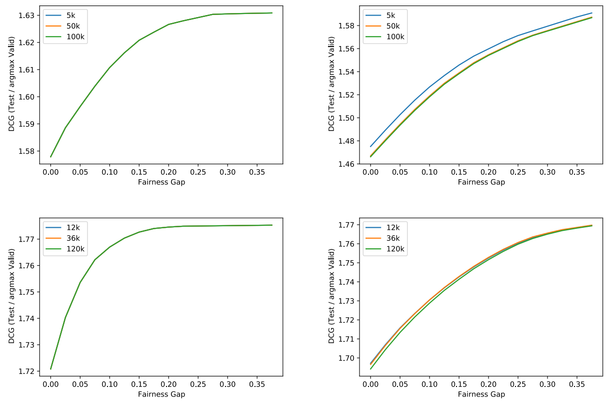

Figure 6: Top left: German Credit Unweighted Fairness; Top right: German Credit Merit-Weighted Fairness; Bottom left: MSLR Unweighted Fairness; Bottom right: MSLR Merit-Weighted Fairness. Average DCG vs allowed fairness gap, SPOFR on datasets of 3 different sizes

A SPOFR: IMPLEMENTATION DETAILS AND EFFICIENCY

For the implementation of SPOFR used to produce the experiments, the linear programming solver of Google OR-Tools was used [?]. The algorithm has worst-case complexity that is exponential with respect to the size of the problem, but is well known to be extremely efficient in the average case [3]. From an efficiency standpoint, there are three steps which must carried out to solve one instance of LP: **(1)** Instantiation of the data structures through the solver API **(2)** Finding a basic feasible solution to the LP problem **(3)** Finding the optimal solution, given a basic feasible solution.

A straightforward implementation that carries out all three steps is inefficient and can lead to long training times. Fortunately, steps **(1)** and **(2)** can be avoided by instantiating Model 1 only once for each distinct (combination of) fairness constraints, and storing the respective solver states in memory. These constraints depend only on the group identities of the items to be ranked. Each time a query is encountered during training, the solving of Model 1 can be resumed beginning with the solution found to the last instance sharing the same group composition. Using this solution as a hot start, only step **(3)** is required to find the optimal policy and complete the forward pass.

For a list of length n, the number of distinct possible fairness constraints in the case of 2 groups is 2^n, which leads to unreasonable memory requirements for storing each required solver state. However, simple manipulations can be used to carry out the required calculations with only n solvers held in memory, by exploiting symmetry. The group identities within an item list take the form of a binary vector G of length n. Once sorted, there are only n such distinct binary vectors.

For an input sample (x_q) and corresponding group identity vector G, let $I_{sort} = argsort(G)$. The fairness constraints in Model 1 are then formulated based on the sorted $G' = G[I_{sort}]$.

Dataset size		**Models**		
		SPOFR	SPOFR*	FULTR
5k	Time per Epoch (s)	425	39	5
	Epochs to Convergence	5	5	29
50k	Time per Epoch (s)	2201	202	28
	Epochs to Convergence	1	1	18
100k	Time per Epoch (s)	4338	398	60
	Epochs to Convergence	1	1	18

Table 2: Runtime comparison

Let C_{ij} be the objective coefficients corresponding to Π_{ij} in Model 1. The rows of C (corresponding to individual items) are permuted by the sorting indices of G:

$$C' = C[I_{sort}][:].$$

Model 1 is then solved using G' and C' in place of G and C. The resulting optimal policy Π' then need only be reverse-permuted in its rows to restore the original orders with respect to items:

$$\Pi = \Pi'[argsort(I_{sort})][:].$$

While the number of variables in Model 1 increases quadratically with the size of the list to be ranked, its linear form ensures that it can scale efficiently to handle lists of reasonable size. Linear programming problems are routinely solved with millions of variables [3], and additionally are well-suited to benefit from hot-starting, when solutions to similar problem instances are known. Fortunately, the Smart Predict-and-Optimize framework is particularly amenable to hot-starting. Since a LP instance for each data sample must be solved at each epoch, a feasible solution to each LP is available from the previous epoch, corresponding to the same constraints and a cost vector which changes based on updates to the DNN model parameters during training [14]. Storing a hot-start solution to each LP instance in a training set requires memory no larger than that of training set, and as the model weights converge, these hot-starts are expected to be very close to the optimal policies for each LP. The implementation described in this paper does not optimize the use of hot-starts since the use of the cached models, as described in this section, already provided large speedups, rendering training times required for SPOFR to replicate the benchmark evaluations of previous works very low.

SPOFR and its improved implementation (SPOFR*) as described above are compared to FULTR with respect to runtime on German Credit datasets. Table 2 records the number of epochs required to converge on average over the hyperparameter search, along with average computation time per epoch. While DELTR reaches convergence relatively efficiently, it cannot produce competitive fair ranking results as shown in Section 7; thus only SPOFR and FULTR are compared. Runtimes are reported as observed Intel(R) Xeon(R) Platinum 8260 CPU @ 2.40GHz.

B EXPERIMENTAL SETTING

B.1 Partial-Information Setting

In the Full-Information setting, all relevance scores are typically elicited via expert judgments, which may be infeasible to obtain. A more practical and common setting is to utilize the implicit feedback

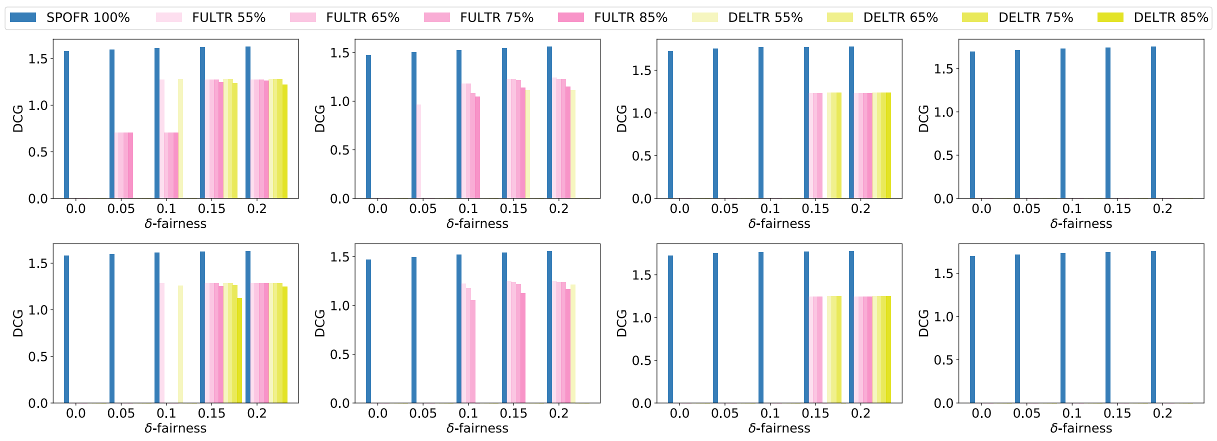

Figure 7: Query level guarantees: German credit ($5K$ top row and $50K$ bottom row) unweighted (1st col.) and merit-weighted fairness (2nd col.); MSLR ($12K$ top row and $36K$ bottom row) unweighted (3rd col.) and merit-weighted fairness (4th col.).

from users (e.g., clicks, views) collected in an existing LTR system as alternative relevance signals. However, this implicit feedback is biased and cannot be directly aligned with true relevance. Note that for a given query q, a user click is observed exclusively if the user examined an item i and the item was found relevant by the underlying LTR model. Let c_i and o_i be a binary variables denoting, respectively, whether the item i was clicked and whether it was examined by the user; Then, $c_i = o_i \cdot y_i$ holds. This is called the *Partial-Information* setting [11].

Of course, it is not appropriate to directly replace labels y_i with c_i in Equation (1). To learn an unbiased ranking policy in the Partial-Information setting, the implicit feedback data must be *debiased*. A widely adopted method is to use *Inverse Propensity Scoring (IPS)*, which introduces the "propensity" $p(o_i = 1)$ of when the implicit feedback was logged. The propensity can be modelled in various ways [9, 11]. The most common one is the position-based examination model, where the propensity depends on only the position of item i in the ranking when the click was logged. This means $p(o_i = 1) = v_{\sigma(i)}$, where v_k denotes the examination probability at position k. These position biases v_k can be estimated with swap experiments [11] or intervention harvesting [2]. With knowledge of the propensity, the unbiased estimator for Δ becomes [11]:

$$\widehat{\Delta}(\sigma, c) = \sum_{i:c_i=1} \frac{\Delta\left(\sigma, y_q\right)}{p(o_i = 1|\sigma)}. \tag{17}$$

The estimator is unbiased if all propensities are bounded away from zero [11]. Replacing $\Delta(\sigma, y_q)$ in Equation (1) with $\widehat{\Delta}(\sigma, c)$, leads to the unbiased utility estimator used to learn from implicit feedback in the Partial-Information setting.

B.2 Hyper-parameters

The prediction of item scores is the same for each model, with a single neural network which acts at the level of individual feature vectors as described in Section 5.1. The number of layers in each case is the maximum possible when each subsequent layer is halved in size; this depends on the length of a dataset's item feature vectors, which constitute the model inputs. Hyperparameters were selected as the best-performing on average among those listed in Table B.2). Final hyperparameters for each model are as stated also in Table 3, and Adam optimizer is used in the production of each result. Asterisks (*) indicate that there is no option for a final value, as all values of each parameter are of interest in the analysis of fairness-utility tradeoff, as reported in the experimental setting Section.

Hyperparameter	Min	Max	Final Value		
			SPOFR	FULTR	DELTR
learning rate	$1e^{-6}$	$1e^{-3}$	$\mathbf{1e^{-5}}$	$\mathbf{2.5e^{-4}}$	$\mathbf{2.5e^{-4}}$
violation penalty λ	0	100	**N/A**	*	*
allowed violation δ	0	0.4	*	**N/A**	**N/A**
entropy regularization decay	0.1	0.5	**N/A**	**0.3**	**N/A**
batch size	4	64	**64**	**16**	**16**

Table 3: Hyperparameters

B.3 Additional Experiments

Additional results are provided which investigate the effect of dataset size on the performance of SPOFR. For German Credit data, sizes $5k$, $50k$ and $100k$ are used along with $12k$, $36k$ and $120k$ for MSLR data. In both cases, the size of a dataset represents the number of training 'clicks' gathered by the click simulation. Figure 6 indicates that the size of the training set does not affect test accuracy at convergence. In the case of merit-weighted fairness, the slight divergence in utility can be attributed to the difference in relative merit calculated on each differently-sized dataset. Figure 7 shows that both baseline methods benefit in terms of query-level fairness from increased dataset size, but the effect on SPOFR is negligible. Note also that FULTR reports an increase in accuracy as training samples are added [20]. This may indicate an advantage in terms of data-efficiency attributable to SPOFR.