

# END-TO-END LEARNING FOR FAIR MULTIOBJECTIVE OPTIMIZATION UNDER UNCERTAINTY


A PREPRINT

 **My H. Dinh**

University of Virginia  
Charlottesville, VA, USA  
fqw2tz@virginia.edu

 **James Kotary**

University of Virginia  
Charlottesville, VA, USA  
jk4pn@virginia.edu

 **Ferdinando Fioretto**

University of Virginia  
Charlottesville, VA, USA  
fioretto@virginia.edu

February 13, 2024

## ABSTRACT

Many decision processes in artificial intelligence and operations research are modeled by parametric optimization problems whose defining parameters are unknown and must be inferred from observable data. The Predict-Then-Optimize (PtO) paradigm in machine learning aims to maximize downstream decision quality by training the parametric inference model end-to-end with the subsequent constrained optimization. This requires backpropagation through the optimization problem using approximation techniques specific to the problem’s form, especially for nondifferentiable linear and mixed-integer programs. This paper extends the PtO methodology to optimization problems with nondifferentiable Ordered Weighted Averaging (OWA) objectives, known for their ability to ensure properties of fairness and robustness in decision models. Through a collection of training techniques and proposed application settings, it shows how optimization of OWA functions can be effectively integrated with parametric prediction for fair and robust optimization under uncertainty.

**Keywords** Predict-then-Optimize · Multi-Objectives Optimization · Fairness

## 1 Introduction

The *Predict-Then-Optimize* (PtO) framework models decision-making processes as optimization problems with unspecified parameters  $c$ , which must be estimated by a machine learning (ML) model, given correlated features  $z$ . An estimation of  $c$  completes the problem’s specification, whose solution defines a mapping:

$$x^*(c) = \operatorname{argmax}_{x \in \mathcal{S}} f(x, c) \quad (1)$$

The goal is to learn a model  $\hat{c} = \mathcal{M}_\theta(z)$  from observable features  $z$ , such that the objective value  $f(x^*(\hat{c}), c)$  under ground-truth parameters  $c$  is maximized on average.

This setting is common to many real-world applications requiring decision-making under uncertainty, such as planning the fastest route through a city with unknown traffic delays, or determining optimal power generation schedules based on demand forecasts. A classic example is the Markowitz portfolio problem, wherein the optimization model (1) may regard  $f$  as the total return due to asset allocations  $x$  under predicted prices  $c$ , while  $\mathcal{S}$  includes constraints on price covariance as a measure of risk [Markowitz, 1991]. Modern approaches are based on *end-to-end learning*, and train  $\hat{c} = \mathcal{M}_\theta(z)$  to maximize  $f(x^*(\hat{c}), c)$  directly as loss function. This requires backpropagation through  $x^*(\hat{c})$ , which is challenging when (1) defines a nondifferentiable mapping, as it will be further elaborated in Section 2.

Within this context, optimizing multiple objectives becomes an important extension, where the decision-making process needs to balance various competing objectives. Of particular interest is the case when such objectives must be *fairly* optimized, as common in many engineering settings including energy systems [Terlouw et al., 2019], urban planning [Salas and Yepes, 2020], and multi-objective portfolio optimization [Iancu and Trichakis, 2014; Chen and Zhou, 2022]. In this setting, a prevalent approach is based on optimization of the scalar aggregation of all objectives by Ordered

Weighted Averaging (OWA) [Yager, 1993]. Such approach results in Pareto-optimal solutions that fairly balance the values of each individual objective. However, employing an OWA optimization within a Predict-Then-Optimize framework is challenged by its nondifferentiability, which prevents backpropagation of its constrained optimization mapping  $\mathbf{x}^*(c)$  within machine learning models trained by gradient descent.

This paper aims to solve this challenge, and enable the combined learning and optimization of new applications such as fair learning-to-rank models based on OWA optimization of rankings, and Markowitz prediction models based on multiscenario portfolio optimization. By leveraging modern techniques in OWA optimization and Predict-Then-Optimize learning, this paper shows how the optimization of OWA functions can be effectively backpropagated in machine learning models, enabling end-to-end trainable prediction and decision models for applications requiring fair and robust decision-making under uncertainty.

**Contributions.** In particular, the paper makes the following contributions: (1) It proposes novel techniques for differentiating OWA optimization models with respect to their uncertain parameters, allowing their integration in end-to-end trainable ML models. (2) It is the first to show how loss functions based on OWA aggregation can be effectively employed for supervision of such end-to-end training. (3) Based on these contributions, it proposes several effective modeling strategies for combining parametric prediction with OWA optimization, and evaluates them on novel application settings in which optimal decisions must be made fair or robust to multiple uncertain objective criteria.

## 2 Preliminaries

Prior to discuss the paper contribution, this section provides an overview of the concepts of optimizing OWA functions and implementing end-to-end training methods for both prediction and optimization.

### 2.1 OWA and its Optimization

The *Ordered Weighted Average* (OWA) operator [Yager, 1993] is a class of functions meant for aggregating multiple independent values, in settings requiring multicriteria evaluation and comparison [Yager and Kacprzyk, 2012]. Let  $\mathbf{y} \in \mathbb{R}^m$  be a vector of  $m$  distinct criteria, and  $\tau : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be the sorting map for which  $\tau(\mathbf{y}) \in \mathbb{R}^m$  holds the elements of  $\mathbf{y}$  in increasing order. Then for any  $\mathbf{w}$  satisfying  $\{\mathbf{w} \in \mathbb{R}^m : \sum_i w_i = 1, \mathbf{w} \geq 0\}$ , the OWA aggregation with weights  $\mathbf{w}$  is defined as a linear functional on  $\tau(\mathbf{y})$ :

$$\text{OWA}_{\mathbf{w}}(\mathbf{y}) = \mathbf{w}^T \tau(\mathbf{y}), \quad (2)$$

which is concave and piecewise-linear in  $\mathbf{y}$  [Ogryczak and Śliwiński, 2003].

**Fair OWA.** Of particular interest are the *Fair OWA*, whose weights have decreasing order:  $w_1 > w_2 \dots > w_n$ .

The following three properties possessed by Fair OWA functions are key to their use in fairly optimizing multiple objectives: (1) Let  $\mathcal{P}_m$  be the set of all permutations of  $[1, \dots, m]$ . *Impartiality* means that Fair OWA treats all criteria equally, in the sense that  $\text{OWA}_{\mathbf{w}}(\mathbf{y}) = \text{OWA}_{\mathbf{w}}(\mathbf{y}_{\sigma})$  for any  $\sigma \in \mathcal{P}_m$ . (2) *Equitability* is the property that marginal transfers from a criterion with higher value to one with lower value results in an increased OWA aggregated value. This condition holds that  $\text{OWA}_{\mathbf{w}}(\mathbf{y}_{\epsilon}) > \text{OWA}_{\mathbf{w}}(\mathbf{y})$ , where  $\mathbf{y}_{\epsilon} = \mathbf{y}$  except at positions  $i$  and  $j$  where  $(\mathbf{y}_{\epsilon})_i = y_i - \epsilon$  and  $(\mathbf{y}_{\epsilon})_j = y_j + \epsilon$ , assuming  $y_i > y_j + \epsilon$ . (3) *Monotonicity* means that  $\text{OWA}_{\mathbf{w}}(\mathbf{y})$  is an increasing function of each element of  $\mathbf{y}$ . The monotonicity property implies that solutions which optimize (2) are Pareto Efficient solutions of the underlying multiobjective problem, thus that no single criteria can be raised without reducing another [Ogryczak and Śliwiński, 2003]. Intuitively, OWA objectives lead to fair optimal solutions by always assigning the highest weights of  $\mathbf{w}$  to the objective criteria in order of lowest current value.

### 2.2 Predict-Then-Optimize Learning

The problem setting of this paper can be viewed within the framework of Predict-Then-Optimize. Generically, a parametric optimization problem (1) models an optimal decision  $\mathbf{x}^*(c)$  with respect to unknown parameters  $c$  within a distribution  $c \sim \mathcal{C}$ . Although the true value of  $c$  is unknown, correlated *feature* values  $\mathbf{z} \sim \mathcal{Z}$  can be observed. The goal is to learn a predictive model  $\mathcal{M}_{\theta} : \mathcal{Z} \rightarrow \mathcal{C}$  from features  $\mathbf{z}$  to estimate problem parameters  $\hat{c} = \mathcal{M}_{\theta}(\mathbf{z})$ , such that the resulting solution's empirical objective value under ground-truth parameters, is maximized. That is,

$$\arg\max_{\theta} \mathbb{E}_{(\mathbf{z}, c) \sim \Omega} f(\mathbf{x}^*(\mathcal{M}_{\theta}(\mathbf{z})), c), \quad (3)$$

where  $\Omega$  represents the joint distribution between  $\mathcal{Z}$  and  $\mathcal{C}$ .

The above training goal is often best realized by maximizing empirical *Decision Quality* as a loss function [Mandi et al., 2023], defined

$$\mathcal{L}_{DQ}(\hat{c}, c) = f(\mathbf{x}^*(\hat{c}), c). \quad (4)$$

Gradient descent training of (3) with  $\mathcal{L}_{DQ}$  requires a model of gradient  $\frac{\partial \mathcal{L}_{DQ}}{\partial \hat{c}}$ , either directly or through chain-rule composition  $\frac{\partial \mathcal{L}_{DQ}}{\partial \hat{c}} = \frac{\partial x^*(\hat{c})}{\partial \hat{c}} \cdot \frac{\partial \mathcal{L}_{DQ}}{\partial x^*}$ . Here, left-multiplication by the Jacobian is equivalent to backpropagation through the optimization mapping  $x^*$ . When  $x^*$  is not differentiable, as in the case of OWA optimizations, smooth approximations are required, such as those developed in the next section.

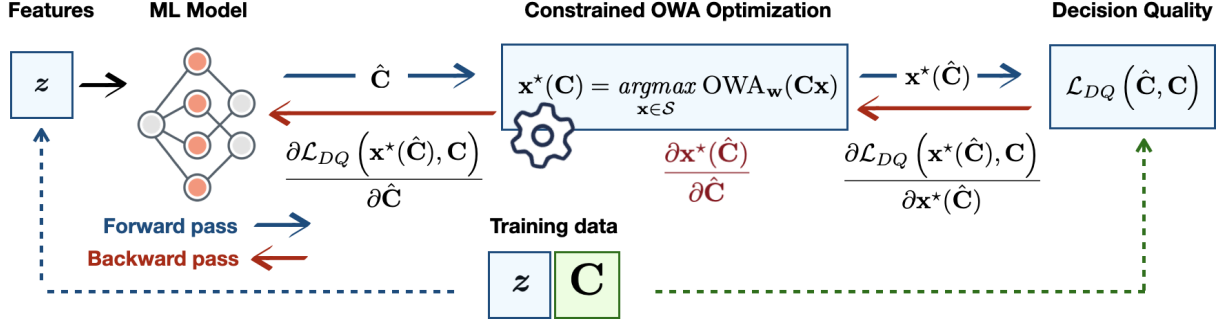


Figure 1: Predict-Then-Optimize for OWA Optimization.

### 3 End-to-End Learning with Fair OWA Optimization

This paper’s proposed methodology and setting are focused on when the objective function  $f$  is composed of an ordered weighted average of  $m$  linear objective functions, each parametrized by one row of a matrix  $C \in \mathbb{R}^{m \times n}$ , so that  $f(x, C) = \text{OWA}_w(Cx)$  and

$$x^*(C) = \argmax_{x \in S} \text{OWA}_w(Cx). \quad (5)$$

Note that the methodology of this paper naturally extends to cases where the OWA objective above is combined with additional smooth objective terms. For simplicity, the exposition is developed primarily in terms of the pure OWA objective as shown in equation (5), wherever applicable.

The goal is to learn a prediction model  $\hat{C} = \mathcal{M}_\theta(z)$  that maximizes decision quality through gradient descent on problem (3), which requires its gradients w.r.t.  $\hat{C}$ :

$$\frac{\partial \mathcal{L}_{DQ}(\hat{C}, C)}{\partial \hat{C}} = \underbrace{\frac{\partial x^*}{\partial \hat{C}}}_J \cdot \underbrace{\frac{\partial \text{OWA}_w(Cx^*)}{\partial x^*}}_g, \quad (6)$$

where  $x^*$  is evaluated at  $\hat{C}$ . The primary strategy for modeling this overall gradient involves initially determining the OWA function gradient  $g$ , followed by computing the product  $Jg$  by backpropagation of  $g$  through  $x^*$ .

While nondifferentiable, the class of OWA functions is *subdifferentiable*, with subgradients as follows:

$$\frac{\partial}{\partial y} \text{OWA}_w(y) = w_{(\sigma^{-1})} \quad (7)$$

where  $\sigma$  are the sorting indices on  $y$  [Do and Usunier, 2022]. Based on this formula, computing an overall subgradient  $g = \partial/\partial x \text{OWA}_w(Cx)$  is a routine application of the chain rule (via automatic differentiation). While the subgradients (7) have been used in OWA optimization, this is the first work which demonstrates their use in training ML models. A schematic illustration highlighting the forward and backward steps required for this process is provided in Figure 1.

As outlined next, the main technical contribution of the paper is to propose differentiable models of OWA optimization (5), through which backpropagation of  $g$  can effectively approximate the decision quality gradient  $Jg$  for end-to-end training of (3). The following sections propose alternative models of differentiable OWA optimization, each tailored to address problem-specific technical challenges.

First, Section 4 shows how the OWA optimization (5) with continuous variables can be effectively smoothed, to yield differentiable approximations that can be backpropagated in end-to-end training (3). Then, Section 5 focuses on a special form of optimization mapping with nonparametric OWA term but an additional parametric objective term, and shows how its backpropagation can be implemented using only a blackbox solver of the underlying problem, without smoothing. Finally Section 6 outlines a method of surrogate solvers, focusing on cases where OWA-aggregation of objectives renders its optimization too difficult to solve.

## 4 Differentiable Approximate OWA Optimization

This section develops two alternative differentiable approximations of the OWA optimization mapping (5). Prior works [Wilder et al., 2019b, Amos et al., 2019] show that when an optimization mapping (1) is discontinuous, as is the case when  $f$  and  $\mathcal{S}$  define a linear program (LP), differentiable approximations to (1) can be formed by regularization of its objective by smooth functions. Section 4.1 will show how linear programming models of OWA optimization can be combined with smoothing techniques for LP, which perform well as differentiable approximations of (5).

However, this model is shown to become computationally intractable for more than a few criteria  $m$ . An efficient alternative is proposed in Section 4.2, in which the mapping (5) is made differentiable by replacing the OWA objective with its smooth Moreau envelope approximation. To the best of the authors knowledge, this is the first time that objective smoothing via the Moreau envelope is used (and shown to be an effective technique) for approximating nondifferentiable optimization programs in end-to-end learning. As approximations of the true mapping (5), both smoothed models are used employed in training and replaced by (5) at test time, similarly to a softmax layer in classification.

### 4.1 OWA LP with Quadratic Smoothing

The mainstay approach to solving problem (5), when  $x \in \mathcal{S}$  is linear, is to transform the problem into a linear program without OWA functions, and solve it with a simplex method [Ogryczak and Śliwiński, 2003]. Our first approach to differentiable OWA optimization combines this transformation with the smoothing technique of [Wilder et al., 2019b], which forms differentiable approximations to linear programs

$$x^*(c) = \operatorname{argmax}_{Ax \leq b} c^T x \quad (8)$$

by adding a scaled euclidean norm term  $\epsilon \|x\|^2$  to the objective function, resulting in a continuous mapping  $x^*(c) = \operatorname{argmax}_{Ax \leq b} c^T x + \epsilon \|x\|^2$ , a quadratic program (QP) which can be differentiated implicitly via its KKT conditions as in [Amos and Kolter, 2017].

We adapt a version of this technique to OWA optimization (5) by first forming an equivalent LP problem. It is observed in [Ogryczak and Śliwiński, 2003] that  $\text{OWA}_w$  can be expressed as the minimum weighted average resulting among all permutations of the OWA weights  $w$ :

$$\text{OWA}_w(r) = \max_z z \quad \text{s.t.} \quad z \leq w_\sigma \cdot r, \quad \forall \sigma \in \mathcal{P}, \quad (9)$$

which allows the OWA optimization (5) to be expressed as

$$x^*(C) = \operatorname{argmax}_{x \in \mathcal{S}, y, z} z \quad (10a)$$

$$\text{s.t.: } y = Cx \quad (10b)$$

$$z \leq w_\tau \cdot y \quad \forall \tau \in \mathcal{P}_m. \quad (10c)$$

When the constraints  $x \in \mathcal{S}$  are linear, problem (10) is a LP. However, its constraints (10c) grow factorially as  $m!$ , where  $m$  is the number of individual objective criteria aggregated by OWA. Smoothing by the scaled norm of joint variables  $x, y, z$  leads to a differentiable QP approximation, viable when  $m$  is small, which can be solved and differentiated using [Amos and Kolter, 2017] or the generic differentiable optimization solver [Agrawal et al., 2019a]:

$$x^*(C) = \operatorname{argmax}_{x \in \mathcal{S}, y, z} z + \epsilon (\|x\|_2^2 + \|y\|_2^2 + z^2) \quad (11a)$$

$$\text{subject to: } (10b), (10c). \quad (11b)$$

While problem (10) does not fit the exact form (8) due to its parameterized constraints (10b), the need for quadratic smoothing (11a) is illustrated experimentally in Section 7.1.1. The main *disadvantage* of this method is poor scalability in the number of criteria  $m$ , due to constraints (10c).

### 4.2 Moreau Envelope Smoothing

In light of the efficiency challenges faced by (11), we propose an alternative smoothing technique to form more scalable differentiable approximations of the optimization mapping (5). Rather than adding a quadratic term as in (11), we replace the piecewise linear function  $\text{OWA}_w$  in (5) by its Moreau envelope, defined for generic  $f$  as:

$$f^\beta(x) = \min_v f(v) + \frac{1}{2\beta} \|v - x\|^2. \quad (12)$$

Compared to its underlying function  $f$ , the Moreau envelope is  $\frac{1}{\beta}$  smooth while sharing the same optima [Beck, 2017]. The Moreau envelope-smoothed OWA optimization is

$$x^*(C) = \operatorname{argmax}_{x \in \mathcal{S}} \text{OWA}_w^\beta(Cx). \quad (13)$$

With its smooth objective function, problem (13) can be solved by gradient-based optimization methods, such as projected gradient descent, or more likely a Frank-Wolfe method if  $\mathbf{x} \in \mathcal{S}$  is linear (see Section 7.1.1). Additionally, it can be effectively backpropagated in end-to-end learning.

Backpropagation of (13) is nontrivial since its objective function lacks a closed form. To proceed, we first note from [Do and Usunier, 2022] that the gradient of the Moreau envelope  $\text{OWA}_w^\beta$  is equal to a Euclidean projection:

$$\frac{\partial}{\partial \mathbf{x}} \text{OWA}_w^\beta(\mathbf{x}) = \text{proj}_{\mathcal{C}(\tilde{\mathbf{w}})} \left( \frac{\mathbf{x}}{\beta} \right), \quad (14)$$

where  $\tilde{\mathbf{w}} = -(w_m, \dots, w_1)$  and the permutahedron  $\mathcal{C}(\tilde{\mathbf{w}})$  is the convex hull of all permutations of  $\tilde{\mathbf{w}}$ . It's further shown in [Blondel et al., 2020] how such a projection can be computed and differentiated in  $\mathcal{O}(m \log m)$  time using isotonic regression. To leverage the differentiable gradient function (14) for backpropagation of the smoothed optimization (13), we model its Jacobian by differentiating the fixed-point conditions of a gradient-based solver.

Letting  $\mathcal{U}(\mathbf{x}, \mathbf{C}) = \text{proj}_{\mathcal{S}}(\mathbf{x} - \alpha \cdot \frac{\partial}{\partial \mathbf{x}} \text{OWA}_w^\beta(\mathbf{x}, \mathbf{C}))$ , a projected gradient descent step on (13) is  $\mathbf{x}^{k+1} = \mathcal{U}(\mathbf{x}^k, \mathbf{C})$ . Differentiating the fixed-point conditions of convergence where  $\mathbf{x}^k = \mathbf{x}^{k+1} = \mathbf{x}^*$ , and rearranging terms yields a linear system for  $\frac{\partial \mathbf{x}^*}{\partial \mathbf{C}}$ :

$$\left( I - \underbrace{\frac{\partial \mathcal{U}(\mathbf{x}^*, \mathbf{C})}{\partial \mathbf{x}^*}}_{\Phi} \right) \frac{\partial \mathbf{x}^*}{\partial \mathbf{C}} = \underbrace{\frac{\partial \mathcal{U}(\mathbf{x}^*, \mathbf{C})}{\partial \mathbf{C}}}_{\Psi} \quad (15)$$

The partial Jacobian matrices  $\Phi$  and  $\Psi$  above can be found given a differentiable implementation of  $\mathcal{U}$ . This is achieved by computing the inner gradient  $\frac{\partial}{\partial \mathbf{x}} \text{OWA}_w^\beta(\mathbf{x}, \mathbf{C})$  via the differentiable permutahedral projection (14), and solving the outer projection mapping  $\text{proj}_{\mathcal{S}}$  using a generic differentiable solver such as `cvxpy` [Agrawal et al., 2019a]. As such, applying  $\mathcal{U}$  at a precomputed solution  $\mathbf{x}^*(\mathbf{C})$  allows  $\Phi$  and  $\Psi$  to be extracted in PyTorch, in order to solve (15); this process is efficiently implemented via the `fold-opt` library [Kotary et al., 2023].

## 5 Blackbox Methods for Nonparametric OWA Objective

This section proposes a special class of techniques for cases where the OWA term of an objective function is specified with *known* coefficients  $\mathbf{B} \in \mathbb{R}^{m \times n}$ , and uncertainty lies instead in an additional parametrized *linear* objective term:

$$\mathbf{x}^*(\mathbf{c}) = \arg\max_{\mathbf{x} \in \mathcal{S}} \mathbf{c}^T \mathbf{x} + \lambda \text{OWA}_w(\mathbf{B}\mathbf{x}). \quad (16)$$

This form is taken by the optimization mapping within the fair learning to rank model proposed in Section 7.2. Appealing again to the reformulation (10) leads (16) to become

$$(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)(\mathbf{c}) = \arg\max_{\mathbf{x} \in \mathcal{S}, \mathbf{y}, \mathbf{z}} \mathbf{c}^T \mathbf{x} + \lambda \mathbf{z} \quad (17a)$$

$$\text{s.t.: } \mathbf{y} = \mathbf{B}\mathbf{x} \quad (17b)$$

$$\mathbf{z} \leq \mathbf{w}_\tau \cdot \mathbf{y} \quad \forall \tau \in \mathcal{P}, \quad (17c)$$

which as discussed in Subsection 4.1 grows intractable with increasing  $m$  since the constraints (17c) number ( $m!$ ).

On the other hand, it fits the particular form  $\mathbf{v}^*(\gamma) = \arg\max_{\mathbf{v} \in \mathcal{C}} \gamma^T \mathbf{v}$ , treated in several works [Elmachtoub and Grigas, 2021, Berthet et al., 2020, Pogančić et al., 2020], wherein an uncertain *linear* objective is paired with *nonparametric* constraints. These works propose differentiable solvers based on *blackbox* solvers of the underlying optimization problem, without smoothing. This is generally accomplished by modeling the gradient as a combination of solutions induced by perturbed input parameters. As shown next, this allows a gradient formula based on (17) to be computed without actually solving (17), given a blackbox solver for the underlying problem (16).

We illustrate the idea using the "Smart Predict-Then-Optimize" scheme [Elmachtoub and Grigas, 2021], which trains to maximize  $\mathcal{L}_{DQ}$  by equivalently minimizing the suboptimality (called *regret*) via a convex subdifferentiable upper bounding function named  $\mathcal{L}_{SPO+}$ . By construction it admits a formula for subgradients directly with respect to  $\hat{\mathbf{c}}$ :

$$\partial / \partial \hat{\gamma} \mathcal{L}_{SPO+}(\hat{\gamma}, \gamma) = \mathbf{v}^*(2\hat{\gamma} - \gamma) - \mathbf{v}^*(\gamma). \quad (18)$$

Given any efficient method which provides optimal solutions  $\mathbf{x}^*(\mathbf{C})$  to the OWA optimization (16), the auxiliary variables of problem (17) can be recovered as  $\mathbf{y}^* = \mathbf{C}\mathbf{x}^*$  and  $\mathbf{z}^* = \text{OWA}_w(\mathbf{y}^*)$ . Defining the variables  $\mathbf{v}^* = (\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$  and noting that  $\gamma = (\mathbf{c}, \mathbf{0}, \lambda)$  in problem (17), its SPO+ loss subgradient can be now computed directly using formula (18). In this way, the problem form (17) is leveraged to derive a *backpropagation model*, while *avoiding its direct solution* as a linear program. Section 7.1.1 will show how this can be applied in combination with an efficient Frank-Wolfe solution of (16) to design a scalable fair learning to rank model.



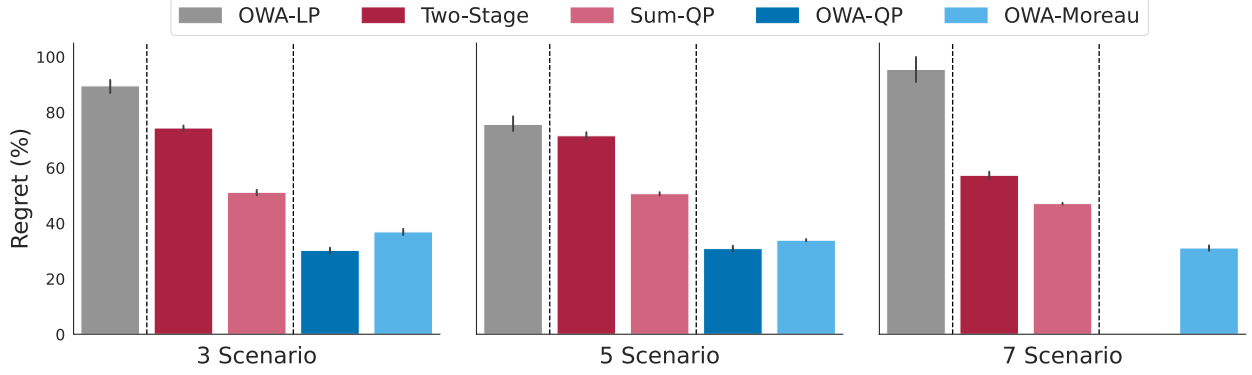


Figure 2: Percentage OWA regret (lower is better) on test set, on robust portfolio problem over 3,5,7 scenarios.

## 6 Differentiable Surrogate Optimization Mappings

OWA optimization problems (5) can be difficult to solve in general, even with modern methods, without exploiting special problem-specific structure. In such cases, an alternative to differentiable approximations of (5), as proposed in Section 4, would be to produce feasible candidate  $\mathbf{x}^* \in \mathcal{S}$  from a simpler differentiable model without OWA objectives.

For example, a *linear* surrogate model proves useful when (5) represents fair OWA optimization of multiple objectives in a linear program (such as shortest path or bipartite matching) which depends on total unimodularity to maintain integral solutions  $\mathbf{x}^* \in \{0, 1\}^n$ :

$$\mathbf{x}^*(\mathbf{c}) = \operatorname{argmax}_{\mathbf{x} \in \mathcal{S}} \mathbf{c}^T \mathbf{x} \quad (19)$$

As illustrated in Section 7.1.2 on a parametric shortest path problem, this surrogate approach is essential to avoid arising an intractable OWA mixed-integer program, since integrality of solutions is guaranteed only under linear objectives.

The main *disadvantage* inherent to the proposed surrogate models is that they do not directly approximate the true OWA problem (5). Thus, the learned model  $\mathcal{M}_\theta(\mathbf{z}) = \hat{\mathbf{c}} \in \mathbb{R}^n$  does not fit the form prescribed in ((3), (5)) as written, and it cannot supply parametric estimates  $\hat{\mathbf{C}} \in \mathbb{R}^{m \times n}$  to an external solver of problem (5). Despite this, using  $\text{OWA}_w(\mathbf{C}\mathbf{x}(\hat{\mathbf{c}}))$  as a loss function trains the surrogate model to learn solutions to (5) with high decision quality.

## 7 Experiments

This section uses the differentiable elements introduced in Sections 4-6 to compose end-to-end trainable prediction and OWA optimization models. Three experimental settings are proposed for their evaluation, across two main application settings. The first application setting is *Fair Multiobjective Prediction and Optimization*, which extends the Predict-Then-Optimize setting of Section 2.2 to cases where multiple uncertain objective functions must be jointly learned and optimized fairly via their OWA aggregation, as per problem (5). Within this setting, *Robust Markowitz Portfolio Optimization* focuses on comparatively evaluating the differentiable approximations proposed in Section 4 against a host of baseline methods. Then, *Multi-Species Warcraft Shortest Path* serves as a case study in which a differentiable surrogate model can enable learning with OWA optimization of integer variables. A second, distinct application setting proposes a *Fair Learning-to-Rank* model, whose OWA-aggregated objectives are known with certainty, corresponding to problem (16) as detailed in Section 5.

### 7.1 Fair Multiobjective Predict-and-Optimize

This setting uses a prediction model  $\hat{\mathbf{C}} = \mathcal{M}_\theta(\mathbf{z})$  to jointly estimate from features  $\mathbf{z}$  the coefficients  $\mathbf{C} \in \mathbb{R}^{m \times n}$  of  $m$  linear objectives, taken together as  $\mathbf{C}\mathbf{x} \in \mathbb{R}^m$ . Its training goal is to maximize empirical decision quality with respect to their Fair OWA aggregation  $f(\mathbf{x}, \mathbf{C}) = \text{OWA}_w(\mathbf{C}\mathbf{x})$ :

$$\mathcal{L}_{DQ}(\hat{\mathbf{C}}, \mathbf{C}) = \text{OWA}_w(\mathbf{C}\mathbf{x}^*(\hat{\mathbf{C}})). \quad (20)$$

Any descending OWA weights  $\mathbf{w}$  can be used to specify (20); we choose the squared *Gini indices*  $w_j = \left(\frac{n-1+j}{n}\right)^2$ .

**Evaluation.** In this section, each model is evaluated on the basis of its ability to train a model  $\hat{C} = \mathcal{M}_\theta(z)$  to attain high decision quality (20) in terms of the OWA-aggregated objective. Results are reported in terms of the equivalent *regret* metric of suboptimality, whose minimum value 0 corresponds to maximum decision quality:

$$\text{regret}(\hat{C}, C) = \text{OWA}_w^*(C) - \text{OWA}_w(Cx^*(\hat{C})) \quad (21)$$

where  $\text{OWA}_w^*(C)$  is the true optimal value of problem (5). This experiment is designed to evaluate the proposed differentiable approximations (11) and (13) of Section 4; for reference they are named *OWA-QP* and *OWA-Moreau*.

**Baseline Models.** In addition to the newly proposed models, the evaluations presented in this section include two main baseline methods: (1) The *two-stage* method is the standard baseline for comparison against proposed methods for Predict-Then-Optimize training (3) [Mandi et al., 2023]. It trains the prediction model  $\hat{C} = \mathcal{M}_\theta(z)$  by MSE regression, minimizing  $\mathcal{L}_{TS}(\hat{C}, C) = \|\hat{C} - C\|^2$  without considering the downstream optimization model, which is employed only at test time. In addition, (2) the *unweighted sum* (UWS) of the objective criteria results in an LP mapping  $x^*(C) = \arg\max_{x \in S} 1^T(Cx)$  which can be employed in end-to-end training by using quadratic smoothing [Wilder et al., 2019b] in 7.1.1 and blackbox differentiation [Pogančić et al., 2020] in 7.1.2; this baseline leverages end-to-end learning but without incorporating the OWA objective.

### 7.1.1 Differentiable OWA Optimization: Robust Markowitz Portfolio Problem

The classic Markowitz portfolio problem is concerned with constructing an optimal investment portfolio, given future returns  $c \in \mathbb{R}^n$  on  $n$  assets, which are unknown and predicted from exogenous data. A common formulation maximizes future returns subject to a risk limit, modeled as a quadratic covariance constraint. Define the set of valid fractional allocations  $\Delta_n = \{x \in \mathbb{R}^n : 1^T x = 1, x \geq 0\}$ , then :

$$x^*(c) = \arg\max_{x \in \Delta_n} c^T x \quad \text{s.t.:} \quad x^T \Sigma x \leq \delta. \quad (22)$$

where  $\Sigma \in \mathbb{R}^{n \times n}$  are the price covariances over  $n$  assets. The optimal portfolio allocation (22) as a function of future returns  $c \in \mathbb{R}^n$  is differentiable using known methods [Agrawal et al., 2019a], and is commonly used in evaluation of Predict-Then-Optimize methods [Mandi et al., 2023].

An alternative approach to risk-aware portfolio optimization views risk in terms of robustness over alternative scenarios. In [Cajas, 2021],  $m$  future price scenarios are modeled by a matrix  $C \in \mathbb{R}^{m \times n}$  whose  $i^{\text{th}}$  row holds per-asset prices in the  $i^{\text{th}}$  scenario. Thus an optimal allocation is modeled as

$$x^*(C) = \arg\max_{x \in \Delta_n} \text{OWA}_w(Cx). \quad (23)$$

This experiment integrates robust portfolio optimization (23) end-to-end with per-scenario price prediction  $\hat{C} = \mathcal{M}_\theta(z)$ .

**Settings.** Historical prices of  $n = 50$  assets are obtained from the Nasdaq online database [Nasdaq, 2022] years 2015-2019, and  $N = 5000$  baseline asset price samples  $c_i$  are generated by adding Gaussian random noise to randomly drawn price vectors. Price scenarios are simulated as a matrix of multiplicative factors uniformly drawn as  $\mathcal{U}(0.5, 1.5)^{m \times n}$ , whose rows are multiplied elementwise with  $c_i$  to obtain  $C_i \in \mathbb{R}^{m \times n}$ . While future asset prices can be predicted on the basis of various exogenous data including past prices or sentiment analysis, this experiment generates feature vectors  $z_i$  using a randomly generated nonlinear feature mapping as described in Appendix A. The experiment is replicated in three settings which assume  $m = 3, 5$ , and 7 scenarios.

The predictive model  $\mathcal{M}_\theta$  is a feedforward neural network. At test time,  $\mathcal{M}_\theta$  is evaluated over a test set for the distribution  $(z, C) \in \Omega$ , by passing its predictions to a projected subgradient solver of (23). Complete details in Appendix A.

**Results.** Figure 2 shows percent regret in the OWA objective attained on average over the test set (lower is better). The end-to-end trained unweighted sum baseline outperforms the two-stage, while notice how both *OWA-QP* and *OWA-Moreau* reach substantially higher decision quality. *OWA-QP* performs slightly better, but cannot scale past 5 scenarios, highlighting the importance of the proposed Moreau envelope smoothing technique (Section 4.2).

For comparison, grey bars represent the result due to a non-smoothed OWA LP (9) implemented with implicit differentiation in cvxpylayers [Agrawal et al., 2019a]. Its poor performance under the OWA subgradient training shows that the proposed approximations of Section 4 are indeed actually necessary for accurate training.

Runtimes of the smoothed models (11) and (13) are compared in Figure 5 (Appendix A.1). These results show that the Moreau envelope smoothing maintains low runtimes as  $m$  increases, while the QP approximation suffers past  $m = 5$  and causes memory overflow past  $m = 6$ .

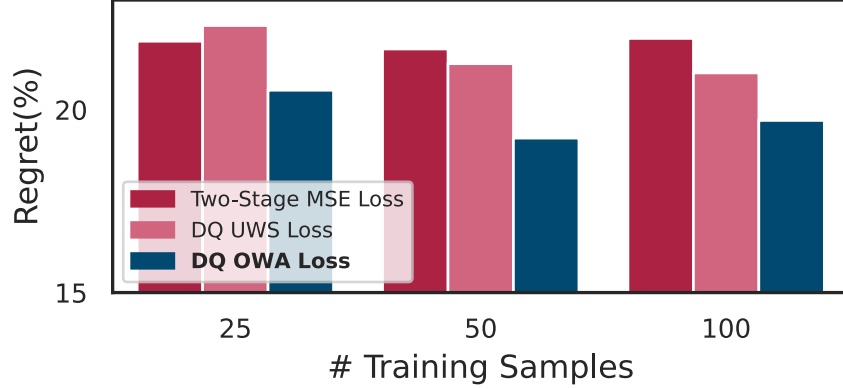


Figure 3: Multi-species OWA path length regret on WSP.

### 7.1.2 Surrogate Learning for OWA Optimization: Multi-species Warcraft Shortest Path

This experiment serves to illustrate how a surrogate model can assist in end-to-end training (3) when the full OWA problem (5) is too difficult to solve. Warcraft Shortest Path (WSP) is a popular dataset for benchmarking PtO methods [Pogančić et al., 2020, Berthet et al., 2020], in which observable features  $z$  are RGB images of  $12 \times 12$  tiled Warcraft maps. A character’s movement speed depends on the terrain type of each tile, and the goal is to predict the node-weighted shortest path from top-left to bottom-right where nodes are tiles and weights depend on movement speed.

This experiment is a multi-objective variation inspired by [Tang and Khalil, 2023], where multiple species have *distinct node weights* based on their movement speeds on each terrain type, and must traverse each map together by a *single path*. So that all species travel together as fast as possible, we aim to minimize their OWA-aggregated path lengths.

Noting that node weights can readily be converted to edge weights, the shortest path problem as a linear program reads

$$\mathbf{x}^*(\mathbf{c}) = \operatorname{argmax}_{\mathbf{A}\mathbf{x}=\mathbf{b}, 0 \leq \mathbf{x} \leq \mathbf{1}} -\mathbf{c}^T \mathbf{x} \quad (24)$$

where  $\mathbf{A}$  is a graph incidence matrix,  $\mathbf{b}$  indicates source and sink nodes, and  $\mathbf{c}$  holds the graph’s edge weights. A classic result states that due to total unimodularity in  $\mathbf{A}$ , solutions  $\mathbf{x}^*$  to (24) are guaranteed to take on integer values, so that  $\mathbf{x}^*(\mathbf{c}) \in \{0, 1\}^n$  form valid paths [Cormen et al., 2022].

Replacing the linear objective of (24) with an OWA aggregation over  $\mathbf{C}\mathbf{x}$  (where rows of  $\mathbf{C} \in \mathbb{R}^{m \times n}$  hold edge weights per species) breaks this property, so that additional integer constraints  $\mathbf{x} \in \{0, 1\}^n$  are required, leading to an intractable OWA integer program:

$$\mathbf{x}^*(\mathbf{C}) = \operatorname{argmax}_{\mathbf{A}\mathbf{x}=\mathbf{b}, \mathbf{x} \in \{0,1\}^n} \text{OWA}_w(-\mathbf{C}\mathbf{x}) \quad (25)$$

Rather than training a predictor of  $\hat{\mathbf{C}} \in \mathbb{R}^{m \times n}$  together with (25), we predict  $\hat{\mathbf{c}} \in \mathbb{R}^n$  with (24) as a differentiable LP surrogate model using [Pogančić et al., 2020], along with the OWA aggregated path length  $\text{OWA}_w(-\mathbf{C}\mathbf{x}^*(\hat{\mathbf{c}}))$  as a loss function. This ensures integrality of  $\mathbf{x}^*(\mathbf{c})$  while maintaining an efficient training procedure which requires only to solve (24) at each training iteration and at inference.

**Settings.** Three species’ node weights are derived by reassigning the movement speeds of each terrain type in the WSP dataset: Humans are fastest on land, Naga on water, and Dwarves on rock, to generate ground-truth  $\mathbf{C} \in \mathbb{R}^{m \times n}$ .  $\mathcal{M}_\theta$  is a ResNet18 CNN trained to map  $12 \times 12$  tiled Warcraft maps to node weights of a shortest path problem (24). Blackbox differentiation [Pogančić et al., 2020] is used to backpropagate its solution by Dijkstra’s method. See Appendix C for an example of the input feature data  $z$ . ResNet18 is strong enough to enable competitive performance by the two-stage given enough data. Following [Tang and Khalil, 2023], we focus on the limited-data regime, where the advantage of end-to-end learning is best revealed, using 25, 50 or 100 training samples and 1000 for testing.

**Results.** Figure 3 records percentage regret due to two-stage and unweighted sum baseline models, along with the proposed differentiable LP surrogate trained under OWA loss. Notice how, in each case, the OWA-trained model shows a clear advantage in minimization of OWA regret.

Table 3 (Appendix C) shows the per-species regret in terms of path length, and reveals that OWA training significantly improves the highest path-length among species, which is intuitive to provide fairness and the main contributor to the aggregated OWA value.



## 7.2 Nonparametric OWA with Blackbox Solver: Fair Learning to Rank

The final application setting studies the fair learning-to-rank problem, in which a prediction model ranks  $n$  web search results in order of relevance to a user query, while maintaining fairness of exposure across protected groups within the search results. The proposed model learns relevance scores  $c$  end-to-end with a fair ranking optimization module:

$$\Pi^*(c) = \operatorname{argmax}_{\Pi \in \mathcal{B}} (1 - \lambda) \cdot c^T \Pi b + \lambda \cdot \text{OWA}_w(\mathcal{E}_G(\Pi)), \quad (26)$$

wherein  $\mathcal{B}$  is the set of all bistochastic matrices,  $\Pi \in \mathbb{R}^{n \times n}$  represents a ranking policy whose  $(i, j)^{\text{th}}$  element is the probability item  $i$  takes position  $j$  in the ranking,  $c$  measures relevance of each item to a user query,  $b$  are position bias factors measuring the exposure of each ranking position, and  $c^T \Pi b$  is the expected Discounted Cumulative Gain, a common measure of user utility. This primary objective is combined with OWA aggregation of the exposure vector  $\mathcal{E}_G(\Pi)$ , whose elements  $\mathcal{E}_g(\Pi) = \mathbf{1}_g^T \Pi b$  measure the exposures attained by each of several protected groups  $g \in G$  where  $\mathbf{1}_g$  hold binary indicators of item inclusion in group  $g$ . The factor  $\lambda$  controls a tradeoff between user utility and group fairness of exposure.

Since  $b$  and  $\mathbf{1}_g$  in  $\mathcal{E}_G(\Pi)$  are known and not modeled parametrically, the problem (26) is an instance of (16) and its SPO+ subgradient can be modeled as per Section 5. Solutions  $\Pi^*(c)$  are obtained for any  $c$  by an adaptation of the Frank-Wolfe method with smoothing proposed in [Do and Usunier, 2022], as detailed in Appendix B.

**Settings.** A feedforward network  $\mathcal{M}_\theta$  is trained to predict for  $n$  items, given features  $z$ , their relevance scores  $c \in \mathbb{R}^n$ . The SPO+ training scheme of Section 5 is used to minimize regret in (26) due to error in  $\hat{c} = \mathcal{M}_\theta(z)$ . The Microsoft Learning to Rank (MSLR) dataset is used, where  $z$  are features of items to be ranked and  $c$  are their relevance scores. Protected item groups are assigned as evenly spaced quantiles of its Quality Score feature. Each method is evaluated on the basis of mean utility  $c^T \Pi b$  and fairness violation  $\frac{1}{|G|} \sum_{g \in G} |\frac{1}{n} \mathbf{1}^T \mathcal{E}_G(\Pi) - \mathcal{E}_g(\Pi)|$ , and their relative tradeoffs over the full range of its fairness parameter.

**Baseline Models.** The model proposed in this section is called Smart OWA Optimization for Fair Ranking (SOFaiR). Selected baseline methods from the fair learning to rank domain include FULTR [Singh and Joachims, 2019], DELTR [Zehlike and Castillo, 2020], and SPOFR [Kotary et al., 2022], further details are provided in Appendix B.

**Results.** Figure 4 (left) shows that by enforcing fairness via an embedded optimization, SoFaiR achieves order-of-magnitude lower fairness violations than FULTR or DELTR, which rely on loss function penalties to drive down violations. However, it is Pareto-dominated over a small regime by those methods. Its fairness-utility tradeoff is comparable to SPOFR, which also uses constrained optimization. Notably though, SoFaiR demonstrates order-of-magnitude runtime advantages over SPOFR in Appendix B.

Figure 4 (right) shows the analogous result over datasets with 3-7 protected groups. None of the baseline methods are equipped to handle multiple groups on this dataset, but SOFaiR accomodates more groups naturally by OWA optimization over their expected group exposures.

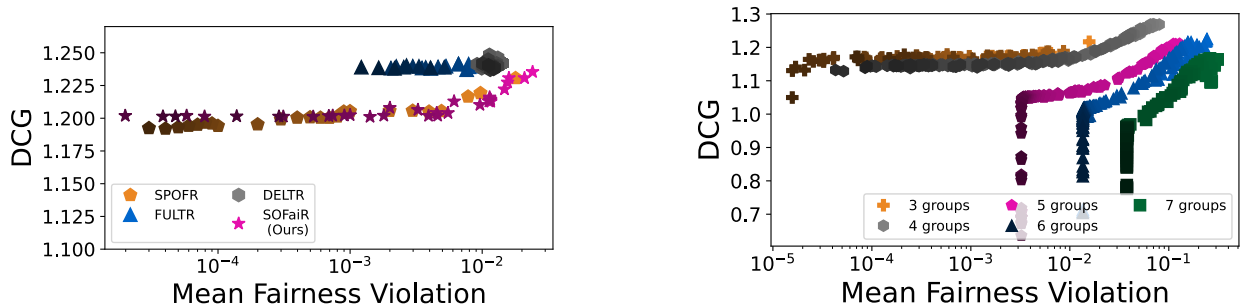


Figure 4: Fairness-utility tradeoffs on MSLR. Left: binary group case, right: multi-group case

## 8 Related Work

Modern approaches to the Predict-Then-Optimize setting, formalized in Section 2.2, typically maximize decision quality as a loss function, enabled by backpropagation through the mapping  $c \rightarrow x^*(c)$  defined by (1). When this mapping is

differentiable, backpropagation can be performed using differentiable optimization libraries [Amos and Kolter [2017], Agrawal et al. [2019a], [b], Kotary et al. [2023]].

However, many important classes of optimization are nondifferentiable, including linear and mixed-integer programs. Effective training techniques are typically based on forming continuous approximations of (1), whether by smoothing the objective function [Amos et al. [2019], Wilder et al. [2019a], Mandi and Guns [2020], introducing random noise [Berthet et al. [2020], Paulus et al. [2020], or estimation by finite differencing [Pogančić et al. [2019]]. This paper falls into that category, due to nondifferentiability of the OWA objective, requiring approximation of (1) by differentiable functions.

## 9 Conclusions

This work has presented a comprehensive methodology for incorporating Fair OWA optimization end-to-end with predictive modeling. Beginning with novel differentiable approximations to OWA programs, its proposed toolset also included special techniques to exploit problem specific structure arising in practical problems, such as nonparametric OWA objectives and totally unimodular constraints. These developments were used to demonstrate the potential of Fair OWA optimization in data-driven decision making, with results not previously possible on important problems, such as robust resource allocation and fair learning to rank.

We believe that this work could pave the way to the use of Fair OWA in learning pipelines to enable an array of important multi-optimization problems in many engineering domains.

## References

- Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019a.
- Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walaa M Moursi. Differentiating through a cone program. *arXiv preprint arXiv:1904.09043*, 2019b.
- Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, pages 136–145. JMLR. org, 2017.
- Brandon Amos, Vladlen Koltun, and J Zico Kolter. The limited multi-label projection layer. *arXiv preprint arXiv:1906.08707*, 2019.
- Amir Beck. *First-order methods in optimization*. SIAM, 2017.
- Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable perturbed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.
- Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR, 2020.
- Dany Cajas. Owa portfolio optimization: A disciplined convex programming framework. Available at SSRN 3988927, 2021.
- Y. Chen and A. Zhou. Multiobjective portfolio optimization via pareto front evolution. *Complex and Intelligent Systems*, 8:4301–4317, 2022. doi:[10.1007/s40747-022-00715-8](https://doi.org/10.1007/s40747-022-00715-8). URL <https://doi.org/10.1007/s40747-022-00715-8>
- Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.
- Virginie Do and Nicolas Usunier. Optimizing generalized gini indices for fairness in rankings. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 737–747, 2022.
- Virginie Do, Sam Corbett-Davies, Jamal Atif, and Nicolas Usunier. Two-sided fairness in rankings via lorenz dominance. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8596–8608. Curran Associates, Inc., 2021. URL [https://proceedings.neurips.cc/paper\\_files/paper/2021/file/48259990138bc03361556fb3f94c5d45-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2021/file/48259990138bc03361556fb3f94c5d45-Paper.pdf)
- Adam N Elmachtoub and Paul Grigas. Smart “predict, then optimize”. *Management Science*, 2021.
- Godfrey Harold Hardy, John Edensor Littlewood, and George Pólya. *Inequalities*. Cambridge university press, 1952.

- Dan A. Iancu and Nikolaos Trichakis. Fairness and efficiency in multiportfolio optimization. *Operations Research*, 62(6):1285–1301, 2014. doi:[10.1287/opre.2014.1310](https://doi.org/10.1287/opre.2014.1310). URL <https://doi.org/10.1287/opre.2014.1310>.
- James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Ziwei Zhu. End-to-end learning for fair ranking systems. In *Proceedings of the ACM Web Conference 2022*, pages 3520–3530, 2022.
- James Kotary, My H. Dinh, and Ferdinando Fioretto. Backpropagation of unrolled solvers with folded optimization. *arXiv preprint arXiv:2301.12047*, 2023.
- Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.
- Jayanta Mandi and Tias Guns. Interior point solving for lp-based prediction+optimisation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Jayanta Mandi, James Kotary, Senne Berden, Maxime Mulamba, Victor Bucarey, Tias Guns, and Ferdinando Fioretto. Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *arXiv preprint arXiv:2307.13565*, 2023.
- Harry M Markowitz. Foundations of portfolio theory. *The journal of finance*, 46(2):469–477, 1991.
- Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR, 2016.
- Nasdaq. Nasdaq end of day us stock prices. <https://data.nasdaq.com/databases/EOD/documentation>, 2022. Accessed: 2023-08-15.
- Włodzimierz Ogryczak and Tomasz Śliwiński. On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research*, 148(1):80–91, 2003.
- Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
- Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems*, 33:5691–5704, 2020.
- Laurent Perron. Operations research and constraint programming at google. In *Principles and Practice of Constraint Programming—CP 2011: 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings 17*, pages 2–2. Springer, 2011.
- Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2019.
- Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*, 2020.
- J Salas and V Yepes. Enhancing sustainability and resilience through multi-level infrastructure planning. *International Journal of Environmental Research and Public Health*, 17(3):962, 2020. doi:[10.3390/ijerph17030962](https://doi.org/10.3390/ijerph17030962).
- Ashudeep Singh and Thorsten Joachims. *Policy Learning for Fairness in Ranking*, pages 1–1. Curran Associates Inc., Red Hook, NY, USA, 2019.
- Bo Tang and Elias B. Khalil. Multi-task predict-then-optimize. In Meinolf Sellmann and Kevin Tierney, editors, *Learning and Intelligent Optimization*, pages 506–522, Cham, 2023. Springer International Publishing. ISBN 978-3-031-44505-7.
- Tom Terlouw, Tarek AlSkaif, Christian Bauer, and Wilfried van Sark. Multi-objective optimization of energy arbitrage in community energy storage systems using different battery technologies. *Applied Energy*, 239:356–372, 2019. ISSN 0306-2619. doi:<https://doi.org/10.1016/j.apenergy.2019.01.227>. URL <https://www.sciencedirect.com/science/article/pii/S0306261919302478>.
- Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 1658–1665, 2019a.
- Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI*, volume 33, pages 1658–1665, 2019b.
- Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. In Didier Dubois, Henri Prade, and Ronald R. Yager, editors, *Readings in Fuzzy Sets for Intelligent Systems*, pages 80–87. Morgan Kaufmann, 1993. ISBN 978-1-4832-1450-4. doi:<https://doi.org/10.1016/B978-1-4832-1450-4.50011-0>. URL <https://www.sciencedirect.com/science/article/pii/B9781483214504500110>.

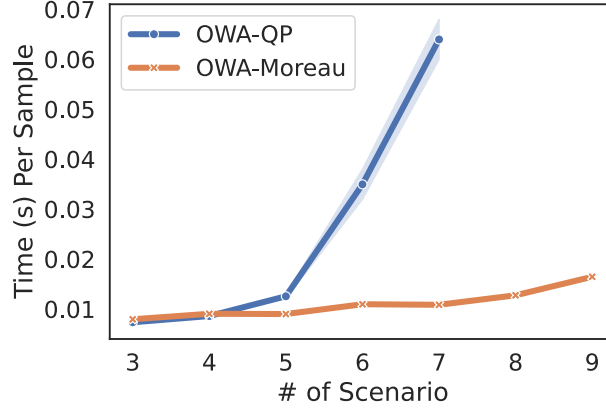


Figure 5: Average solving time of 2 smoothed OWA optimization models, on Robust Portfolio Optimization, over 1000 input samples. Missing datapoints past 7 scenarios are due to memory overflow as the QP model grows factorially.

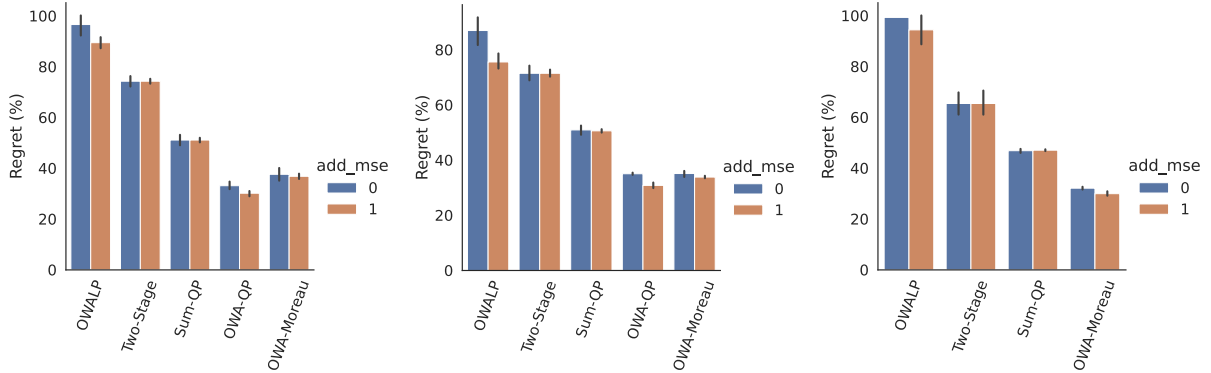


Figure 6: Effect of MSE Loss on differentiable optimization models. From left to right: 3, 5, 7 scenarios

Ronald R. Yager and J. Kacprzyk. *The Ordered Weighted Averaging Operators: Theory and Applications*. Springer Publishing Company, Incorporated, 2012. ISBN 1461378060.

Meike Zehlike and Carlos Castillo. Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of the web conference 2020*, pages 2849–2855, 2020.

## A Portfolio Optimization Experiment

### A.1 Efficiency of Differentiable OWA Solvers

Figure 5 depicts the running times of two differentiable optimization models applied to the Portfolio problem. It is evident that for the OWA-LP model, the running time scales factorially with the number of scenarios due to the number of constraints, while for the OWA-Moreau model, it scales linearly. It is noteworthy that the OWA-LP model cannot run with more than 7 scenarios due to memory constraints (requiring over 300GB+).

### A.2 Effect of adding MSE loss

Figure 6 illustrates the impact of combining the Mean Squared Error loss  $\mathcal{L}_{MSE}$  in a weighted combination with the decision quality loss  $\mathcal{L}_{DQ}$ . With the exception of OWA-LP, which exhibited instability, and Two-Stage, already trained with MSE Loss, the addition of MSE resulted in slight enhancements to the regret performance.

Table 1: Hyperparameters

Hyperparameter	Min	Max	Final Value					
			OWA-LP	Two-Stage	Sum-QP	OWA-QP	OWA-Moreau	Sur-QP
learning rate	$1e^{-3}$	$1e^{-1}$	$1e^{-2}$	$5e^{-3}$	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$
smoothing parameter $\epsilon$	0.1	1.0	N/A	N/A	1.0	1.0	N/A	1.0
smoothing parameter $\beta_0$	0.005	10.0	N/A	N/A	N/A	N/A	0.05	N/A
MSE loss weight $\lambda$	0.1	0.5	0.4	N/A	0.3	0.4	0.1	0.3

### A.3 Models and Hyperparameters

A neural network (NN) with three shared hidden layers following by one separated hidden layer for each species is trained using Adam Optimizer and with a batch size of 64. The size of each shared layer is halved, the output dimension of the separated layer equal to the number of assets. Hyperparameters were selected as the best-performing on average among those listed in Table 1. Results for each hyperparameter setting are averaged over five random seeds. In the OWA-Moreau model, the forward pass is executed using projected gradient descent for 300, 500, and 750 iterations, respectively, for scenarios with 3, 5, and 7 inputs. The update step size is set to  $\gamma = 0.02$ .

### A.4 Solution Methods

The OWA portfolio optimization problem (23) is solved at test time, for each compared method, by projected subgradient descent using OWA subgradients (7) and an efficient projection onto the unit simplex  $\Delta$  as in [Martins and Astudillo 2016]:

$$\mathbf{x}^{k+1} = \text{proj}_{\Delta} \left( \mathbf{x}^k - \alpha \frac{\partial}{\partial \mathbf{x}} \text{OWA}_w(C\mathbf{x}) \right) \quad (27)$$

For the Moreau-envelope smoothed OWA optimization (13) proposed for end-to-end training, the main difference is that its objective function is differentiable (with gradients (14)), which allows solution by a more efficient Frank-Wolfe method [Beck 2017], whose inner optimization over  $\Delta$  reduces to the simple argmax function which returns a binary vector with unit value in the highest vector position and 0 elsewhere, which can be computed in linear time:

$$\mathbf{x}^{k+1} = \frac{k}{k+2} \mathbf{x}^k + \frac{2}{k+2} \text{argmax} \left( \frac{\partial}{\partial \mathbf{x}} \text{OWA}_w(C\mathbf{x}^k) \right) \quad (28)$$

## B Fair Learning to Rank Experiment

### B.1 Fair Ranking Optimization by Frank-Wolfe with Smoothing

This section explains the adaptation of a Frank-Wolfe method with objective smoothing, due to [Do and Usunier 2022], to solve the fair ranking optimization mapping (26) proposed for end-to-end fair learning to rank in this paper.

Frank-Wolfe methods solve a convex constrained optimization problem  $\text{argmax}_{\mathbf{x} \in \mathbb{S}} f(\mathbf{x})$  by computing the iterations

$$\mathbf{x}^{(k+1)} = (1 - \alpha^{(k)}) \mathbf{x}^{(k)} + \alpha^{(k)} \text{argmax}_{\mathbf{y} \in \mathbb{S}} \langle \mathbf{y}, \nabla f(\mathbf{x}^{(k)}) \rangle. \quad (29)$$

Convergence to an optimal solution is guaranteed when  $f$  is *differentiable* and with  $\alpha^{(k)} = \frac{2}{k+2}$  [Beck, 2017]. However, the main obstruction to solving (26) by the method (29) is that  $f$  in our case includes a *non-differentiable* OWA function. A path forward is shown in [Lan, 2013], which shows convergence can be guaranteed by optimizing a smooth surrogate function  $f^{(k)}$  in place of the nondifferentiable  $f$  at each step of (29), in such a way that the  $f^{(k)}$  converge to the true  $f$  as  $k \rightarrow \infty$ .

It is proposed in [Do and Usunier, 2022] to solve a two-sided fair ranking optimization with OWA objective terms, by the method of [Lan, 2013], where  $f^{(k)}$  is chosen to be a Moreau envelope  $h^{\beta_k}$  of  $f$ , a  $\frac{1}{\beta_k}$ -smooth approximation of  $f$  defined as [Beck, 2017]:

$$h^{\beta}(\mathbf{x}) = \min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\beta} \|\mathbf{y} - \mathbf{x}\|^2. \quad (30)$$



**Algorithm 1:** Frank-Wolfe with Moreau Envelope Smoothing to solve (26)**Input:** predicted relevance scores  $\hat{\mathbf{y}} \in \mathbb{R}^n$ , group mask  $\mathbf{A}$ , max iteration  $T$ , smooth seq.  $(\beta_k)$ **Output:** ranking policy  $\Pi^{(T)} \in \mathbb{R}^{n \times n}$ 

- 1 Initialize  $\Pi^{(0)}$  as  $\mathbf{P} \in \mathcal{P}$  which sorts  $\hat{\mathbf{y}}$  in decreasing order;
- 2 **for**  $k = 1, \dots, T$  **do**
- 3    $\tilde{\boldsymbol{\mu}} \leftarrow \text{proj}_{\mathcal{C}(\tilde{\mathbf{w}})}(\mathcal{E}_G(\mathbf{A}\Pi)/\beta_k)$ ;
- 4    $\hat{\boldsymbol{\mu}} \leftarrow (1 - \lambda) \cdot \hat{\mathbf{y}}_q + \lambda \cdot (\mathbf{A}^T \tilde{\boldsymbol{\mu}})$ ;
- 5    $\hat{\sigma} \leftarrow \text{argsort}(-\hat{\boldsymbol{\mu}})$ ;
- 6   Let  $\mathbf{P}^{(k)} \in \mathcal{P}$  such that  $\mathbf{P}^{(k)}$  represents  $\hat{\sigma}$ ;
- 7    $\Pi^{(k)} \leftarrow \frac{k}{k+2} \Pi^{(k-1)} + \frac{2}{k+2} \mathbf{P}^{(k)}$ ;
- 8 **Return**  $\Pi^{(T)}$ ;

When  $f = \text{OWA}_{\mathbf{w}}$ , let its Moreau envelope be denoted  $\nabla \text{OWA}_{\mathbf{w}}^\beta$ ; it is shown in [Do and Usunier, 2022] that its gradient can be computed as a projection onto the permutahedron induced by modified OWA weights  $\tilde{\mathbf{w}} = -(w_m, \dots, w_1)$ . By definition, the permutahedron  $\mathcal{C}(\tilde{\mathbf{w}}) = \text{CONV}(\{\mathbf{w}_\sigma : \forall \sigma \in \mathcal{P}_m\})$  induced by a vector  $\tilde{\mathbf{w}}$  is the convex hull of all its permutations. In turn, it is shown in [Blondel et al., 2020] that the permutahedral projection  $\nabla \text{OWA}_{\mathbf{w}}^\beta(\mathbf{x}) = \text{proj}_{\mathcal{C}(\tilde{\mathbf{w}})}(\mathbf{x}/\beta)$  can be computed in  $m \log m$  time as the solution to an isotonic regression problem using the Pool Adjacent Violators algorithm. To find the overall gradient of  $\text{OWA}_{\mathbf{w}}^\beta$  with respect to optimization variables  $\Pi$ , a convenient form can be derived from the chain rule:

$$\nabla_{\Pi} \text{OWA}_{\mathbf{w}}^\beta(\mathcal{E}(\Pi)) = \boldsymbol{\mu} \mathbf{b}^T. \quad (31)$$

where  $\boldsymbol{\mu} = \text{proj}_{\mathcal{C}(\tilde{\mathbf{w}})}(\mathcal{E}(\Pi)/\beta)$  and  $\mathcal{E}(\Pi)$  is the vector of all item exposures [Do and Usunier, 2022]. For the case where group exposures  $\mathcal{E}_G(\Pi) \mathcal{E}_g(\Pi) = \mathbf{1}_g^T \Pi \mathbf{b}$  are aggregated by OWA,  $\mathcal{E}_G(\Pi) = \mathbf{A} \Pi \mathbf{b}$ , where  $\mathbf{A}$  is the matrix composed of stacking together all group indicator vectors  $\mathbf{1}_g \forall g \in G$ . Since  $\mathcal{E}(\Pi) = \Pi \mathbf{b}$ , this implies  $\mathcal{E}_G(\Pi) = \mathcal{E}(\mathbf{A} \Pi)$ , thus

$$\nabla_{\Pi} \text{OWA}_{\mathbf{w}}^\beta(\mathcal{E}_G(\Pi)) = (\mathbf{A}^T \tilde{\boldsymbol{\mu}}) \mathbf{b}^T. \quad (32)$$

by the chain rule, and where  $\tilde{\boldsymbol{\mu}} = \text{proj}_{\mathcal{C}(\tilde{\mathbf{w}})}(\mathcal{E}_G(\mathbf{A} \Pi)/\beta)$ . It remains now to compute the gradient of the user relevance term  $u(\Pi, \hat{\mathbf{y}}_q) = \hat{\mathbf{y}}_q^T \Pi \mathbf{b}$  in Problem (26). As a linear function of the matrix variable  $\Pi$ , its gradient is  $\nabla_{\Pi} u(\Pi, \hat{\mathbf{y}}_q) = \hat{\mathbf{y}}_q \mathbf{b}^T$ , which is evident by comparing to the equivalent vectorized form  $\hat{\mathbf{y}}_q^T \Pi \mathbf{b} = \overrightarrow{\hat{\mathbf{y}}_q}^T \cdot \overrightarrow{\Pi \mathbf{b}}$ . Combining this with (14), the total gradient of the objective function in (26) with smoothed OWA term is  $(1 - \lambda) \cdot \hat{\mathbf{y}}_q \mathbf{b}^T + \lambda \cdot (\mathbf{A}^T \tilde{\boldsymbol{\mu}}) \mathbf{b}^T$ , which is equal to  $((1 - \lambda) \cdot \hat{\mathbf{y}}_q + \lambda \cdot (\mathbf{A}^T \tilde{\boldsymbol{\mu}})) \mathbf{b}^T$ . Therefore the SOFaiR module's Frank-Wolfe linearized subproblem is

$$\underset{\Pi \in \mathcal{B}}{\text{argmax}} \langle \Pi, ((1 - \lambda) \cdot \hat{\mathbf{y}}_q + \lambda \cdot (\mathbf{A}^T \tilde{\boldsymbol{\mu}})) \mathbf{b}^T \rangle \quad (33)$$

To implement the Frank-Wolfe iteration (29), this linearized subproblem should have an efficient solution. To this end, the form of each gradient above as a cross-product of some vector with the position biases  $\mathbf{b}$  can be exploited. Note that as the expected DCG under relevance scores  $\mathbf{y}$ , the function  $\mathbf{y}^T \Pi \mathbf{b}$  is maximized by the permutation matrix  $\mathbf{P} \in \mathcal{P}_n$  which sorts the relevance scores  $\mathbf{y}$  decreasingly. But since  $\mathbf{y}^T \Pi \mathbf{b} = \overrightarrow{\mathbf{y}}^T \cdot \overrightarrow{\Pi \mathbf{b}}$ , we identify  $\mathbf{y}^T \Pi \mathbf{b}$  as the linear function of  $\overrightarrow{\Pi \mathbf{b}}$  with gradient  $\overrightarrow{\mathbf{y}} \mathbf{b}^T$ . Therefore problem (33) can be solved in  $\mathcal{O}(n \log n)$ , simply by finding  $\mathbf{P} \in \mathcal{P}_n$  as the argsort of the vector  $((1 - \lambda) \cdot \hat{\mathbf{y}}_q + \lambda \cdot (\mathbf{A}^T \tilde{\boldsymbol{\mu}}))$  in decreasing order. A more formal proof, cited in [Do et al., 2021], makes use of [Hardy et al., 1952].

## B.2 Running Time Analysis

Our analysis begins with a runtime comparison between SOFaiR and other LTR frameworks, to show how it overcomes inefficiency at training and inference time. Figure 7 shows the average training and inference time per query for each method, focusing on the binary group MSLR dataset across various list sizes. First notice the drastic runtime reduction of SOFaiR compared to SPOFR, both during training and inference. While SPOFR's training time exponentially increases with the ranking list size, SOFaiR's runtime increases only moderately, reaching over one order of magnitude speedup over SPOFR for large list sizes. Notably, the number of iterations of Algorithm 1 required for sufficient accuracy in training to compute SPO+ subgradients are found to be less than those required for solution of (26) at inference. Thus the reported results use 100 iterations in training and 500 at inference. Importantly, reported runtimes under-estimate the

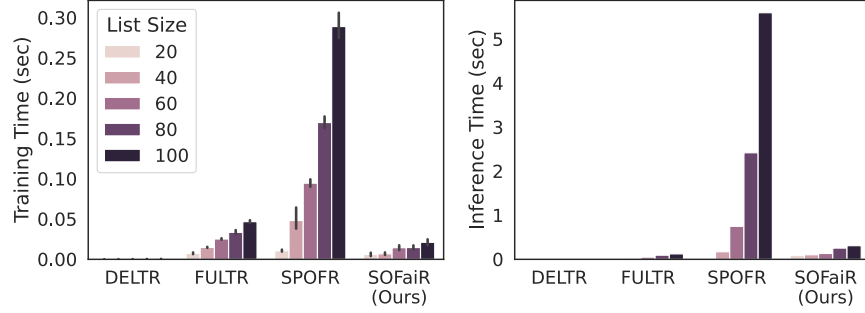


Figure 7: Running time benchmark on MSLR-Web10k dataset

Table 2: Hyperparameters

Hyperparameter	Min	Max	Final Value			
			SOFaiR	SPOFR	FULTR	DELTR
learning rate	$1e^{-5}$	$1e^{-1}$	$1e^{-1}$	$1e^{-1}$	$2.5e^{-4}$	$2.5e^{-4}$
violation penalty $\lambda$	$1e^{-5}$	400	*	N/A	*	*
allowed violation $\delta$	0	0.01	N/A	*	N/A	N/A
entropy regularization decay	0.1	0.5	N/A	N/A	0.3	N/A
batch size	64	512	256	256	256	256
smoothing parameter $\beta_0$	0.1	100	*	N/A	N/A	N/A
sample size	32	64	N/A	64	64	N/A

efficiency gained by SOFaiR, since its PyTorch [Paszke et al., 2017] implementation in Python is compared against the highly optimized code implementation of Google OR-Tools solver [Perron, 2011]. DELTR and FULTR, being penalty-based methods, demonstrate competitive runtime performance. However, this efficiency comes at the expense of their ability to ensure fairness in every generated policy.

### B.3 Models and Hyperparameters

**Models and hyperparameters.** A neural network (NN) with three hidden layers is trained using Adam Optimizer with a learning rate of 0.1 and a batch size of 256. The size of each layer is halved, and the output is a scalar item score. Results of each hyperparameter setting is are taken on average over five random seeds.

Fairness parameters, considered as hyperparameters, are treated differently. LTR systems aim to offer a trade-off between utility and group fairness, since the cost of increased fairness results in decreased utility. In DELTR, FULTR, and SOFaiR, this trade-off is indirectly controlled through the fairness weight, denoted as  $\lambda$  in (26). Larger values of  $\lambda$  indicate more preference towards fairness. In SPOFR, the allowed violation of group fairness is specified directly. Ranking utility and fairness violation are assessed using average DCG and fairness violation, respectively. The metrics are computed as averages over the entire test dataset.

Hyperparameters were selected as the best-performing on average among those listed in Table 2. Final hyperparameters for each model are as stated also in Table 2 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.

For OWA optimization layers,  $\mathbf{w}$  is set as  $w_j = \frac{n-1+j}{n}$ ,  $T = 100$  during training, and  $T = 500$  during testing.

### B.4 Additional Results

This section includes additional results for fair learning to rank on MSLR, in which list sizes to be ranked are increased to 100 items. This allows runtimes to be compared as a function of list size, which determines the size of the fair ranking optimization problem. It also reveals how penalty-based methods DELTR and FULTR suffer in their ability to satisfy fairness accurately.

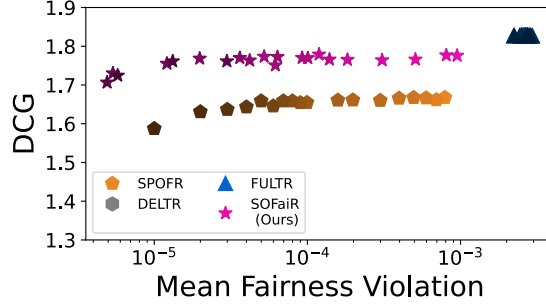


Figure 8: Fairness-utility tradeoffs on MSLR

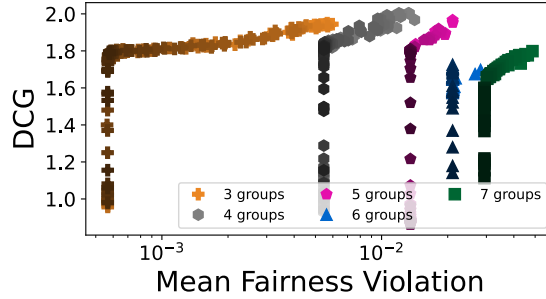


Figure 9: Fairness-utility tradeoffs on MSLR Multi-group

## C Multi-species Warcraft Shortest Path

Figure 10 showcases the Warcraft map featuring the shortest paths for three distinct species. The paths for Humans, Naga, and Dwarves are depicted in green, red, and blue, respectively. Humans excel on land, Naga traverse water most efficiently, while Dwarves navigate rocky terrain with the greatest speed.

Table 3 presents the regrets for each species across three models with different number of training data. It is notable that the model trained with OWA Loss significantly outperforms the two-stage model by more than 10% for the Human race. Conversely, the two-stage model exhibits slightly better performance for the Dwarf, albeit by a very small margin (<3%).



Figure 10: Shortest paths for each species in a Warcraft map

Model	Human			Naga			Dwarf		
	25	50	100	25	50	100	25	50	100
Two-Stage MSE Loss	44.4	44.6	46.3	<b>34.2</b>	<b>34.1</b>	<b>33.9</b>	44.1	<b>41.6</b>	<b>42.8</b>
End2End Sum Loss	51.5	49.0	47.9	35.2	33.6	34.6	43.8	43.8	43.4
<b>End2End OWA Loss</b>	<b>43.8</b>	<b>31.8</b>	<b>33.6</b>	<b>34.2</b>	37.6	34.8	<b>41.3</b>	43.1	43.1

Table 3: Regret (%) per species