Bridging Machine Learning and Optimization: Learning Fair and Scalable Problem Solving

by

My Dinh

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

University of Virginia

April 10, 2025

Advisors
Prof. Ferdinando Fioretto

Committee Members
Prof. Anil Vullikanti
Department of Computer Science,
University of Virginia

Prof. Henry Kautz
Department of Computer Science,
University of Virginia

Prof. Jundong Li
Departments of Electrical Engineering and Computer Science,
University of Virginia

Prof. Max Biggs
Darden School of Business,
University of Virginia

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Nando, for his years of investment, support, and collaboration. Beyond being a great mentor and advisor, he has fostered a nurturing lab environment that has greatly enriched my academic journey. I am deeply appreciative of the freedom he has given me to explore topics of personal interest while also encouraging me to take on more challenging and impactful projects. Over the past few years, I have learned an immense amount from him, both academically and personally, and I will truly miss our spontaneous lab discussions.

I would especially like to thank James for being an incredibly supportive and brilliant collaborator. His enthusiasm for optimization and our countless discussions on new ideas have been a great source of inspiration for my research. His high standards and sharp insights have consistently challenged me to think more critically and rigorously.

My appreciation also extends to all the past and present members of the RAISE lab—Cuong, Vincenzo, Saswat, Jacob, Michael, Jinhao, St. John, and Rocket—who have been fantastic colleagues and sources of both inspiration and laughter. Though our academic paths have diverged, each of you has contributed to making my PhD journey more enriching and enjoyable.

My sincere appreciation goes to my collaborators outside of UVA, whose valuable discussions and contributions have shaped this thesis: Mostafa Mohammadian, Kyri Baker, Lauryn P. Gouldin, and William

Yeoh. I am also incredibly thankful to my thesis committee—Anil Vullikanti, Henry Kautz, Maxwell Biggs, and Jundong Li—for their guidance and support throughout this process.

I am also grateful to my Vietnamese friends—Lan Anh, Kevin, Duong, Hien, Minh, and Huyen—for making my first two years at Syracuse feel like home. Their companionship filled those years with warmth, joy, countless hotpot gatherings, and unforgettable sleepless Vietnamese poker nights.

I extend my heartfelt gratitude to my colleagues at Trusting Social in Vietnam. They were my first mentors and collaborators, helping me lay the foundation for my journey in deep learning and machine learning. Their warm-hearted support made the beginning of this adventure incredibly exciting. Without them, I would not have taken the first steps on this path.

Looking back, I am deeply grateful for my years in graduate school and the many lessons—both technical and personal—that I have learned along the way. Every challenge, discussion, and breakthrough has been part of a journey that I have truly enjoyed and will always cherish.

Finally, and most importantly, I dedicate my deepest gratitude to my family, especially my mother, for her unconditional love, support, and sacrifices. Her unwavering belief in me has been my greatest source of inspiration and motivation. Without her, none of this would have been possible.

Abstract

This thesis explores the integration of Machine Learning (ML) and Optimization through two frameworks: Predict-Then-Optimize (PtO) and Learning to Optimize (LtO), with a focus on enhancing fairness, scalability, and efficiency in complex decision-making systems. The PtO framework integrates ML and optimization by incorporating the optimization layer directly into the ML training process. Our contributions in this area focus on multi-objective optimization applications, including learning-to-rank, court scheduling, and portfolio management. By integrating optimization layers into ML models, we address challenges such as fairness and risk management, leading to more robust and effective decision-making processes. The LtO framework utilizes ML models, particularly neural networks, to accelerate the solution of constrained optimization problems. We demonstrate its application in power systems, specifically for the AC Optimal Power Flow problem, where our approach significantly reduces computational costs while maintaining high accuracy and low constraint violation. Additionally, we investigate a hybrid of PtO and LtO by learning surrogate models for intractable Mixed-Integer Programming (MIP) problems in an end-to-end framework, illustrated through a court scheduling application. These studies highlight the potential of combining both domains to enhance real-world problem-solving capabilities.

Contents

List of Publications **xii**

1 Introduction **1**
1.1 Problem Settings 3
1.1.1 Predict-then-Optimize 3
1.1.2 Learning to Optimize 4
1.2 Motivation and Research Questions 5

2 Background **9**
2.1 Constrained Optimization 10
2.2 Deep Learning 12
2.3 Predict-then-Optimize 14
2.4 Learning-to-Optimize 17

3 End-to-End Learning for Fair Multiobjective Optimization Under Uncertainty **21**
3.1 End-to-End Learning for Fair Multiobjective Optimization Under Uncertainty 22
3.2 Preliminaries 23
3.2.1 Fair OWA and its Optimization 23
3.2.2 Predict-Then-Optimize Learning 24
3.3 End-to-End Learning with Fair OWA Optimization 25
3.4 Differentiable Approximations of OWA Optimization . . . 26

3.4.1 OWA LP with Quadratic Smoothing 26
3.4.2 Moreau Envelope Smoothing 29
3.5 Experiments 30
3.5.1 Robust Markowitz Portfolio Problem 31
3.5.2 Moreau Envelope as a Loss Function 35
3.6 Conclusions 37

4 Integrating Machine Learning and Constrained Optimization: Fairness-Aware Learning-to-Rank 39
4.1 Learning Fair Ranking Policies via Integration with Constrained Optimization 40
4.2 Preliminaries 43
4.2.1 Problem Setting and Goals 43
4.2.2 Fairness of Exposure 45
4.3 Limitations of Fair LTR Methods 47
4.4 Smart OWA Optimization for Fair Learning to Rank (SOFaiR) 49
4.4.1 Ordered Weighted Averaging Operator 50
4.4.2 End-to-End Learning in SOFaiR 51
4.5 Forward Pass Optimization 53
4.6 Backpropagation 56
4.7 Experiments 59
4.7.1 Running Time Analysis 60
4.7.2 Fairness and Utility Tradeoffs Analysis 61
4.7.3 Multi-Group Fairness Analysis 63
4.8 Conclusions 64

5 Learning to Optimize with Application in AC-OPF Problem 65
5.1 Deep Learning and Optimal Power Flow Problem 66
5.2 Related Work 68
5.3 Preliminaries 69
5.4 OPF Learning Goals 71
5.5 Deep Learning Proxies for AC-OPF: Roadmap 72
5.6 Generator's Characteristics. 73

5.7 Network Characteristics 76
5.8 Constraints 81
5.9 A Novel RNN-based Learning Framework 85
5.10 Conclusions 89

6 End-to-End Optimization and Learning of Fair Court Schedules 91
6.1 Optimization and Learning of Fair Court Schedules 92
6.2 Related Work 96
6.3 Motivations and Problem Setting 97
6.4 Preliminaries: Fair OWA Aggregation 101
6.5 Fair Optimization of Court Schedules 102
6.5.1 Group Fairness 103
6.5.2 Complexity of the Optimization Models 103
6.6 Optimization and Learning for Fair Court Schedules . . . 104
6.6.1 End-to-End Trainable Scheduling Model 105
6.6.2 Differentiable Matching Layer 106
6.6.3 OWA as a Loss Function 107
6.7 Experimental Settings 108
6.7.1 Data Generation Process 109
6.7.2 Model Settings and Evaluation Metrics 110
6.7.3 Baseline Models 111
6.8 Results 111
6.8.1 OWA Utility Regret 111
6.8.2 Normalized mean pairwise distances 113
6.8.3 Running Time 114
6.9 Conclusions 116

7 Future Directions: Diffusion for Learning-to-Optimize Constrained Optimization 117
7.1 Background and Related Work 118
7.1.1 Diffusion Models 118
7.1.2 Diffusion Models for Continuous Optimization . . . 119
7.1.3 Diffusion Models for Discrete Optimization 120

7.2 Preliminaries: Diffusion Model on Learning to Solve Simple CO Problems 120
7.2.1 Quadratic Programming 120
7.2.2 Maximal Independent Set 122
7.3 Proposed Solutions: Neural Optimization via Energy-Based Diffusion Models 125
7.3.1 Optimization as Energy-Based Inference 125
7.3.2 Continuous Domains: Sliced Score Matching and Langevin Sampling 125
7.3.3 Discrete Domains: Score Entropy Diffusion 126
7.4 Conclusion 127

8 Conclusion **129**

Appendices **147**

Appendix for Chapter 6 **149**
.1 Causal Graph Conditional Probability Tables 150

List of Publications

Part of the work reported in this thesis was published in the following publications:

- My H. Dinh, James Kotary, Ferdinando Fioretto. **Learning fair ranking policies via differentiable optimization of ordered weighted averages**. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2024
- My H. Dinh, James Kotary, Ferdinando Fioretto. **Differentiable approximations of fair OWA optimization**. In *ICML 2024 Workshop on Differentiable Almost Everything*, 2024
- My H. Dinh, James Kotary, Ferdinando Fioretto. **End-to-End Learning for Fair Multiobjective Optimization Under Uncertainty**. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI).*, 2024
- My H. Dinh, James Kotary, Lauryn P. Gouldin, William Yeoh, and Ferdinando Fioretto. **End-to-End Optimization and Learning of Fair Court Schedules**. CoRR, 2024
- My H. Dinh, Ferdinando Fioretto, Mostafa Mohammadian, Kyri Baker. **An Analysis of the Reliability of AC Optimal Power Flow Deep Learning Proxies**. In *IEEE PES Innovative Smart Grid Technologies*, 2023

Other publications which are not part of the thesis:

- James Kotary, My H. Dinh, Ferdinando Fioretto. **Backpropagation of Unrolled Solvers with Folded Optimization**. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023
- My H. Dinh, Ferdinando Fioretto. **Context-Aware Differential Privacy for Language Modeling**, CoRR abss/2301.12288, 2023
- Cuong Tran, My H. Dinh, Kyle Beiter, Ferdinando Fioretto. **Learning Solutions for Intertemporal Power Systems Optimization with Recurrent Neural Networks**. In *17th International Conference on Probabilistic Methods Applied to Power Systems*, 2022
- Cuong Tran, My H. Dinh, Kyle Beiter, Ferdinando Fioretto. **A fairness analysis on private aggregation of teacher ensembles**. In *AAAI Workshop on Privacy Preserving Artificial Intelligence (PPAI)*, 2022
- Cuong Tran, My H. Dinh, and Ferdinando Fioretto. **Differentially private deep learning under the fairness lens**. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021

CHAPTER 1

Introduction

"Innovation is taking two things that already exist and putting them together in a new way."

Tom Freston

Machine Learning (ML) and Optimization have traditionally developed as separate fields, each addressing distinct challenges with specialized methods. However, as real-world decision-making grows increasingly complex and large-scale, their integration presents new opportunities to leverage their strengths. Optimization is crucial for tasks like scheduling, logistics, and resource allocation but solving them can be computationally expensive, especially for large or complex problems. ML, on the other hand, excels at learning patterns, handling vast data, and adapting to new scenarios but often lacks explicit constraint enforcement, leading to suboptimal decisions. By combining these fields, we can create powerful and practical toolkits addressing challenges that neither field can solve alone.

The integration between these two fields can be largely divided into two main frameworks: Predict-Then-Optimize (PtO) and Learning-to-Optimize (LtO). The PtO framework embeds optimization directly into ML training to align predictions with decision-making objectives. Traditionally, ML predictions serve as inputs to optimization models, but when these stages are disconnected, prediction errors can propagate, leading to suboptimal decisions. For example, in court scheduling, an ML model predicts defendants' appearance probabilities across time slots, which an optimization module then uses to generate schedules that maximize attendance while ensuring fair treatment to the individual's satisfaction. However, if ML predictions are inaccurate or misaligned with the optimization goal, the final scheduling outcomes may suffer. PtO addresses this by incorporating optimization layers into ML training, allowing them to learn how their predictions influence final decisions. Although solving optimization problems within ML training can sometimes be computationally challenging, PtO has shown promise in multi-objective optimization tasks such as fair ranking, equitable resource allocation, and risk-aware portfolio management.

The LtO framework, on the other hand, involves using ML models, particularly neural networks, to assist in solving constrained optimization problems. Many real-world optimization tasks, such as nonlinear or mixed-integer programming, become computationally expensive as they scale, making traditional solvers inefficient, especially for real-time ap-

plications. A prime example is the electrical power grid, which powers modern life but suffers from inefficiencies. In the U.S, it supports $400 billion worth of electricity annually over 600,000 miles of power lines, yet it accounts for about 25% of global carbon emissions. For optimization problems like optimal power flow (OPF) in energy networks, ML can help reduce computation time by learning from past problem instances. This speed is crucial for real-time decision-making tasks such as unit commitment and power dispatch, where fast and accurate solutions are necessary. Additionally, hybrid approaches combine ML with traditional solvers, where an ML model provides a fast, approximate solution that is then corrected using conventional optimization methods.

This thesis aims to advance the integration of ML and Optimization, focusing on fairness, scalability, and efficiency. By bridging the gap between these two fields, we aim to develop more robust, practical, and socially responsible decision-making systems for complex real-world challenges.

1.1. Problem Settings

1.1.1 Predict-then-Optimize

The *Predict-Then-Optimize* (PtO) framework models decision-making processes as optimization problems with unspecified parameters $\boldsymbol{c}$, which must be estimated by a machine learning (ML) model, given correlated features $\boldsymbol{z}$. An estimation of $\boldsymbol{c}$ completes the specification of an optimization problem which is then solved to produce a final decision. The problem is posed as estimating the solution $\boldsymbol{x}^\star(\boldsymbol{c}) \in \mathcal{S} \subset \mathbb{R}^n$ of a parametric optimization problem:

$$\boldsymbol{x}^\star(\boldsymbol{c}) = \underset{\boldsymbol{x}\in\mathcal{S}}{\operatorname{argmax}} \quad f(\boldsymbol{x}, \boldsymbol{c}) \tag{1.1}$$

Although the true value of $\boldsymbol{c}$ is unknown, correlated *feature* values $\boldsymbol{z} \sim \mathcal{Z}$ can be observed. The goal is to learn a predictive model $\mathcal{M}_\theta : \mathcal{Z} \to \mathcal{C}$ from features $\boldsymbol{z}$ to estimate problem parameters $\hat{\boldsymbol{c}} = \mathcal{M}_\theta(\boldsymbol{z})$, such that the resulting solution's empirical objective value under ground-truth

parameters, is maximized. That is,

$$\underset{\theta}{\text{argmax}} \quad \mathbb{E}_{(\boldsymbol{z},\boldsymbol{c})\sim\Omega}\, f\left(\boldsymbol{x}^\star(\mathcal{M}_\theta(\boldsymbol{z})), \boldsymbol{c}\right), \tag{1.2}$$

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

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.

The above training goal is often best realized by maximizing empirical *Decision Quality* as a loss function Mandi et al. (2024a), defined

$$\mathcal{L}_{DQ}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = f\left(\boldsymbol{x}^\star(\hat{\boldsymbol{c}}), \boldsymbol{c}\right). \tag{1.3}$$

Gradient descent training of (1.2) with $\mathcal{L}_{DQ}$ requires a model of gradient $\frac{\partial \mathcal{L}_{DQ}}{\partial \hat{\boldsymbol{c}}}$, either directly or through chain-rule composition $\frac{\partial \mathcal{L}_{DQ}}{\partial \hat{\boldsymbol{c}}} = \frac{\partial \boldsymbol{x}^\star(\hat{\boldsymbol{c}})}{\partial \hat{\boldsymbol{c}}} \cdot \frac{\partial \mathcal{L}_{DQ}}{\partial \boldsymbol{x}^\star}$. The primary strategy for modeling this overall gradient involves initially determining the decision quality loss function's gradient $\frac{\partial \mathcal{L}_{DQ}}{\partial \boldsymbol{x}^\star}$, followed by its backpropagation through $\boldsymbol{x}^\star$. Here, left-multiplication by the Jacobian is equivalent to backpropagation through the optimization mapping $\boldsymbol{x}^\star$. When $\boldsymbol{x}^\star$ is not differentiable, optimizations, and smooth approximations are required.

1.1.2 Learning to Optimize

Consider a generic constraints optimization problem. All are parameterized by a vector of coefficients $\boldsymbol{c} \in \mathbb{R}^m$. This setup defines a mapping from any given coefficient vector $\boldsymbol{c}$ to the corresponding optimal solution $\boldsymbol{x}^\star(\boldsymbol{c}) \in \mathbb{R}^n$ as follows:

$$\boldsymbol{x}^\star(\boldsymbol{c}) \in \underset{\boldsymbol{x}}{\text{argmax}} \quad f_{\boldsymbol{c}}(\boldsymbol{x}) \tag{1.4a}$$

$$\text{s.t} \quad \boldsymbol{h}_{\boldsymbol{c}}(\boldsymbol{x}) = 0 \tag{1.4b}$$

$$\boldsymbol{g}_{\boldsymbol{c}}(\boldsymbol{x}) \leq 0 \tag{1.4c}$$

For any choice of $\boldsymbol{c}$ this formulation specifies an optimization problem by defining the objective function $f_{\boldsymbol{c}} : \mathbb{R}^n \to \mathbb{R}$, inequality constraints $g_{\boldsymbol{c}} : \mathbb{R}^n \to \mathbb{R}^m$, and quality constraints $h_{\boldsymbol{c}} : \mathbb{R}^n \to \mathbb{R}^p$ which together determine the optimal solution $\boldsymbol{x}^\star(\boldsymbol{c})$.

The goal is to train an optimization proxy solver $\mathcal{X}_\theta : \mathbb{R}^m \to \mathbb{R}^n$ over a distribution of problem instances $\boldsymbol{c} \sim \mathcal{C}$ hich approximates the mapping $\boldsymbol{x}^\star(\boldsymbol{c})$ as defined by (1.4a). The proxy model $\mathcal{X}_\theta$ may consist of a deep neural network $\mathcal{N}_\theta$ with trainable weights θ, potentially combined with a non-trainable correction mechanism $\mathcal{K}$ to refine the quality of the solution, so that $\mathcal{X}_\theta = \mathcal{K} \circ \mathcal{N}_\theta$. The training objective for this proxy model $\mathcal{X}_\theta$ can be defined as:

$$\max_{\theta} \ \mathbb{E}_{\boldsymbol{c}\sim\mathcal{C}} \ [\ f_{\boldsymbol{c}}(\mathcal{X}_\theta(\boldsymbol{c}))\ ] \tag{1.5a}$$

$$\text{s.t} \quad \boldsymbol{h}_{\boldsymbol{c}}(\mathcal{X}_\theta(\boldsymbol{c})) = 0 \quad \forall \boldsymbol{c} \sim \mathcal{C} \tag{1.5b}$$

$$\boldsymbol{g}_{\boldsymbol{c}}(\mathcal{X}_\theta(\boldsymbol{c})) \leq 0 \quad \forall \boldsymbol{c} \sim \mathcal{C} \tag{1.5c}$$

The training goal (1.5) highlights that each solution $\mathcal{X}_\theta(\boldsymbol{c})$ generated by the proxy solver must satisfy the original problem's constraints. Subject to these constraints, the goal is to maximize the expected objective value across the distribution of problem instances.

1.2. Motivation and Research Questions

In both frameworks, this thesis explores how integrating ML and Optimization can lead to fair, scalable, and computationally efficient solutions. Within each of its main topics of interest, this thesis begins by outlining the motivations behind its research contributions. The following research questions establish the perspective of this work, defining its aims and goals relative to the current state of the literature.

Research Questions in Predict-then-Optimize Traditional ML pipelines separate prediction from decision-making, often leading to misalignment between predictive accuracy and downstream decision quality. This disconnect is particularly problematic in fairness-sensitive and

risk-averse settings, such as learning-to-rank applications, where systems must provide highly relevant recommendations while preventing winner-take-all dynamics. Conventional methods either enforce fairness constraints in post-processing—compromising accuracy—or rely on rigid optimization formulations that struggle with scalability and feasibility.

Designing a PtO-based system involves two key challenges: formulating an optimization problem that is both fair and computationally efficient and differentiating through the optimization mapping to enable gradient-based learning. Instead of treating fairness as constraints of the problem, we adopt a multi-objective optimization approach, balancing competing priorities such as individual utility and overall system performance. To guide our contributions in Chapters 4 and 3, we investigate:

1. What does it mean to optimize independent objectives fairly?
2. How can the PtO framework be extended to multi-objective setting?
3. How can optimization-aware training remain computationally efficient at scale?

To answer these questions, we leverage Ordered Weighted Averages (OWA) optimization, a widely used approach for fair multi-objective optimization. In Chapter 4.4.1, we discuss OWA's properties and its connection to fairness, followed by an intuitive explanation of its application to court scheduling in Chapter 6.3.

Employing optimization of an OWA objective in PtO is challenging due to its nondifferentiability, preventing backpropagation of its constrained optimization mapping $\mathbf{x}^\star(\mathbf{c})$ within machine learning models trained by gradient descent. To address this, we developed novel approximation methods for their incorporation in end-to-end learning, detailed in chapter 4.5, 4.6, 3.4 Our proposed framework exploits problem structure, enabling fast and scalable solutions, with empirical validation presented in Chapters 4.7, 3.5.

Research Questions in Learning to Optimize Optimization is crucial for decision-making in complex systems, yet increasing problem com-

plexity—due to factors such as renewable energy integration and fairness constraints—creates significant computational challenges. This thesis investigates how ML can enhance optimization by leveraging problem structures to develop scalable, data-driven solutions that improve both efficiency and solution quality. We focus on two applications where traditional optimization struggles:

- Power grid systems that require rapid, near-optimal solutions under physical and engineering constraints for real-time decision-making.
- Court scheduling, where optimization can reduce nonappearance rates by accounting for uncertainty in defendants' availability

Across these domains, we explore:

1. How can constraints and problem-specific structure be effectively embedded into the learning pipeline to improve solution quality?
2. How can ML enhance robustness under uncertainty while maintaining computational efficiency?

These questions motivate the research contributions of this thesis. For power grid systems, Chapter 5 presents both theoretical and empirical analysis of AC-OPF problems, revealing how problem complexity impacts ML model performance. Our analysis identifies latent factors that determine prediction accuracy to enhance the interpretability and reliability of ML-driven solutions. Building on these insights, we develop a novel autoregressive RNN architecture that learns the solution trajectories of iterative nonlinear solvers. To improve feasibility, we augment the training loss with a Lagrangian-based loss function (Chapter 5.9).

For the court scheduling problem, we introduce a proxy model with an OWA decision-quality loss function, approximating an intractable OWA Mixed-Integer Program (MIP). This hybrid PtO and LtO approach integrates ML with efficient scheduling algorithms to enhance fairness while maintaining computational efficiency (Chapter 6.6).

Through comprehensive empirical evaluation against state-of-the-art benchmarks, we demonstrate that our framework achieves superior runtime efficiency, particularly for real-time applications like power grid dispatch, enhanced solution stability across varying problem instances, and improved fairness satisfaction in resource allocation tasks.

Before detailing the original contributions of this thesis work, the next chapter gives a broad overview of preliminary concepts and related research work in the existing literature.

CHAPTER 2

Background

"To boldly go where no one has gone before."

Captain Kirk

This chapter provides the necessary background and related work for this thesis. It first introduces key concepts in constrained optimization and deep learning. Then, it reviews modern research at the intersection of machine learning and optimization, focusing on end-to-end trainable models, particularly in the predict-then-optimize and learning-to-optimize settings.

2.1. Constrained Optimization

Constrained optimization (CO) is a fundamental problem in mathematical optimization, where the goal is to minimize an objective function $f : \mathbb{R}^n \to \mathbb{R}_+$ over a set of decision variables $\boldsymbol{x} \in \mathbb{R}^n$, subject to constraints that define a feasible region. Mathematically, this is expressed as:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m, \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p. \end{aligned} \tag{2.1}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$ represents the inequality constraints, $h_j : \mathbb{R}^n \to \mathbb{R}$ represents the equality constraints. Together, these constraints define the feasible set defines the feasible set $\mathcal{C}$ in which a valid solution must lie. A solution $\boldsymbol{x}$ within $\mathcal{C}$ is a feasible solution. Among feasible solutions, one that achieves the lowest objective function value, i.e., $f(\boldsymbol{x}) \leq f(\boldsymbol{w})$ for all feasible $\boldsymbol{w}$, is an *optimal solution*.

Convex and Nonconvex optimization A well-studied subclass of constrained optimization problems is convex optimization, where the feasible set $\mathcal{C}$ is convex, and the objective function f is convex. Convex optimization problems offer several advantages: they can be solved efficiently using polynomial-time algorithms and provide strong theoretical guarantees on the existence, uniqueness, and stability of solutions (Boyd et al., 2004).

A problem is nonconvex if either the feasible region or the objective function is nonconvex. Unlike convex problems, nonconvex problems

may have multiple local minima, making it difficult to find the global optimum. Such problems can have multiple local minima. Many non-convex problems are NP-hard, making them computationally intractable for large instances.

Common Class of Constrained Optimization Problems

Another important class of constrained optimization problems involves linear constraints:

$$\mathcal{C} = \{\mathbf{x} \,:\, \boldsymbol{A}\mathbf{x} \leq \boldsymbol{b}\}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. In this case, the feasible set $\mathcal{C}$ is convex. Depending on the nature of the objective function f, different types of problems emerge:

1. Linear Programming (LP): The objective function is affine, i.e., $f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}$ for some $\boldsymbol{c} \in \mathbb{R}^n$. LPs are widely used in various fields and can be solved efficiently using the simplex method (Dantzig, 1951), interior-point methods (Boyd et al., 2004), and Augmented Lagrangian methods (Hestenes, 1969; Powell, 1969).

2. Quadratic Programming (QP): The objective function is quadratic, i.e., $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T Q\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$ where Q is symmetric. If Q is positive semidefinite, the problem remains convex and can be efficiently solved.

3. Mixed-Integer Programming (MIP): Some decision variables must take integer value ($\boldsymbol{x} \in \mathbb{N}^n$), introducing combinatorial complexity. MIPs are NP-hard, as their feasible set is discrete and non-convex. Common solution methods include branch-and-bound, branch-and-cut, and cutting-plane techniques (**?**).

4. Nonlinear Programming (NLP): The objective function or constraints are nonlinear. Many NLPs are nonconvex and lack efficient, general-purpose solvers. Classical methods include Lagrangian relaxation, sequential quadratic programming (SQP), and the interior-point method (Nocedal & Wright, 2006).

Lagrangian Formulation

To solve the general constrained optimization problem in (2.1), the Lagrangian function is introduced:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \mu_j h_j(\mathbf{x}) \tag{2.2}$$

where $\lambda_i \geq 0$ are the Lagrange multipliers associated with the inequality constraints and μ_j are the multipliers for the equality constraints.

Karush-Kuhn-Tucker (KKT) Conditions

The KKT conditions provide necessary (and, under some conditions, sufficient) conditions for a solution $\mathbf{x}^*$ to be optimal. These conditions include:

1. Primal feasibility: $g_i(\mathbf{x}) \leq 0, \quad h_j(\mathbf{x}) = 0$
2. Dual feasibility: $\lambda_i^* \geq 0$
3. Stationary: $\nabla f(\mathbf{x}^\star) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}^\star) + \sum_{j=1}^{p} \mu_j \nabla h_j(\mathbf{x}^\star) = 0$
4. Complementary slackness: $\lambda_i^* g_i(\mathbf{x}^*) = 0, \quad \forall i$

If these conditions hold, then $\mathbf{x}^\star$ is a candidate for an optimal solution.

This section primarily focuses on constrained optimization problems where the constraints are linear, and the objective function is either linear or quadratic. Additionally, we consider problems that involve continuous, integer, or mixed-integer decision variables. The Lagrangian formulation and KKT conditions provide background for the solution methods discussed in subsequent sections.

2.2. Deep Learning

Deep learning has emerged as a powerful approach for modeling complex patterns in high-dimensional data. Supervised Deep Learning can be

viewed as the task of approximating a complex non-linear mapping from labeled data. Deep Neural Networks (DNNs) are composed of a sequence of layers, where each layer takes inputs as the results of the previous layer LeCun et al. (2015). Formally, a DNN defines a function $f_\theta : \mathbb{R}^m \to \mathbb{R}^n$ parameterized by θ , which maps an input $\boldsymbol{x} \in \mathbb{R}^m$ to an output $f_\theta(\boldsymbol{x})$.

A standard feedforward neural network with L layers is given by:

$$\boldsymbol{o}^l = \sigma(\boldsymbol{W}^l \boldsymbol{x}^{l-1} + \boldsymbol{b}^l), \quad l = 1, \ldots, L$$

where $\boldsymbol{o}^{(0)} = \boldsymbol{x}$ is the input, $\boldsymbol{o}^l \in \mathbb{R}^{m_l}$ the output vector at layer l, $\boldsymbol{W} \in \mathbb{R}^{m_l \times m_{l-1}}$ and $\boldsymbol{b} \in \mathbb{R}^{m_l}$ are the weight matrix and bias for layer l, The function $\sigma(\cdot)$ is an activation function (e.g., ReLU or sigmoid). The final layer produces an output $\boldsymbol{o}^{(L)}$, which is used for classification or regression tasks.

Training deep networks involves optimizing a loss function $\mathcal{L}$, typically through stochastic gradient descent (SGD) or its variants:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(\boldsymbol{x}_i), y_i)$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ is the training dataset. Modern optimizers such as Adam (Kingma & Ba, 2015) and adaptive learning rates methods have improved convergences and stability. Regularization methods, including dropout (Srivastava et al., 2014) and batch normalization (Ioffe & Szegedy, 2015), also improve generalization.

Recent advances in deep learning include architectures such as convolutional neural networks (CNNs) (LeCun et al., 1998), recurrent neural networks (RNNs) (Hochreiter & Schmidhuber, 1997), and transformer models (Vaswani et al., 2017), each designed for specific data modalities. These innovations, combined with improved training and regularization techniques, have led to state-of-the-art performance across various domains.

Challenges in Enforcing Constraints in Deep Learning Despite these advances, deep learning models inherently operate in an unconstrained setting, making them struggle to enforce domain-specific constraints, such as physical laws and fairness criteria on their outputs.

Several approaches address this limitation by adding additional post-processing steps or using specialized architectures. Projection-based methods (Márquez-Neila et al., 2017) project the unconstrained output onto the feasible set after inference. Regularization techniques (Pathak et al., 2015) incorporate soft penalties into the loss function to encourage the satisfaction of constraints. Alternatively, constrained neural architectures explicitly integrate constraints into the model structure, such as Lagrangian-based approaches (Fioretto et al., 2020a).

2.3. Predict-then-Optimize

The Predict-Then-Optimize (PTO) framework has emerged as a critical paradigm in operations research, machine learning, and decision-making under uncertainty. This framework consists of two sequential stages: (1) predicting uncertain parameters using machine learning (ML) models and (2) solving an optimization problem based on the predicted parameters to make decisions.

In this setting, the decision-making processes can be described by parametric CO problems, defined as:

$$\boldsymbol{x}^\star(\boldsymbol{c}) = \underset{\boldsymbol{x}\in\mathcal{S}_{\boldsymbol{c}}}{\operatorname{argmax}} \quad f_{\boldsymbol{c}}(\boldsymbol{x}) \tag{2.3}$$

The goal is to predict the unknown parameters $\hat{\boldsymbol{c}}$ from empirical data using supervised learning, such that the optimal solution $\boldsymbol{x}^\star(\hat{\boldsymbol{c}})$ best matches a targeted optimal solution $\boldsymbol{x}^\star(\boldsymbol{c})$, as measured by a task-specific loss function. In this framework, the parameters $\boldsymbol{c}$ are predicted from the features $\boldsymbol{z}$ using an ML model $\mathcal{M}_\theta$, parameterized by θ. The empirical data χ captures the correlation between features and targeted solutions to (2.3) for some $\boldsymbol{c}$.

Early approach to PtO, known as two-stage learning, treats prediction and optimization as independent stages. In the first stage, an ML model is trained to predict the uncertain parameters using features derived from historical data. The second stage involves solving a constrained optimization problem using the predicted parameters. However, two-stage training assumes that reducing prediction error will directly

translate into improved decision quality, which is not always true. Research has shown that small prediction errors can sometimes lead to large decision errors and vice versa Elmachtoub & Grigas (2021); Mandi et al. (2024a).

To address the limitations of this approach, PtO methods have been developed. PtO trains the ML model directly to optimize the decision quality by minimizing the task loss, which measures the suboptimality of the decisions made using the predicted parameters. A common task loss is regret, defined as:

$$regret(\hat{\boldsymbol{c}}, \boldsymbol{c}) = f_{\hat{\boldsymbol{c}}}(\boldsymbol{x}^\star(\hat{\boldsymbol{c}})) - f_{\boldsymbol{c}}(\boldsymbol{x}^\star(\boldsymbol{c})).$$

where $\boldsymbol{x}^\star(\boldsymbol{c})$) is the optimal decision under the under the true parameters $\boldsymbol{c}$, and $\boldsymbol{x}^\star(\hat{\boldsymbol{c}})$) is the decision induced by the predicted parameters $\hat{\boldsymbol{c}}$

PtO techniques can be categorized into two broad classes: (i) gradient-free techniques, which include tree-based methods or explicitly specified models such as linear models Elmachtoub et al. (2020); Jeong et al. (2022), and ii) gradient-based techniques. These are the preferred approaches for training neural networks and involve differentiating through the optimization problem to compute gradients of the task loss with respect to the ML model parameters $\mathcal{M}_\theta$ Amos & Kolter (2017a); Agrawal et al. (2019a). This thesis focuses on gradient-based techniques.

Gradient-Based PtO Techniques

Gradient-based PtO techniques require differentiating through the optimization problem. This can be challenging due to the non-differentiability of many optimization problems, especially those involving discrete variables. To address this, several techniques have been developed to approximate or smooth the optimization mapping. These techniques can be grouped into four categories:

1. Analytical Differentiation of Optimization Mappings: This approach computes exact derivatives by differentiating the optimality conditions of certain optimization problems, for which the deriva-

tive exists and is non-zero. For example, Gould et al. (2016) proposed implicit differentiation for unconstrained convex problems, while subsequent work extended this to constrained problems using KKT conditions Amos & Kolter (2017b); Amos et al. (2019). General-purpose differentiable solvers, such as `cvxpy` Agrawal et al. (2019a), leverage convex cone programming to enable differentiation for broad classes of convex problems. Additionally, Kotary et al. (2023) studied the relationship between unrolling and differentiation of the fixed-point conditions, showing that backpropagation by unrolling is equivalent to solving the linear system by fixed-point iteration.

2. Analytical Smoothing of Optimization Mappings: For with combinatorial optimization problems (for which the analytical derivatives are zero almost everywhere), Smoothing techniques approximate problems in which the gradient can be differentiated analytically. Common methods include adding smooth regularizers such as Euclidean norms or entropy functions Wilder et al. (2019); Ferber et al. (2020); Mandi & Guns (2020), enabling end-to-end learning for discrete structures such as ranking policies and shortest paths Kotary et al. (2022); Elmachtoub & Grigas (2021).

3. Smoothing by Random Perturbations: This approach uses random perturbations to construct smooth approximations of the optimization mapping. For instance, Berthet et al. (2020) introduced a method to differentiate linear programs by adding noise to the cost vector and computing the expected solution. Similarly, Pogančić et al. (2019) approximates gradients using finite differences.

4. Differentiation of Surrogate Loss Functions, which approximate task-specific losses like regret, provide easy-to-compute gradients or subgradients for training Elmachtoub & Grigas (2021).

Challenges in PtO Despite its advantages, PtO faces several challenges:

- Scalability: Solving large-scale optimization problems repeatedly during training can be computationally expensive.

- Multistage Decision-Making: Applying PtO to sequential decision-making problems remains an active area of research.
- Robustness and Risk-Sensitivity: Developing methods that handle worst-case scenarios and uncertain constraints is critical for real-world applications

This thesis focuses on addressing the challenges of scalability and robustness in PtO, with particular emphasis on risk-aware decision-making.

2.4. Learning-to-Optimize

This section reviews the literature on *learning CO solutions*, focusing on techniques that incorporate constraints into the learning process to predict feasible or near-feasible solutions to both continuous and discrete CO problems.

Data-Driven Learning of CO Solutions A diverse body of work of research has explored ML architectures that predict fast, approximate solutions to predefined CO problems. These methods employ supervised learning on dataset of solved instances or execution traces, where each dataset $\boldsymbol{\chi} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$ consists of problem instance specification $\boldsymbol{x}_i$ and corresponding optimal solution $\boldsymbol{y}_i$ obtained from a solver. Inputs $\boldsymbol{x}_i$ encode problem-specific parameters, such as matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ defining linear constraints in LPs. Notably, each sample in the dataset may correspond to a different problem instance with varying objective function coefficients and constraints.

One of early approaches to apply ML for learning CO problem solutions was the work of Hopfield & Tank (1985), which used Hopfield Networks Hopfield (1982) to solve the traveling salesman problem (TSP). The approach modified energy functions to emulate the objective of TSP while enforcing feasibility using Lagrange multipliers. However, Wilson & Pawley (1988) later demonstrated that this approach suffers from several limitations, including sensitivity to parameter initialization and difficulty tuning hyperparameters. As noted in Bello et al. (2017), modern deep learning techniques have largely replaced such approaches.

More recent frameworks explicitly model constraint satisfaction within the loss function of a neural network to improve feasibility and generalization. One line of work leverages Lagrangian duality to guide the solution prediction to better satisfy the problem's constraints. These methods have been successfully applied to continuous NLPs, including energy optimization (Fioretto et al., 2020c; Velloso & Van Hentenryck, 2020) and constrained prediction, such as transprecision computing and fair classification (Fioretto et al., 2020a; Tran et al., 2021).

Another promising direction involves iterative learning strategies that refine predicted solutions by leveraging external solvers. For example, Detassis et al. (2020) proposed an feedback framework where a solver adjusts targeted solutions to more better match model predictions while maintaining feasibility, reducing the degree of constraint violation in the model predictions in subsequent iterations.

Learning Solutions for Graph-Structured CO Problems Beyond general unstructured CO problems, a significant line of research focuses on learning solutions for graph-based CO problems. Advances in deep learning architectures such as sequence models, attention mechanisms, and graph neural networks (GNNs), have provided substantial improvements in solving combinatorial optimization tasks on graphs.

Vinyals et al. (2015) introduced the *pointer network*, a sequence-to-sequence model using an encoder-decoder architecture paired with an attention mechanism to generate structured permutations over variable-sized inputs. The model was trained in a supervised manner to learn solutions for the TSP and Delaunay triangulation problems, demonstrating some ability to generalize over variable-sized problem instances. Expanding on this, Bello et al. (2017) adopted the pointer network architecture but trained it using RL, using tour length as the reward signal. The move from supervised to RL was motivated by the difficulties of obtaining optimal solutions and the existence of multiple optimal solutions.

Kool et al. (2018) further improved learning efficiency by applying an attention-based RL model to the TSP and variants of the vehicle routing problem. Their approach employs a graph attention network (Velickovic et al., 2018) inspired by the Transformer architecture (Vaswani et al.,

2017). This neural network design introduces invariance to permutations of the input nodes, improving learning efficiency.

While RL-oriented frameworks have gained popularity, Nowak et al. (2018) demonstrated that supervised learning can still be an effective method for the general quadratic assignment problem. Their approach used GNNs trained on individual problem instances and their targeted solutions, predicting permutations that are then converted into feasible solutions by a beam search. For a comprehensive survey on combinatorial optimization and reasoning with GNNs, see Cappart et al. (2021), which provides an in-depth review of recent developments in this field.

CHAPTER 3

End-to-End Learning for Fair Multiobjective Optimization Under Uncertainty

"The worst form of inequality is to try to make unequal things equal."

Aristotle

Decision-making often operates under uncertainty, where the Predict-Then-Optimize (PtO) framework provides a structured approach by learning unknown parameters from data and solving an optimization problem based on these estimates. This chapter extends PtO to a multiobjective setting, where conflicting objectives must be optimized while ensuring fairness, as seen in real-world problems like resource allocation. To address this, we propose a PtO framework for Ordered Weighted Averaging (OWA) optimization, which offers fairness guarantees across multiple objectives. However, OWA's nondifferentiability poses a challenge for gradient-based learning in ML models. We overcome this limitation by introducing a differentiable approximation of OWA, enabling end-to-end training for fair and scalable decision-making. This chapter is based on the author's publications Dinh et al. (2024a,b).

3.1. End-to-End Learning for Fair Multiobjective Optimization Under Uncertainty

The *Predict-Then-Optimize* (PtO) framework Mandi et al. (2024a) models decision-making processes as optimization problems with unspecified parameters $\boldsymbol{c}$, which must be estimated by a machine learning (ML) model, given correlated features $\boldsymbol{z}$. An estimation of $\boldsymbol{c}$ completes the problem's specification, whose solution defines a mapping:

$$\boldsymbol{x}^\star(\boldsymbol{c}) = \underset{\boldsymbol{x}\in\mathcal{S}}{\operatorname{argmax}} \quad f(\boldsymbol{x}, \boldsymbol{c}) \tag{3.1}$$

The goal is to learn a model $\hat{\boldsymbol{c}} = \mathcal{M}_\theta(\boldsymbol{z})$ from observable features $\boldsymbol{z}$, such that the objective value $f(\boldsymbol{x}^\star(\hat{\boldsymbol{c}}), \boldsymbol{c})$ under ground-truth parameters $\boldsymbol{c}$ is maximized on average. This is common in many applications requiring decision-making under uncertainty, like planning the fastest route through a city with unknown traffic delays or predicting optimal power generation schedules based on demand forecasts.

Optimization of multiple objectives is crucial in contexts requiring a balance of competing goals, especially when fairness is essential in fields like energy systems Terlouw et al. (2019), urban planning Salas & Yepes (2020), and multi-objective portfolio optimization Iancu & Trichakis

(2014); Chen & Zhou (2022). A common approach is using Ordered Weighted Averaging (OWA) Yager (1993) to achieve Pareto-optimal solutions that fairly balance each objective. However, optimizing an OWA objective in PtO is challenging due to its nondifferentiability, which prevents backpropagation through $\boldsymbol{x}^\star(\boldsymbol{c})$ within machine learning models trained by gradient descent. To our knowledge, no prior PtO models encounter a non-differentiable objective, making this challenge novel.

3.2. Preliminaries

3.2.1 Fair OWA and its Optimization

The *Ordered Weighted Average* (OWA) operator Yager (1993) is used in various decision-making fields to fairly aggregate multiple objective criteria Yager & Kacprzyk (2012). Let $\mathbf{y} \in \mathbb{R}^m$ be a vector of m distinct criteria, and $\tau : \mathbb{R}^m \to \mathbb{R}^m$ be the sorting map that orders $\mathbf{y}$ in increasing order. For any $\mathbf{w}$ satisfying $\mathbf{w} \in \mathbb{R}^m$, $\sum_i w_i = 1$, and $\mathbf{w} \geq 0$, the OWA aggregation with weights $\mathbf{w}$ is piecewise-linear in $\mathbf{y}$ Ogryczak & Śliwiński (2003):

$$\mathrm{OWA}_{\mathbf{w}}(\mathbf{y}) = \mathbf{w}^T \tau(\mathbf{y}), \tag{3.2}$$

this thesis uses its concave version, *Fair OWA* Ogryczak et al. (2014), characterized by weights in descending order: $w_1 > \ldots > w_n > 0$.

The following three properties of Fair OWA functions are crucial for fairly optimizing multiple objectives: **(1)** *Impartiality*: Permutations of a utility vector are equivalent solutions. **(2)** *Equitability*: Marginal transfers from a higher value criterion to a lower one increase the OWA aggregated value. **(3)** *Monotonicity*: $\mathrm{OWA}_{\mathbf{w}}(\mathbf{y})$ is an increasing function of each element of $\mathbf{y}$. This ensures that solutions optimizing the OWA objectives are Pareto Efficient, meaning no criterion can be improved without worsening another Ogryczak & Śliwiński (2003). Optimization of aggregation functions that possess these properties leads to *equitably efficient solutions*, which satisfy a rigorously defined notion of fairness Kostreva & Ogryczak (1999).

3.2.2 Predict-Then-Optimize Learning

Our problem setting fits within the PtO framework. Generally, a parametric optimization problem (3.1) models an optimal decision $\boldsymbol{x}^\star(\boldsymbol{c})$ with respect to unknown parameters $\boldsymbol{c}$ drawn from a distribution $\boldsymbol{c} \sim \mathcal{C}$. While the true value of $\boldsymbol{c}$ is unknown, correlated feature values $\boldsymbol{z} \sim \mathcal{Z}$ can be observed. The goal is to learn a predictive model $\mathcal{M}_\theta : \mathcal{Z} \to \mathcal{C}$ from features $\boldsymbol{z}$ to estimate problem parameters $\hat{\boldsymbol{c}} = \mathcal{M}_\theta(\boldsymbol{z})$, by maximizing the empirical objective value of the resulting solution under ground-truth parameters. That is,

$$\underset{\theta}{\operatorname{argmax}} \quad \mathbb{E}_{(\boldsymbol{z},\boldsymbol{c})\sim\Omega}\, f\left(\boldsymbol{x}^\star(\mathcal{M}_\theta(\boldsymbol{z})), \boldsymbol{c}\right), \tag{3.3}$$

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

The above training goal is often achieved by maximizing empirical *Decision Quality* as a loss function Mandi et al. (2024a), defined:

$$\mathcal{L}_{DQ}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = f\left(\boldsymbol{x}^\star(\hat{\boldsymbol{c}}), \boldsymbol{c}\right). \tag{3.4}$$

Gradient descent training of (3.3) with $\mathcal{L}_{DQ}$ requires a model of gradient $\frac{\partial \mathcal{L}_{DQ}}{\partial \hat{\boldsymbol{c}}}$, either directly or through chain-rule composition $\frac{\partial \mathcal{L}_{DQ}}{\partial \hat{\boldsymbol{c}}} = \frac{\partial \boldsymbol{x}^\star(\hat{\boldsymbol{c}})}{\partial \hat{\boldsymbol{c}}} \cdot \frac{\partial \mathcal{L}_{DQ}}{\partial \boldsymbol{x}^\star}$. When $\boldsymbol{x}^\star$ is not differentiable, as in OWA optimizations, smooth approximations are required, such as those developed in the next section.

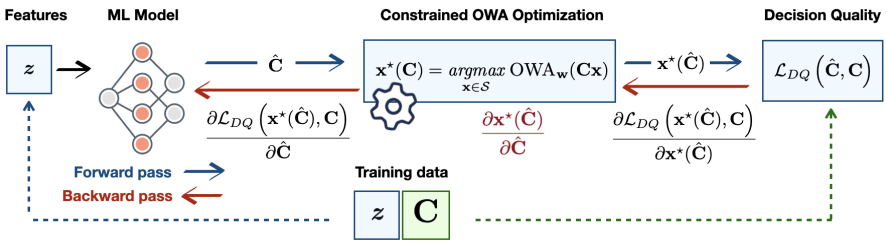


Figure 3.1: Predict-Then-Optimize for OWA Optimization.

3.3. End-to-End Learning with Fair OWA Optimization

this thesis focuses on scenarios where the objective function f is an ordered weighted average of m linear objective functions, each parameterized by a row of a matrix $\boldsymbol{C} \in \mathbb{R}^{m \times n}$ so that $f(\boldsymbol{x}, \boldsymbol{C}) = \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x})$ and

$$\boldsymbol{x}^\star(\boldsymbol{C}) = \underset{\boldsymbol{x} \in \mathcal{S}}{\operatorname{argmax}} \ \ \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x}). \tag{3.5}$$

Note that this methodology extends to cases where the OWA objective is combined with additional smooth terms. For simplicity, the exposition primarily focuses on the pure OWA objective as shown in equation (3.5).

The goal is to learn a prediction model $\hat{\boldsymbol{C}} = \mathcal{M}_\theta(\boldsymbol{z})$ that maximizes decision quality through gradient descent on problem (3.3), which requires obtaining its gradients w.r.t. $\hat{\boldsymbol{C}}$:

$$\frac{\partial \mathcal{L}_{DQ}(\hat{\boldsymbol{C}}, \boldsymbol{C})}{\partial \hat{\boldsymbol{C}}} = \underbrace{\frac{\partial \boldsymbol{x}^\star}{\partial \hat{\boldsymbol{C}}}}_{\boldsymbol{J}} \cdot \underbrace{\frac{\partial \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x}^\star)}{\partial \boldsymbol{x}^\star}}_{\boldsymbol{g}}, \tag{3.6}$$

where $\boldsymbol{x}^\star$ is evaluated at $\hat{\boldsymbol{C}}$. The main strategy involves determining the OWA function's gradient $\boldsymbol{g}$ and then computing $\boldsymbol{J}\boldsymbol{g}$ by backpropagating $\boldsymbol{g}$ through $\boldsymbol{x}^\star$. A schematic illustration highlighting the forward and backward steps required for this process is provided in Figure 3.1.

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

$$\frac{\partial}{\partial \boldsymbol{y}} \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{y}) = \boldsymbol{w}_{(\sigma^{-1})} \tag{3.7}$$

where σ are the sorting indices on $\boldsymbol{y}$ Do & Usunier (2022). Based on this formula, computing an overall subgradient $\boldsymbol{g} = {}^{\partial}/_{\partial \boldsymbol{x}} \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x})$ is a routine application of the chain rule (via automatic differentiation). The use of OWA subgradients (3.7) in training ML models has been previously explored in the context of reinforcement learning Siddique et al. (2020). This work primarily relies on these subgradients to optimize OWA as a loss function by gradient descent. Section 3.5.2 separately investigates the use of OWA's Moreau envelope as a differentiable loss

function, and finds that it yields similar results.

3.4. Differentiable Approximations of OWA Optimization

This section develops two alternative differentiable approximations of the OWA optimization mapping (3.5). Prior works Wilder et al. (2019); Amos et al. (2019) show that when an optimization mapping (3.1) is discontinuous, as is the case when f and $\mathcal{S}$ define a linear program (LP), differentiable approximations to (3.1) can be formed by regularization of its objective by smooth functions. Section 3.4.1 will demonstrate how linear programming models of OWA optimization can be combined with smoothing techniques for LP, yielding effective differentiable approximations of (3.5).

However, this model becomes computationally intractable for more than a few criteria m. An efficient alternative is proposed in Section 3.4.2, where the mapping (3.5) is made differentiable by replacing the OWA objective with its smooth Moreau envelope approximation. To the best of the author's knowledge, this is the first time that objective smoothing via the Moreau envelope is used (and shown be an effective technique) for approximating nondifferentiable optimization programs in end-to-end learning. As approximations of the true mapping (3.5), both smoothed models are used employed in training and replaced by (3.5) at test time, similarly to a softmax layer in classification.

3.4.1 OWA LP with Quadratic Smoothing

The mainstay approach to solve problem (3.5) when $\boldsymbol{x} \in \mathcal{S}$ is linear is to transform the problem into a linear program without OWA functions and solve it with a simplex method Ogryczak & Śliwiński (2003). Our first approach to differentiable OWA optimization combines this transformation with the smoothing technique of Wilder et al. (2019), which forms differentiable approximations to linear programs

$$\boldsymbol{x}^\star(\boldsymbol{c}) = \operatorname{argmax}_{\boldsymbol{Ax} \leq \boldsymbol{b}} \ \boldsymbol{c}^T \boldsymbol{x} \tag{3.8}$$

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

We adopt a version of this technique to OWA optimization (3.5) by first forming an equivalent LP problem. It is observed in Ogryczak & Śliwiński (2003) that $\text{OWA}_{\boldsymbol{w}}$ can be expressed as the minimum weighted average among all permutations of the OWA weights $\boldsymbol{w}$:

$$OWA_{\mathbf{w}}(\mathbf{r}) = \max_z\ z \quad \text{subject to} \quad z \leq \mathbf{w}_\sigma \cdot \mathbf{r} \quad \forall\sigma \in \mathcal{P}, \tag{3.9}$$

which allows the OWA optimization (3.5) to be expressed as

$$\boldsymbol{x}^\star(\boldsymbol{C}) = \text{argmax}_{\boldsymbol{x}\in\mathcal{S},\boldsymbol{y},z}\ z \tag{3.10a}$$

$$\text{subject to:}\quad \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \tag{3.10b}$$

$$z \leq \boldsymbol{w}_\tau \cdot \boldsymbol{y} \quad \forall\tau \in \mathcal{P}_m. \tag{3.10c}$$

When the constraints $\boldsymbol{x} \in \mathcal{S}$ are linear, problem (3.10) is a LP. However, its constraints (3.10c) grows factorially as $m!$, where m is the number of individual objective criteria aggregated by OWA. Smoothing by the scaled norm of joint variables $\boldsymbol{x}, \boldsymbol{y}, z$ leads to a differentiable QP approximation, viable when m is small. This optimization can be solved and differentiated using techniques from Amos & Kolter (2017a) or a generic differentiable optimization solver such as Agrawal et al. (2019a):

$$\boldsymbol{x}^\star(\boldsymbol{C}) = \underset{\boldsymbol{x}\in\mathcal{S},\boldsymbol{y},z}{\text{argmax}}\ z + \epsilon\left(\|\boldsymbol{x}\|_2^2 + \|\boldsymbol{y}\|_2^2 + z^2\right) \tag{3.11a}$$

$$\text{subject to:}\quad (3.10\text{b}), (3.10\text{c}). \tag{3.11b}$$

Due to the huge number of constraints, this alternative LP form of OWA optimization is mostly of theoretical significance and should only be applied when m is small. To reduce the number of constraints, an alternative LP formulation of OWA optimization is proposed by Ogryczak & Śliwiński (2003). It represents each ordered outcome by the differences between cumulative sums, expressed as $\tau_i(\boldsymbol{y}) = \bar{\tau}_i(\boldsymbol{y}) - \bar{\tau}_{i-1}(\boldsymbol{y})$ for $i = 2, \ldots, m$ where $\bar{\tau}_k(\boldsymbol{y}) = \sum_{j=1}^k \tau_j(\boldsymbol{y})$. The OWA problem (3.5) can thereby be expressed in the form:

$$OWA(\boldsymbol{x}) = \min \sum_{i=1}^{m} w'_i \bar{\tau}_i(\boldsymbol{C}\boldsymbol{x}), \tag{3.12}$$

where coefficients w_i are defined as $w'_i = w_i - w_{i+1}$ for $i = 1, 2, \cdots, m-1$ and $w'_m = w_m$.

This leads to the following LP formulation of the OWA optimization (3.5):

$$\begin{aligned}
\boldsymbol{x}^\star(\boldsymbol{C}) = \text{argmax}_{\boldsymbol{x}\in\mathcal{S},\boldsymbol{y},\boldsymbol{r},\boldsymbol{d}} \quad & \sum_{k=1}^{m} k w'_k r_k - \sum_{k=1}^{m}\sum_{i=1}^{m} w'_k d_{ik} && \text{(3.13a)}\\
\text{subject to:} \quad & \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} && \text{(3.13b)}\\
& d_{ik} \geq r_k - y_i \quad \textit{for} \quad i,k = 1, \cdots, m && \text{(3.13c)}\\
& d_{ik} \geq 0 \quad \textit{for} \quad i,k = 1, \cdots, m. && \text{(3.13d)}
\end{aligned}$$

Again when $x \in \mathcal{S}$ are linear, problem (3.13) is an LP with m^2+m+p constraints. Quadratic smoothing can then be applied as in (3.11):

$$\begin{aligned}
\boldsymbol{x}^\star(\boldsymbol{C}) = \text{argmax}_{\boldsymbol{x}\in\mathcal{S},\boldsymbol{y},\boldsymbol{r},\boldsymbol{d}} \quad & \sum_{k=1}^{m} k w'_k r_k - \sum_{k=1}^{m}\sum_{i=1}^{m} w'_k d_{ik} + && \text{(3.14a)}\\
& \epsilon \left(\|\boldsymbol{x}\|_2^2 + \|\boldsymbol{y}\|_2^2 + \|\boldsymbol{r}\|_2^2 + \|\boldsymbol{d}\|_2^2 \right) && \text{(3.14b)}\\
\text{subject to:} \quad & \text{(3.13b), (3.13c), (3.13d).} && \text{(3.14c)}
\end{aligned}$$

Though the number of constraints is far less than in (3.10), it still grows polynomially, and can only solved efficiently when m numbers up to a few dozen. The main *disadvantage* of both QP-smoothing models is poor scalability in the number of criteria m, due to constraints (3.10c) or (3.13c). We note that in each case, the transformed QP is much harder to solve than the associated LP problems. These drawbacks motivate the next smoothing method, which yields a more tractable optimization problem by replacing the OWA objective itself with a smooth Moreau Envelope approximation.

3.4.2 Moreau Envelope Smoothing

In light of the efficiency challenges faced by (3.11), we propose an alternative smoothing technique to form more scalable differentiable approximations of the optimization mapping (3.5). Instead of adding a quadratic term as in (3.11), we replace the piecewise linear function $\text{OWA}_{\boldsymbol{w}}$ in (3.5) with its Moreau envelope, defined for a convex function f as:

$$f^{\beta}(\mathbf{x}) = \min_{\mathbf{v}} \; f(\mathbf{v}) + \frac{1}{2\beta}\|\mathbf{v} - \mathbf{x}\|^2. \tag{3.15}$$

Moreau envelopes of concave functions are defined analogously. Compared to its underlying function f, the Moreau envelope is $\frac{1}{\beta}$ smooth while sharing the same (unconstrained) optima Beck (2017). The Moreau envelope-smoothed OWA optimization problem is then

$$\boldsymbol{x}^{\star}(\boldsymbol{C}) = \text{argmax}_{\boldsymbol{x}\in\mathcal{S}} \; \text{OWA}^{\beta}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x}). \tag{3.16}$$

With its smooth objective function, problem (3.16) can be solved by gradient-based optimization methods, such as projected gradient descent, or more likely a Frank-Wolfe method if $\boldsymbol{x} \in \mathcal{S}$ is linear (see Section 3.5.1). More importantly, it can be effectively backpropagated for end-to-end training.

Backpropagation of (3.16) is nontrivial since its objective function lacks a closed form. To proceed, we first note from Do & Usunier (2022) that the gradient of the Moreau envelope $\text{OWA}^{\beta}_{\boldsymbol{w}}$ is equal to a Euclidean projection:

$$\frac{\partial}{\partial \boldsymbol{x}}\text{OWA}^{\beta}_{\boldsymbol{w}}(\mathbf{x}) = \text{proj}_{\mathcal{C}(\tilde{\boldsymbol{w}})}\left(\frac{\boldsymbol{x}}{\beta}\right), \tag{3.17}$$

where $\tilde{\boldsymbol{w}} = -(w_m, \ldots, w_1)$ and the permutahedron $\mathcal{C}(\tilde{\boldsymbol{w}})$ is the convex hull of all permutations of $\tilde{\boldsymbol{w}}$. It's further shown in Blondel et al. (2020) how such a projection can be both *computed* and *differentiated* using isotonic regression in $\mathcal{O}(m \log m)$ time. To leverage a differentiable projection (3.17) for backpropagation of the overall optimization (3.16), we model its Jacobian by differentiating the fixed-point conditions of a gradient-based solution method.

Letting $\mathcal{U}(\boldsymbol{x}, \boldsymbol{C}) = \mathit{proj}_{\mathcal{S}}(\boldsymbol{x} - \alpha \cdot \frac{\partial}{\partial \boldsymbol{x}}\text{OWA}^{\beta}_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{C}))$, a projected gradient descent step on (3.16) is $\boldsymbol{x}^{k+1} = \mathcal{U}(\boldsymbol{x}^k, \boldsymbol{C})$. Differentiating the

fixed-point conditions of convergence where $\boldsymbol{x}^k = \boldsymbol{x}^{k+1} = \boldsymbol{x}^\star$, and rearranging terms yields a linear system for $\frac{\partial \boldsymbol{x}^\star}{\partial \boldsymbol{C}}$:

$$\left(I - \underbrace{\frac{\partial \mathcal{U}(\boldsymbol{x}^\star, \boldsymbol{C})}{\partial \boldsymbol{x}^\star}}_{\boldsymbol{\Phi}} \right) \frac{\partial \boldsymbol{x}^\star}{\partial \boldsymbol{C}} = \underbrace{\frac{\partial \mathcal{U}(\boldsymbol{x}^\star, \boldsymbol{C})}{\partial \boldsymbol{C}}}_{\boldsymbol{\Psi}} \tag{3.18}$$

The partial Jacobian matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ above can be found given a differentiable implementation of $\mathcal{U}$. This is achieved by computing the inner gradient $\frac{\partial}{\partial \boldsymbol{x}} \mathrm{OWA}^{\beta}_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{C})$ via the differentiable permutahedral projection (3.17), and solving the outer projection mapping $\mathrm{proj}_{\mathcal{S}}$ using a generic differentiable solver such as `cvxpy` Agrawal et al. (2019a). As such, applying $\mathcal{U}$ at a precomputed solution $\boldsymbol{x}^\star(\boldsymbol{C})$ allows $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ to be extracted in PyTorch, in order to solve (3.18); this process is efficiently implemented via the `fold-opt` library Kotary et al. (2023).

3.5. Experiments

This section focuses on evaluating the differentiable approximations of Fair OWA optimization introduced in Section 3.4. A robust Markowitz portfolio optimization problem is chosen for the experimental setting, detailed in 3.5.1. In generic terms we employ a prediction model $\hat{\boldsymbol{C}} = \mathcal{M}_\theta(\boldsymbol{z})$ to jointly estimate, from features $\boldsymbol{z}$, the coefficients $\boldsymbol{C} \in \mathbb{R}^{m \times n}$ of m linear objectives, taken together as $\boldsymbol{C}\boldsymbol{x} \in \mathbb{R}^m$. Its training goal is to maximize empirical decision quality with respect to their Fair OWA aggregation $f(\boldsymbol{x}, \boldsymbol{C}) = \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x})$:

$$\mathcal{L}_{DQ}(\hat{\boldsymbol{C}}, \boldsymbol{C}) = \mathrm{OWA}_{\boldsymbol{w}}\left(\boldsymbol{C}\boldsymbol{x}^\star(\hat{\boldsymbol{C}})\right). \tag{3.19}$$

Any descending OWA weights $\boldsymbol{w}$ can be used to specify (3.19); we choose the squared *Gini indices* $w_j = \left(\frac{n-1+j}{n}\right)^2$.

Evaluation Results in this section are reported in terms of the equivalent *regret* metric of suboptimality, whose minimum value 0 corresponds

to maximum decision quality:

$$\textit{regret}(\hat{\boldsymbol{C}}, \boldsymbol{C}) = \text{OWA}^{\star}_{\boldsymbol{w}}\left(\boldsymbol{C}\boldsymbol{x}^{\star}(\boldsymbol{C})\right) - \text{OWA}_{\boldsymbol{w}}\left(\boldsymbol{C}\boldsymbol{x}^{\star}(\hat{\boldsymbol{C}})\right) \tag{3.20}$$

where $\text{OWA}^{\star}_{\boldsymbol{w}}(\boldsymbol{C})$ is the true optimal value of problem (3.5). This experiment is designed to evaluate the proposed differentiable approximations (3.11), (3.14), and (3.16) of Section 3.4; for reference, they are named **OWA-QP-Fac**, **OWA-QP-Dev**, and **OWA-Moreau** respectively.

Baseline Models In addition to the newly proposed models, the evaluations presented in this section include two main baseline methods:

1. Two-stage Method: This standard baseline (denoted as **Two-stage**) serves as a comparison for Predict-Then-Optimize training (3.3) Mandi et al. (2024a). It trains the prediction model $\hat{\boldsymbol{C}} = \mathcal{M}_{\theta}(\boldsymbol{z})$ by MSE regression, minimizing $\mathcal{L}_{TS}(\hat{\boldsymbol{C}}, \boldsymbol{C}) = \|\hat{\boldsymbol{C}} - \boldsymbol{C}\|^2$ without considering the downstream optimization model, which is employed only at test time.

2. Unweighted Sum (UWS) LP: This baseline aggregates the objective criteria by their unweighted sum, resulting in an LP problem $\boldsymbol{x}^{\star}(\boldsymbol{C}) = \text{argmax}_{\boldsymbol{x}\in\mathcal{S}}\, \mathbf{1}^T(\boldsymbol{C}\boldsymbol{x})$. Quadratic smoothing is applied to the LP problem Wilder et al. (2019), to allow end-to-end training on its decision quality loss. This problem is a linear programming approximation of the true Fair OWA problem, for which existing Predict-Then-Optimize methods including Wilder et al. (2019) may be applied. However, it does not incorporate the OWA objective. It is included as a baseline to provide contrasting results with the Two-Stage, relative to our proposals which enable Predict-Then-Optimize learning on the true Fair OWA problem. Throughout, it is referred to as **Sum-QP**.

3.5.1 Robust Markowitz Portfolio Problem

The classic Markowitz portfolio problem is concerned with constructing an optimal investment portfolio, given future returns $\boldsymbol{c} \in \mathbb{R}^n$ on n assets, which are unknown and predicted from exogenous data. The goal is to

maximize future return, while managing the overall risk taken by the asset allocation. One common notion of risk is the covariance among assets, which can be limited with a convex quadratic constraint. Define the set of valid fractional allocations $\Delta_n = \{\boldsymbol{x} \in \mathbb{R}^n : \mathbf{1}^T\boldsymbol{x} = 1, \boldsymbol{x} \geq 0\}$, then:

$$\boldsymbol{x}^\star(\boldsymbol{c}) = \underset{\boldsymbol{x}\in\Delta_n}{\operatorname{argmax}} \ \boldsymbol{c}^T\boldsymbol{x} \quad \text{subject to:} \quad \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x} \leq \delta. \tag{3.21}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{n\times n}$ are the price covariances over n assets.

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

$$\boldsymbol{x}^\star(\boldsymbol{C}) = \underset{\boldsymbol{x}\in\Delta_n}{\operatorname{argmax}} \ \operatorname{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x}). \tag{3.22}$$

The following experiment shows how to integrate robust portfolio optimization (3.22) end-to-end with per-scenario price prediction $\hat{\boldsymbol{C}} = \mathcal{M}_\theta(\boldsymbol{z})$, in order to maximize the risk-aware OWA objective value.

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 $\boldsymbol{c}_i$ 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 $\boldsymbol{c}_i$ to obtain $\boldsymbol{C}_i \in \mathbb{R}^{m\times n}$. While future asset prices can be predicted based on various exogenous data including past prices or sentiment analysis, this experiment generates feature vectors $\boldsymbol{z}_i$ using a randomly generated nonlinear feature mapping. The experiment is replicated in three settings which assume $m = 3$, 5, and 7 scenarios.

Two sets of stocks were selected to generate two different datasets based on their average returns across observations. The first set consists of assets from the index with average returns within the 25th to 75th quantile range, while the second set includes assets from the 75th quantile.

Table 3.1: Model's Hyperparameter Settings in Portfolio Problems

Hyperparameter	Min	Max	Final Value				
			OWA-LP	Two-Stage	Sum-QP	OWA-QP-Fac	OWA-Moreau
learning rate	$1e^{-3}$	$1e^{-1}$	$1e^{-2}$	$5e^{-3}$	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$
smoothing parameter ϵ	0.1	1.0	N/A	N/A	1.0	1.0	N/A
smoothing parameter β_0	0.005	0.1	N/A	N/A	N/A	N/A	0.05
MSE loss weight λ	0.1	0.5	0.4	N/A	0.3	0.4	0.1

The predictive model $\mathcal{M}_\theta$ is a feedforward neural network. A neural network (NN) with three shared hidden layers followed by one separated hidden layer for each species is trained using Adam Optimizer and with a batch size 64. The size of each shared layer is halved, and the output dimension of the separated layer is equal to the number of assets. Hyperparameters were selected as the best-performing on average among those listed in Table 3.1). 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 250, 500, and 750 iterations for scenarios with 3, 5, and 7 inputs. The update step size is set to $\gamma = 0.02$. At test time, $\mathcal{M}_\theta$ is evaluated over a test set for the distribution $(\boldsymbol{z}, \boldsymbol{C}) \in \Omega$, by passing its predictions to an LP solver of (3.22).

For the Moreau-envelope smoothed OWA optimization (3.16) proposed for end-to-end training, the main difference is that its objective function is differentiable (with gradients (3.17)), which allows solution by a more efficient Frank-Wolfe method Beck (2017), whose inner optimization over Δ 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:

$$\boldsymbol{x}^{k+1} = \frac{k}{k+2}\boldsymbol{x}^k + \frac{2}{k+2}\operatorname{argmax}\left(\frac{\partial}{\partial \boldsymbol{x}}\mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x}^k)\right) \tag{3.23}$$

Results Figure 3.2 shows percent regret in the OWA objective attained on average over the test sets (lower is better). The y-axis represents the percentage of regret based on optimal OWA values. A consis-

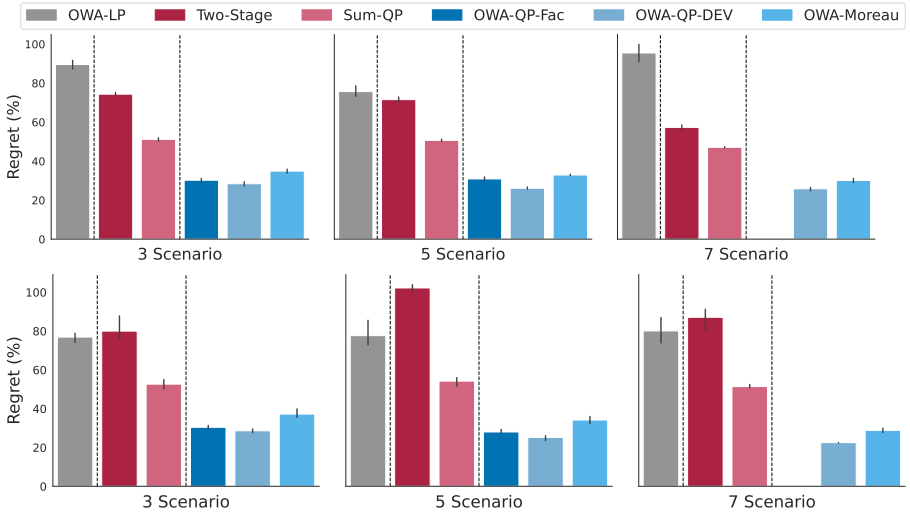


Figure 3.2: Percentage OWA regret (lower is better) on test set, on robust portfolio problem over 3,5,7 scenarios. Performance measured on datasets generated from assets with averages within the 25th-75th quantile (top) and returns above the 75th quantile (bottom) respectively.

tent trend is observed in both datasets: end-to-end approaches (all the bars except **Two-stage**) tend to outperform two-stage approaches. Additionally, our proposed frameworks (**OWA-QP-Fac**, **OWA-QP-Dev**, and **OWA-Moreau**) perform better than **Sum-QP**, with improvements ranging from 5-30%. Both versions of QP smoothing (**OWA-QP-Fac** and **OWA-QP-Dev**) perform well when the number of scenarios is small but face scalability challenges, highlighting the importance of the proposed Moreau envelope smoothing technique in addressing these limitations (Section 3.4.2).

OWA-LP represents a baseline method where the OWA's equivalent linear program (LP) is used as a differentiable optimization without smoothing. For comparison, the grey bars indicate the results from a non-smoothed OWA LP (3.9) implemented with implicit differentiation in `cvxpylayers` Agrawal et al. (2019a). This comparison highlights the improvement in accuracy due to applying quadratic smoothing in **OWA-**

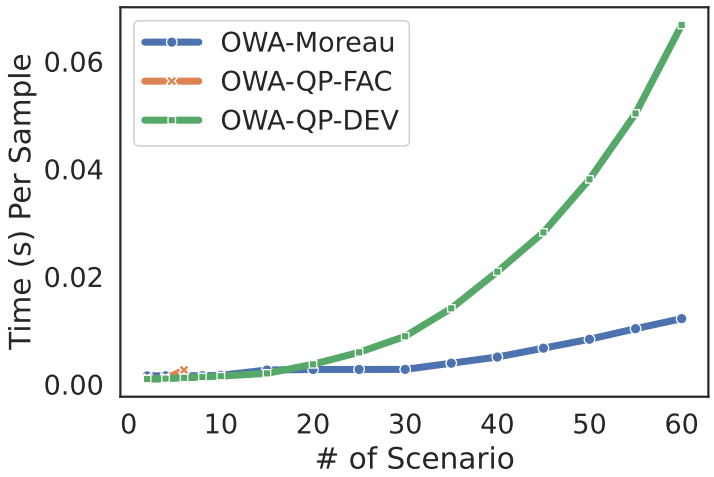


Figure 3.3: Average solving time of 3 smoothed OWA optimization models, on Robust Portfolio Optimization, over 1000 input samples. Missing data points past 7 scenarios on OWA-QP-Fac are due to memory overflow as the number of constraints grows factorially.

QP-Fac and **OWA-QP-Dev**. The poor performance of the OWA subgradient training under the non-smoothed **OWA-LP** demonstrates that the proposed approximations in Section 3.4 are necessary for accurate training.

Runtime of the smoothed models (3.11), (3.14) and (3.16) are compared in Figure 3.3. These results show that the Moreau envelope smoothing maintains low runtime as m increases, while **OWA-QP-Fac** approximation suffers past $m = 5$ and causes memory overflow beyond $m = 6$. On the other hand, **OWA-QP-Dev** demonstrates faster runtimes compared to the original model, though it still struggles when $m > 25$.

3.5.2 Moreau Envelope as a Loss Function

A sensible alternative to subgradients is to use the gradients of the OWA Moreau envelope (3.17). That is, we hypothesize that the Moreau enve-

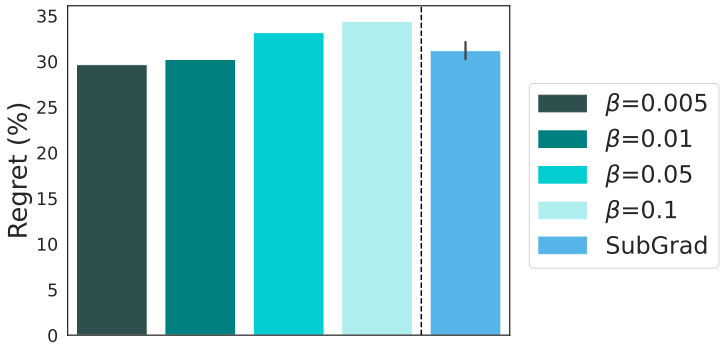


Figure 3.4: Effect of training with OWA as DQ Loss vs its Moreau envelope. The first four bars represent regret on OWA DQ loss, due to training with gradients of its Moreau Envelope under various smoothing hyperparameters β. The rightmost bar shows regret on OWA loss due to training with subgradients. Results reported on portfolio problem with 7 scenarios, described in Section 3.5.

lope $\text{OWA}\boldsymbol{w}^{\beta}$ can also be employed as decision quality loss in equation (3.6) by setting $\boldsymbol{g} = \partial/\partial \boldsymbol{x}, \text{OWA}^{\beta}_{\boldsymbol{w}}(\boldsymbol{C}\boldsymbol{x})$. As discussed in Section 5, this gradient is equal to the projection of an input vector $\mathbf{x}$ onto the permutahedron $\mathcal{C}(\tilde{\boldsymbol{w}})$ induced by the OWA weights $\boldsymbol{w}$:

$$\frac{\partial}{\partial \boldsymbol{x}} \text{OWA}^{\beta}_{\boldsymbol{w}}(\mathbf{x}) = \text{proj}_{\mathcal{C}(\tilde{\boldsymbol{w}})}\left(\frac{\boldsymbol{x}}{\beta}\right), \tag{3.24}$$

Using Propositions 2 and 5 from Blondel et al. (2020), we can analyze the asymptotic behavior as $\beta \to 0$. In this limit, the projection solutions converge to the subgradient of the OWA, represented as $\boldsymbol{w}_{(\sigma^{-1})}$ Blondel et al. (2020).

Figure 3.4 compares the effect of training OWA Decision Loss with these two gradient-based methods. The smoothness of the Moreau envelope, controlled by the hyperparameter β, is reflected in the performance shown by the four leftmost bars in the figure. Smaller values of β result in improved performance, with $\beta = 0.005$ yielding results comparable to the subgradient method. These findings align with the theoretical

asymptotic behavior. Therefore, for simplicity, we have adopted the subgradient model $\boldsymbol{g}$ in (3.7) as our preferred approach throughout the paper.

3.6. Conclusions

This work presents an efficient methodology for integrating Fair OWA optimization with predictive models. This proposal shows the potential of OWA optimization in data-driven decision-making, which has important applications in areas such as risk management and fair resource allocation.

CHAPTER 4

Integrating Machine Learning and Constrained Optimization: Fairness-Aware Learning-to-Rank

"There are no solutions. There are only trade-offs."

Thomas Sowell

This chapter extends the Predict-Then-Optimize (PtO) framework to a setting that combines a nonparametric OWA term with an additional parametric objective term. We demonstrate how backpropagation can be efficiently implemented using only a black-box solver for the underlying optimization problem, without the need for smoothing techniques. This framework is applied to a practical fairness-aware ranking task, integrating learning-to-rank models with fair ranking optimization programs in an end-to-end fashion. This approach optimizes web search results while ensuring fair exposure guarantees across user-defined content categories, based on the author's published work Dinh et al. (2024c).

4.1. Learning Fair Ranking Policies via Integration with Constrained Optimization

Ranking models have become a pervasive aspect of everyday life. They are the center of how people find information online, serving as the main mechanisms by which we interact with products, content, and others. In these systems, the items to be ranked are videos, job candidates, research papers, and almost anything else. As models based on machine learning, they are primarily trained to provide maximum utility to users, by serving the results deemed most relevant to their search queries. In the modern economy of information, the position of an item in the ranking has a strong influence on its exposure, selection, and, ultimately its economic success.

Because of this influence, increasing attention has been placed on the disparate impacts of ranking systems on underrepresented groups. In these data-driven systems, the relevance of an item is measured by implicit feedback from users such as clicks and dwell times. As such, the disparate impacts of rankings can go well beyond their immediate effects. Disproportionate exposure in rankings results leads to higher selection rates, in turn boosting relevance scores based on implicit feedback Yadav et al. (2019); Sun et al. (2020). This can create self-reinforcing feedback loops, leading to winner-take-all dynamics. The ability to control these disparate impacts is essential to avoid reinforcement of systemic biases,

ensure the health and stability of online markets, and implement anti-discrimination measures Edelman et al. (2017); Singh & Joachims (2019).

When search results are ranked purely based on relevance, disparate exposure between groups may be greatly increased in order to achieve marginal gains in relevance. For example, in a job search system it is possible for male candidates to receive overwhelmingly more exposure even when female candidates may have been rated only marginally lower in relevance. It has indeed been found in Elbassuoni et al. (2019) that in a job candidate ranking system, small differences in relevance can lead to large differences in exposure for candidates from a minority group. Thus, fairness-aware ranking models often suffer little to no degradation to user utility when compared to their conventional counterparts Zehlike & Castillo (2020).

On the other hand, these fair learning to rank models are more difficult to design, since they require the outputs of a machine learning model to obey potentially complex constraints while simultaneously achieving high relevance. For this reason, conventional learning to rank methods are often maladjusted to incorporate fairness. For example, the popular listwise learning to rank method, through modifications to its loss function, is only capable of modeling fairness of exposure in the top ranking position Zehlike & Castillo (2020). In an alternative paradigm, first investigated by Kotary et al. (2022), a fair ranking linear programming model is integrated with LTR in an end-to-end training process. By incorporating fair ranking optimization into the model's training loop, rather than in post-processing, utility of the downstream fair ranking policies can be maximized as a loss function. This allows to provide fairness guarantees at the level of each predicted ranking policy, and precise control over the fairness-utility trade-off. However, the method comes with a significant computational cost, as it requires solving a large optimization problem for each sample in each training iteration, challenging its application to real-world ranking systems. A further limitation of fair LTR systems, including that of Kotary et al. (2022), is their inability to effectively deal with multi-group fairness criteria (i.e., going beyond binary group treatment), which are overwhelmingly common in real-world applications.

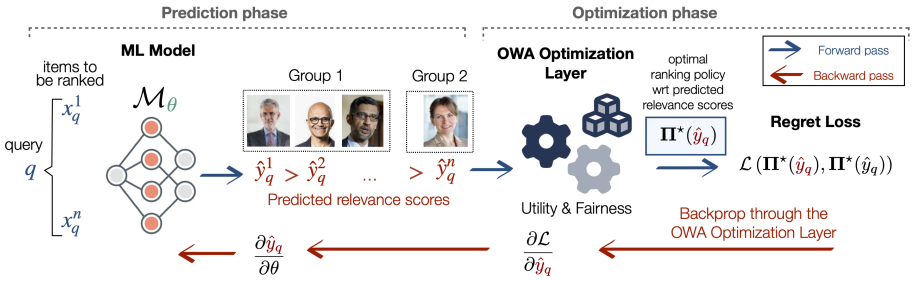


Figure 4.1: The differentiable optimization module proposed in SOFaiR. Its forward pass is calculated by an efficient Frank-Wolfe method, and its backward pass computes the SPO+ subgradient of the OWA problem's regret due to prediction error.

Contributions. To address these limitations, we make the following novel contributions: **(1)** It shows how to adopt an alternative approach based on Ordered Weighted Averages (OWA) to design efficient policy optimization modules for the fair learning-to-rank setting. **(2)** For the first time, it shows how to backpropagate gradients through the highly discontinuous optimization of OWA functions, enabling its use in end-to-end learning. **(3)** The resulting end-to-end optimization and learning scheme, called Smart OWA Optimization for Fair Learning to Rank (**SOFaiR**), is compared with contemporary fair LTR methods, demonstrating not only substantial advantages in fairness over previous fair LTR, but also advantages in efficiency and modeling flexibility over the end-to-end fair LTR scheme of Kotary et al. (2022). A schematic illustration of the proposed scheme is depicted in Figure 4.1.

These contributions are significant: They demonstrate that by incorporating modern fair ranking optimization techniques, the integration of post-processing optimization models in end-to-end LTR training can be a viable and scalable paradigm to achieve highly accurate learning to rank system that also provides strong fairness properties.

4.2. Preliminaries

Throughout the paper,vectors and matrices are denoted in bold font. The inner product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is written $\mathbf{a}^T\mathbf{b}$, while the outer product is $\mathbf{a}\,\mathbf{b}^T$. For a matrix $\mathbf{M}$, the vector $\overrightarrow{\mathbf{M}}$ is formed by concatenation of its rows. A hatted vector $\hat{\mathbf{a}}$ is the prediction of a machine learning model, and a starred vector $\mathbf{a}^\star$ is the optimal solution to some optimization problem. The list of integers $\{1 \dots n\}$ is written $[n]$. When $\mathbf{a} \in \mathbb{R}^n$ and σ is a permutation of $[n]$, $\mathbf{a}_\sigma$ is the corresponding permuted vector. The vectors of all ones and zeros are denoted $\mathbf{1}$ and $\mathbf{0}$, respectively. Commonly used symbols throughout the paper are organized in Table 4.1 for reference.

4.2.1 Problem Setting and Goals

Given a user query, the goal is to predict a ranking over n items, in order of most to least relevant, with respect to the query. Relevance of each item to be ranked, with respect to a search query q, is generally measured by a vector of *relevance scores* $\boldsymbol{y}_q \in \mathbb{R}^n$, often modeled on the basis of empirical observations such as historical click rates Xu et al. (2010). This setting considers a ground-truth dataset $(\boldsymbol{x}_q, \boldsymbol{a}_q, \boldsymbol{y}_q)_{q=1}^N$, where $\boldsymbol{x}_q \in \mathcal{X}$ is a list of feature vectors $(x_q^i)_{i=1}^n$, one for each of n items to be ranked in response to query q. $\boldsymbol{a}_q = (a_q^i)_{i=1}^n$ is a vector that indicates which (protected) group g within domain G to which each item belongs. $\boldsymbol{y}_q = (y_q^i)_{i=1}^n \in \mathcal{Y}$ is a vector of relevance scores, for each item with respect to query q. For example, on an image web-search context as depicted in Figure 4.1, a query denotes the search keywords, e.g., "CEO", the vectors x_q^i in $\boldsymbol{x}_q$ are feature embeddings for the images relative to q, each associated with a gender (attribute a_q^i), and the associated relevance scores y_q^i describe the relevance of item i to query q.

Rankings can be viewed as *permutations* which rearrange the order of a predefined *item list*. Intermediate between the user input and final ranking is often a ranking *policy* which produces discrete rankings (randomly or deterministically).

Learning to Rank. In learning to rank (LTR), a ML model $\mathcal{M}_\theta$ is often

Table 4.1: Common symbols adopted throughout the paper.

Symbol	**Semantic**
N	Size of the training dataset
n	Number of items to be ranked
m	Number of protected groups
$\boldsymbol{x}_q = (x_q^i)_{i=1}^n$	List of feature embeddings for items to rank, given 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 for each of n items given query q
G	The set of all protected group indicators
$\mathcal{M}_\theta$	End-to-end trainable fair ranking model with weights θ
σ	A permutation of the list $[n]$ for some n
$\mathbf{P}$	A permutation matrix corresponding to some σ
$\mathcal{P}_n$	The set of all permutations of $[n]$
τ	The sorting operator
Π	A ranking policy, or its representative bistochastic matrix
$\boldsymbol{u}(\Pi, y)$	Expected utility of policy Π under relevance scores y
$\mathcal{B}$	Birkhoff Polytope, the convex set of all ranking policies
$\mathcal{E}(i, \sigma)$	Exposure of item i

adopted to estimate relevance scores $\hat{\mathbf{y}}_q$ of items given their features $\mathbf{x}_q$ relative to user query q (see figure 4.1). From this a ranking *policy* $\mathbf{\Pi}$ is constructed. Its *expected utility* u is

$$u(\mathbf{\Pi}, \mathbf{y_q}) = \mathbb{E}_{\sigma \sim \mathbf{\Pi}}[\mathbf{\Delta}(\sigma, \mathbf{y_q})], \tag{4.1}$$

where $\mathbf{\Pi}$ is viewed as a distribution from which rankings σ are sampled randomly, and their utility Δ is a measure of the overall relevance of a given ranking σ, with respect to given relevance scores $\mathbf{y}_q$. Although its framework is applicable to any linear utility metric Δ for rankings, this

thesis uses the widely adopted Discounted Cumulative Gain (DCG):

$$\Delta(\sigma, \mathbf{y_q}) = DCG(\sigma, \mathbf{y_q}) = \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{y_q^i b_{\sigma_i}} = \mathbf{y_q^T P^{(\sigma)} b}, \tag{4.2}$$

where $\mathbf{P}^{(\sigma)}$ is the corresponding permutation matrix, $\boldsymbol{y}_q$ are the true relevance scores, and $\boldsymbol{b}$ is a *position bias* vector which models the probability that each position is viewed by a user, defined with elements $b_j = 1/\log_2(1+j)$, for $j \in [n]$.

Ranking policy representation. The methods of this thesis adopt a particular representation of the ranking policy, as bistochastic matrix $\mathbf{\Pi} \in \mathbb{R}^{n \times n}$, where $\mathbf{\Pi}_{jk}$ indicates the probability that item j takes position k in the ranking. The set of feasible ranking policies is expressed as $\mathbf{\Pi} \in \mathcal{B}$ where $\mathcal{B}$ is the *Birkhoff Polytope*:

$$\mathcal{B} = \{\mathbf{\Pi} \;\; s.t. \;\; \mathbf{1}^T\mathbf{\Pi} = \mathbf{1}, \;\; \mathbf{\Pi}\,\mathbf{1} = \mathbf{1}, \;\; \mathbf{0} \leq \mathbf{\Pi} \leq \mathbf{1}\}. \tag{4.3}$$

Its conditions on a matrix $\mathbf{\Pi}$ require, in the order of 4.3, that each column of $\mathbf{\Pi}$ sums to one, each row of $\mathbf{\Pi}$ sums to one, and each element of $\mathbf{\Pi}$ lies between 0 and 1. Each of these conditions is a linear constraint on the variables $\mathbf{\Pi}$.

Linearity of the DCG function (4.1) w.r.t. $\mathbf{P}$ allows it to commute with the expectation, leading to the practical closed form $u(\mathbf{\Pi}, \mathbf{y}) = \mathbf{y}^\top \mathbf{\Pi}\, \mathbf{b}$ for u as a linear function of $\mathbf{\Pi}$:

$$\begin{aligned} u(\mathbf{\Pi}, \mathbf{y}) &= \mathbb{E}_{\sigma \sim \mathbf{\Pi}} \Delta(\sigma, \boldsymbol{y}) \\ &= \mathbb{E}_{\sigma \sim \mathbf{\Pi}} \left[\boldsymbol{y}^\top \mathbf{P}^{(\sigma)} \boldsymbol{b} \right] = \boldsymbol{y}^\top \left(\mathbb{E}_{\sigma \sim \mathbf{\Pi}} \mathbf{P}^{(\sigma)} \right) \boldsymbol{b} = \boldsymbol{y}^\top \mathbf{\Pi} \boldsymbol{b} \end{aligned} \tag{4.4}$$

This is an important observation that enables the constrained optimization of utility functions on the policy $\mathbf{\Pi}$ in end-to-end differentiable pipelines, as discussed later in the thesis.

4.2.2 Fairness of Exposure

Item exposure is commonly adopted in ranking systems, where items in higher ranking positions receive more exposure, and it is with respect to this metric that fairness is concerned. This thesis aims at learning

ranking policies that satisfy **group fairness of exposure**, while maintaining high relevance to user queries. 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. Throughout the paper, the common modeling choice $\mathcal{E}(i, \sigma) = b_{\sigma_i}$.

Notions of item exposure in rankings can also be extended to group exposure in ranking policies. The exposure of group g in ranking σ is measured by the mean exposure in σ of items belonging to g. The exposure of group g in ranking policy $\mathbf{\Pi}$ is the mean value of its exposure over all rankings sampled from the policy:

$$\mathcal{E}_g(\mathbf{\Pi}) = \mathbb{E}_{\substack{\sigma \sim \mathbf{\Pi} \\ i \sim [n]}} \left[\mathcal{E}\left(i, \sigma\right) | a_q^i = g \right], \tag{4.5}$$

and we let $\mathcal{E}_G(\mathbf{\Pi})$ be the vector of values (4.5) for each g in G. Derived similarly to (4.4), linearity of $\mathcal{E}$ leads to a closed form for (4.5) when $\mathbf{\Pi}$ is represented by a bistochastic matrix, where $\mathbf{1}_g$ indicates 1 for items in g and 0 elsewhere Singh & Joachims (2018):

$$\mathcal{E}_g(\mathbf{\Pi}) = \frac{1}{|g|} \mathbf{1}_g^T \mathbf{\Pi}\, \boldsymbol{b}. \tag{4.6}$$

Imposing fairness in LTR. It is well-known that individual rankings σ, as discrete structures, cannot exactly satisfy most notions of individual or group fairness Zehlike et al. (2017). Therefore a common strategy in fair ranking optimization is to view ranking policies as random distributions of rankings, upon which a feasible notion of fairness can be imposed *in expectation* Zehlike et al. (2017); Singh & Joachims (2018); Do & Usunier (2022). For ranking policy $\mathbf{\Pi}$ and query q, fairness of exposure requires that every group indicated by $g \in G$ receives equal exposure on average over rankings produced by the policy. This condition can be expressed by requiring that the average exposure among items of each group is equal to the average exposure among all items:

$$\mathcal{E}_g(\mathbf{\Pi}) = \mathcal{E}_\alpha(\mathbf{\Pi}), \quad \forall g \in G, \tag{4.7}$$

where α is the group containing all items. Enforcing the condition (4.7) on each predicted policy $\mathbf{\Pi}$ is the mechanism by which protected groups are ensured equal exposure in SOFaiR. In the image search example, it corresponds to male and female candidates receiving equal exposure on

average over rankings sampled from $\mathbf{\Pi}$. The violation of fairness with respect to group g is measured by the absolute gap in this condition:

$$\nu_g(\mathbf{\Pi}) = \mid \mathcal{E}_g(\mathbf{\Pi}) - \mathcal{E}_\alpha(\mathbf{\Pi}) \mid . \tag{4.8}$$

Note that group fairness encompasses individual item fairness is a special case, where each item belongs to a distinct group. While the fairness and utility metrics described above are the ones used throughout the paper, the methodology of the paper is compatible with any alternative metrics u and $\mathcal{E}$ which are *linear* functions of the policy $\mathbf{\Pi}$. This is because the methodology of Sections 4.5 and 4.6 depend on linearity of (4.4) and (4.6).

4.3. Limitations of Fair LTR Methods

Current Fair LTR models present a combination of the following limitations: **(A)** inability to ensure fairness in each of its generated policies, **(B)** inability or ineffectiveness to handle multiple protected groups, and **(C)** inefficiency at training and inference time. This section reviews current fair LTR methods in light of these limiting factors.

Regardless of how the policy is represented, fair learning to rank methods typically train a model $\mathcal{M}_\theta$ to find parameters θ^* that maximize its empirical utility, along with possibly a weighted penalty term F which promotes fairness:

$$\theta^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ \frac{1}{N} \sum_{q=1}^{N} u(\mathcal{M}_{\boldsymbol{\theta}}(\boldsymbol{x}_q), \boldsymbol{y}_q) + \lambda \cdot F(\mathcal{M}_{\boldsymbol{\theta}}(\boldsymbol{x}_q)) \tag{4.9}$$

For example, the fair LTR method of Zehlike & Castillo (2020) (called DELTR) is based on listwise learning to rank Cao et al. (2007), and thus uses the model $\mathcal{M}_\theta$ to predict activation scores per each individual item, over which a softmax layer defines the probabilities of each item taking the top ranking position. Thus, Zehlike & Castillo (2020) can only use F to encourage group fairness of exposure in the top position, leading to poor overall satisfaction of the fairness condition (4.7) (**limitation A**) as illustrated in Figure 4.2. To impose fairness over all ranking positions, Singh & Joachims (2019) (FULTR) also uses softmax over the activations

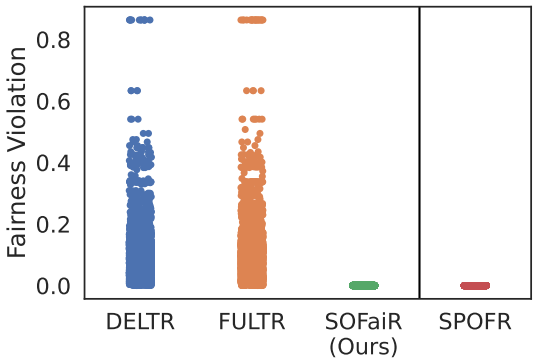


Figure 4.2: Yahoo-20: Fairness violation at query level.

of $\mathcal{M}_\theta$ to define probabilities, which are sampled without replacement to generate rankings using a policy gradient method. However, this penalty-based method still does not ensure fairness in each predicted policy, as illustrated in figure 4.2, since the penalty is imposed only *on average* over all predicted policies (**limitation A**). By a similar reasoning, these methods do not translate naturally to the case of multigroup fairness, where $m > 2$ (**limitation B**): Because the penalty F must scalarize the collection of all group fairness violations (4.8) (by taking their overall sum), it is possible to reduce F while increasing the exposure of a single outlier group Kotary et al. (2022).

Later work Kotary et al. (2022) shows how to overcome limitation A, by integrating the fair ranking optimization model of Singh & Joachims (2018) together with prediction of relevance scores $\hat{\mathbf{y}}_q = \mathcal{M}_\theta$. The modeling of predicted policies $\mathbf{\Pi}$ as solutions to an optimization problem under fairness constraints allows for their representation as bistochastic matrices which satisfies the fairness notions (4.7) exactly. However, this method suffers **limitation C** as it requires to solve a linear programming problem at each iteration of training and at inference, whose number of variables in $\mathbf{\Pi} \in \mathbb{R}^{n \times n}$ scales quadratically as $\mathcal{O}(n^2)$ becoming prohibitively large as the item list grows. Additionally, at inference time, the policy must be sampled to produce rankings; this requires a Birhoff-Von Neumann (BVM) decomposition of the matrix $\mathbf{\Pi}$ into a con-

vex combination of permutation matrices, which is also expensive when n is large Singh & Joachims (2018). Finally, in the case of multiple groups ($m > 2$), the fairness constraints can become infeasible, making this formulation unwieldly (**limitation B**).

Figure 4.2 shows the query-level fairness violations due to each method discussed in this section, where fairness parameters in each case are increased maximally without substantially compromising utility. In addition to higher average violations, penalty-based methods Zehlike & Castillo (2020); Singh & Joachims (2019) also lead to prevalence of outliers. These three existing fair LTR methods are used as baselines for comparison in Section 4.7. The SOFaiR framework proposed next most resembles Kotary et al. (2022), as it combines learning of relevance scores end-to-end with constrained optimization. At the same time, it aims to improve over Kotary et al. (2022) by addressing the three main limitations stated above. By integrating an alternative optimization component with its predictive model, SOFaiR can achieve faster runtime, and avoid the BVM decomposition at inference time, while naturally accommodating fairness over an arbitrary number of groups.

4.4. Smart OWA Optimization for Fair Learning to Rank (SOFaiR)

This section provides an overview of the proposed SOFaiR framework for learning fair ranking policies that overcomes limitations A, B, and C. Sections 4.5 and 4.6 will then detail the core solution approaches required to incorporate its proposed fair ranking optimization module into efficient, end-to-end trainable fair LTR models. As illustrated in figure 4.1, SOFaiR's core concept is to intergrate the learning of relevance scores with a module which optimizes fair ranking policies in-the-loop. By doing so it achieves a favorable balance of fairness and utility relative to other in-processing methods. The key difference in its approach relative to Kotary et al. (2022) is in the design of its optimization model which leverages Ordered Weighted Average (OWA) objectives (reviewed next) to enforce fairness of exposure. By avoiding the imposition of fairness of

exposure (see Equation 4.7) as a set of hard constraints on the optimization as in Kotary et al. (2022); Singh & Joachims (2018), it maintains the simple feasible region $\mathbf{\Pi} \in \mathcal{B}$, over which efficient Frank-Wolfe based solution methods can be employed to optimize its OWA objective function as described in Section 4.5. In turn, the particular form of the OWA optimization model in-the-loop necessitates a novel technique for its backpropagation, detailed in Section 4.6. The OWA aggregation and its fairness properties in optimization problems are introduced next, followed by its role in the SOFaiR learning framework.

4.4.1 Ordered Weighted Averaging Operator

The *Ordered Weighted Average* (OWA) operator Yager (1993) has found applications in various decision-making fields Yager & Kacprzyk (2012) as a means of fairly aggregating multiple objective criteria. Let $\mathbf{x} \in \mathbb{R}^m$ be a vector of m distinct criteria, and $\tau : \mathbb{R}^m \to \mathbb{R}^m$ be the sorting map for which $\tau(\mathbf{x}) \in \mathbb{R}^m$ holds the elements of $\mathbf{x}$ 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 weight $\mathbf{w}$ is defined as a linear functional on $\tau(\mathbf{x})$:

$$\mathrm{OWA}_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \tau(\mathbf{x}), \tag{4.10}$$

which is convex and piecewise-linear in $\mathbf{x}$ Ogryczak & Śliwiński (2003). The so-called Generalized Gini Functions, or Fair OWA, are those for which the OWA weights $w_1 > w_2 \ldots > w_n$ are decreasing. Fair OWA functions possess the following three key properties for fairness in optimizing multiple criteria Ogryczak & Śliwiński (2003). **(1)** *Impartiality* means that all criteria are treated equally, in the sense that $\mathrm{OWA}_{\mathbf{w}}(\mathbf{x}) = \mathrm{OWA}_{\mathbf{w}}(\mathbf{x}_\sigma)$ 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 increase in aggregated OWA value. That is , when $x_i > x_j + \epsilon$ and letting $\mathbf{x}_\epsilon = \mathbf{x}$ except at positions i and j where $(\mathbf{x}_\epsilon)_i = \mathbf{x}_i - \epsilon$ and $(\mathbf{x}_\epsilon)_j = \mathbf{x}_j + \epsilon$, it holds that $\mathrm{OWA}_{\mathbf{w}}(\mathbf{x}_\epsilon) > \mathrm{OWA}_{\mathbf{w}}(\mathbf{x})$. **(3)** *Monotonicity* means that $\mathrm{OWA}_{\mathbf{w}}(\mathbf{x})$ is an increasing function of each element of $\mathbf{x}$. The monotonicity property implies that solutions which optimize (4.10) are Pareto Efficient solutions of the underlying multi-objective problem, thus that no single criteria can be raised without

reducing another Ogryczak & Śliwiński (2003). Taken together, it is known that maximization of aggregation functions which satisfy these three properties produces so-called *equitably efficient solutions*, which possess the main intuitive properties needed for a solution to be deemed "fair"; see Kostreva & Ogryczak (1999) for a formal definition. As shown next, the SOFaiR framework ensures group fairness by leveraging a fair OWA aggregation of group exposures $\text{OWA}_{\mathbf{w}}(\mathcal{E}_G(\mathbf{\Pi}))$ in the objective function of its integrated fair ranking optimization module.

4.4.2 End-to-End Learning in SOFaiR

As illustrated in figure 4.1, the SOFaiR framework uses a prediction model $\mathcal{M}_\theta$ with learnable weights θ, which produces relevance scores $\hat{\boldsymbol{y}}_q$ from a list of item features $\mathbf{x}_q$. Its key component is an optimization module which maps the prediction $\hat{\boldsymbol{y}}_q$ to an associated ranking policy $\mathbf{\Pi}^\star(\hat{\boldsymbol{y}}_q)$. The following optimization problem defines $\mathbf{\Pi}^\star(\hat{\boldsymbol{y}}_q)$ as the ranking policy which optimizes a trade-off between fair OWA aggregation of group exposures with the expected DCG (as per Equation 4.4) under relevance scores $\hat{\boldsymbol{y}}_q$. In SOFaiR, it defines, for any chosen weight $0 \leq \lambda \leq 1$, a mapping which can be viewed akin to a neural network layer, representing the last layer of $\mathcal{M}$:

$$\mathbf{\Pi}^\star(\hat{\boldsymbol{y}}_q) = \text{argmax}_{\mathbf{\Pi}\in\mathcal{B}} \;\; (1-\lambda)\cdot u(\mathbf{\Pi}, \hat{\boldsymbol{y}}_q) + \lambda \cdot \text{OWA}_{\boldsymbol{w}}(\mathcal{E}_G(\mathbf{\Pi})), \quad (4.11)$$

wherein the Birkhoff Polytope $\mathcal{B}$ is the set of all bistochastic matrices, as defined in Equation 4.3. Let the objective function of Equation 4.11 be named $f(\mathbf{\Pi}, \hat{\boldsymbol{y}}_q)$. It is a convex combination of two terms measuring user utility and fairness, whose trade-off is controlled by a single coefficient $0 \leq \lambda \leq 1$. The former term measures expected user utility $u(\mathbf{\Pi}, \hat{\boldsymbol{y}}_q) = \hat{\boldsymbol{y}}_q^\top \mathbf{\Pi}\, \boldsymbol{b}$, while the latter term measures OWA aggregation of the group exposures. It is intuitive to see that when $\lambda = 1$, the optimization (4.11) returns a ranking policy that minimizes disparities in group exposure, without regard for relevance. When $\lambda = 0$, it returns a deterministic policy which ranks the items in order of the estimated scores $\hat{\boldsymbol{y}}_q$. Intermediate values $0 < \lambda < 1$ result in policies which trade off the effects of each term, balancing utility and fairness to various degrees. As λ increases, disparity between the exposure of protected groups must

decrease; this leads to a practical mechanism for achieving a desired level of fairness with minimal compromise to utility.

Since Equation (4.11) defines a direct mapping from $\hat{\boldsymbol{y}}_q$ to $\boldsymbol{\Pi}^\star(\hat{\boldsymbol{y}}_q)$, the problem of learning fair ranking policies reduces to a problem of learning relevance scores. This corresponds to estimating the objective function f via its missing coefficients $\boldsymbol{y}_q$. The SOFaiR training method defines a loss function between predicted and ground-truth relevance scores, as the loss of optimality in $\boldsymbol{\Pi}^\star(\hat{\boldsymbol{y}}_q)$ with respect to objective f under ground-truth $\boldsymbol{y}_q$, caused by prediction error in $\hat{\boldsymbol{y}}_q$. That is, the training objective is to minimize *regret* in f induced by $\hat{\boldsymbol{y}}_q$, defined as:

$$\textit{regret}(\hat{\boldsymbol{y}}_q, \boldsymbol{y}_q) = f(\ \Pi^\star(\boldsymbol{y}_q), \boldsymbol{y}_q\ ) - f(\ \Pi^\star(\hat{\boldsymbol{y}}_q), \boldsymbol{y}_q\ ). \tag{4.12}$$

The composition $\boldsymbol{\Pi}^\star \circ \mathcal{M}_\theta$ defines an integrated prediction and optimization model which maps item features to fair ranking policies. Training the integrated model by stochastic gradient descent follows these steps in a single iteration:

1. For sample query q and item features $\mathbf{x}_q$, a predictive model $\mathcal{M}_\theta$ produces estimated relevance scores $\hat{\mathbf{y}}_q$.
2. The predicted scores $\hat{\mathbf{y}}_q$ are used to populate the unknown parameters of an optimization problem (4.11). A solution algorithm is employed to find $\Pi^*(\hat{\mathbf{y}}_q)$, the optimal fair ranking policy relative to $\hat{\mathbf{y}}_q$.
3. The regret loss (4.12) is backpropagated through the calculations of steps (1) and (2), in order to update the model weights θ by a gradient descent step.

The following sections detail the main solution schemes for implementing steps (2) and (3). Section 4.5 shows how recently proposed fair ranking optimization techniques from Do & Usunier (2022) can be adapted to the setting of this thesis, in which fair ranking policies must be learned from empirical data. From this choice of optimization design arises a novel challenge in the backpropagation step (3), since no known work has shown how to backpropagate the regret of a highly discontinuous OWA optimization program. Section 4.6 shows how to efficiently backpropagate the regret due to Equation 4.11 for end-to-end learning. Then, Section 4.7 evaluates the SOFaiR framework against several other methods for learning fair ranking policies, on a set of benchmark tasks

from the web search domain.

4.5. Forward Pass Optimization

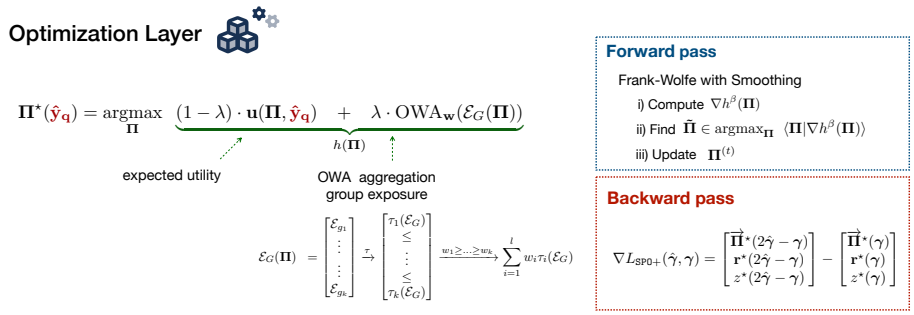


Figure 4.3: The differentiable optimization module employed in SOFaiR. It forward pass solves the problem (4.11) by an efficent Frank-Wolfe method. Its backward pass calculates the SPO+ subgradient, relative to its equivalent, but intractably large LP form.

The main motivation for the formulation (4.11) of SOFaiR's fair ranking optimization layer is to render the optimization problem efficiently solvable. Its main exploitable attribute is its feasible region $\mathbf{\Pi}$, over which a *linear* objective function can be quickly optimized by simply sorting a vector in $\mathbb{R}^n$, which has time complexity $n \log n$ Cormen et al. (2022). This suggests an efficient solution by Frank-Wolfe methods, which solve a constrained optimization problem by a sequence of subproblems optimizing a linear approximation of the true objective function Beck (2017). This efficient solution pattern is made possible by the absence of additional group fairness constraints on the policy variable $\mathbf{\Pi}$.

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

$$\mathbf{x}^{(k+1)} = (1 - \alpha^{(k)})\mathbf{x}^{(k)} + \alpha^{(k)} \operatorname*{argmax}_{\mathbf{y} \in \mathbb{S}} \langle \mathbf{y}, \nabla f(\mathbf{x}^{(k)}) \rangle. \tag{4.13}$$

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 (4.11) by the method (4.13) 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 (4.13), in such a way that the $f^{(k)}$ converge to the true f as $k \to \infty$.

It is proposed in Do & 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^{β_k} of f, a $\frac{1}{\beta_k}$-smooth approximation of f defined as Beck (2017):

$$h^{\beta}(\mathbf{x}) = \min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\beta}\|\mathbf{y} - \mathbf{x}\|^2. \tag{4.14}$$

When $f = \text{OWA}_{\boldsymbol{w}}$, let its Moreau envelope be denoted $\nabla\text{OWA}^{\beta}_{\boldsymbol{w}}$; it is shown in Do & Usunier (2022) that its gradient can be computed as a projection onto the permutahedron induced by modified OWA weights $\tilde{\boldsymbol{w}} = -(w_m, \ldots, w_1)$. By definition, the permutahedron $\mathcal{C}(\tilde{\boldsymbol{w}}) = \text{CONV}(\{\boldsymbol{w}_\sigma : \forall \sigma \in \mathcal{P}_m\})$ induced by a vector $\tilde{\boldsymbol{w}}$ is the convex hull of all its permutations. In turn, it is shown in Blondel et al. (2020) that the permutahedral projection $\nabla\text{OWA}^{\beta}_{\boldsymbol{w}}(\mathbf{x}) = \text{proj}_{\mathcal{C}(\tilde{\boldsymbol{w}})}(\boldsymbol{x}/\beta)$ 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 $\text{OWA}^{\beta}_{\boldsymbol{w}}$ with respect to optimization variables $\boldsymbol{\Pi}$, a convenient form can be derived from the chain rule:

$$\nabla_{\boldsymbol{\Pi}} \text{OWA}^{\beta}_{\boldsymbol{w}}(\mathcal{E}(\boldsymbol{\Pi})) = \boldsymbol{\mu}\boldsymbol{b}^T. \tag{4.15}$$

where $\boldsymbol{\mu} = \text{proj}_{\mathcal{C}(\tilde{\boldsymbol{w}})}(\mathcal{E}(\boldsymbol{\Pi})/\beta)$ and $\mathcal{E}(\boldsymbol{\Pi})$ is the vector of all item exposures Do & Usunier (2022). For the case where group exposures $\mathcal{E}_G(\boldsymbol{\Pi})$ are aggregated by OWA, first note that by Equation 4.6, $\mathcal{E}_G(\boldsymbol{\Pi}) = \boldsymbol{A}\boldsymbol{\Pi}\boldsymbol{b}$, where $\boldsymbol{A}$ is the matrix composed of stacking together all group indicator vectors $\mathbf{1}_g \ \forall g \in G$. Since $\mathcal{E}(\boldsymbol{\Pi}) = \boldsymbol{\Pi}\boldsymbol{b}$, this implies $\mathcal{E}_G(\boldsymbol{\Pi}) = \mathcal{E}(\boldsymbol{A}\boldsymbol{\Pi})$, thus

$$\nabla_{\boldsymbol{\Pi}} \text{OWA}^{\beta}_{\boldsymbol{w}}(\mathcal{E}_G(\boldsymbol{\Pi})) = (\boldsymbol{A}^T\tilde{\boldsymbol{\mu}})\ \boldsymbol{b}^T. \tag{4.16}$$

by the chain rule, and where $\tilde{\boldsymbol{\mu}} = \text{proj}_{\mathcal{C}(\tilde{\boldsymbol{w}})}(\mathcal{E}_G(\boldsymbol{A}\boldsymbol{\Pi})/\beta)$. It remains now to compute the gradient of the user relevance term $u(\boldsymbol{\Pi}, \hat{\boldsymbol{y}}_q) = \hat{\boldsymbol{y}}_q^T\boldsymbol{\Pi}\ \boldsymbol{b}$

Algorithm 1: *Frank-Wolfe with Moreau Envelope Smoothing to solve* (4.11)

Input: predicted relevance scores $\hat{\boldsymbol{y}} \in \mathbb{R}^n$, group mask $\boldsymbol{A}$, max iteration T, smooth seq. (β_k)

Output: ranking policy $\boldsymbol{\Pi}^{(T)} \in \mathbb{R}^{n \times n}$

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

in Equation 4.11. As a linear function of the matrix variable $\boldsymbol{\Pi}$, its gradient is $\nabla_{\boldsymbol{\Pi}}\, u(\boldsymbol{\Pi}, \hat{\boldsymbol{y}}_q) = \hat{\boldsymbol{y}}_q\, \boldsymbol{b}^T$, which is evident by comparing to the equivalent vectorized form $\hat{\boldsymbol{y}}_q^T \boldsymbol{\Pi}\, \boldsymbol{b} = \overrightarrow{\hat{\boldsymbol{y}}_q\, \boldsymbol{b}^T} \cdot \overrightarrow{\boldsymbol{\Pi}}$. Combining this with Equation 4.16, the total gradient of the objective function in Equation 4.11 with smoothed OWA term is $(1-\lambda) \cdot \hat{\boldsymbol{y}}_q\, \boldsymbol{b}^T + \lambda \cdot (\boldsymbol{A}^T \tilde{\boldsymbol{\mu}})\, \boldsymbol{b}^T$, which is equal to $\left((1-\lambda) \cdot \hat{\boldsymbol{y}}_q + \lambda \cdot (\boldsymbol{A}^T \tilde{\boldsymbol{\mu}})\right)\, \boldsymbol{b}^T$. Therefore the SOFaiR module's Frank-Wolfe linearized subproblem is

$$\underset{\boldsymbol{\Pi} \in \mathcal{B}}{\text{argmax}} \left\langle \boldsymbol{\Pi}, \left((1-\lambda) \cdot \hat{\boldsymbol{y}}_q + \lambda \cdot (\boldsymbol{A}^T \tilde{\boldsymbol{\mu}})\right)\, \boldsymbol{b}^T \right\rangle \tag{4.17}$$

To implement the Frank-Wolfe iteration (4.13), 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 $\boldsymbol{b}$ can be exploited. Note that as the expected DCG under relevance scores $\boldsymbol{y}$, the function $\boldsymbol{y}^T \Pi\, \boldsymbol{b}$ is maximized by the permutation matrix $\boldsymbol{P} \in \mathcal{P}_n$ which sorts the relevance scores $\boldsymbol{y}$ decreasingly. But since $\boldsymbol{y}^T \boldsymbol{\Pi}\, \boldsymbol{b} = \overrightarrow{\boldsymbol{y}\, \boldsymbol{b}^T} \cdot \overrightarrow{\boldsymbol{\Pi}}$, we identify $\boldsymbol{y}^T \boldsymbol{\Pi}\, \boldsymbol{b}$ as the linear function of $\overrightarrow{\boldsymbol{\Pi}}$ with gradient $\overrightarrow{\boldsymbol{y}\, \boldsymbol{b}^T}$. Therefore equation Equation 4.17 can be solved in $\mathcal{O}(n \log n)$, simply by finding $\boldsymbol{P} \in \mathcal{P}_n$ as the argsort of the vector

$((1-\lambda)\cdot\hat{\boldsymbol{y}}_q+\lambda\cdot(\boldsymbol{A}^T\tilde{\boldsymbol{\mu}}))$ in decreasing order. A more formal proof, cited in Do et al. (2021), makes use of Hardy et al. (1952).

The overall method is presented in Algorithm 1. Decay of the smoothing parameter $\beta_t=\frac{\beta_0}{\sqrt{t}}$ satisfies the conditions for convergence stated in Lan (2013) when β_0 is sufficiently large. Sparse matrix additions each require $\mathcal{O}(n)$ operations, so that Algorithm 1 maintains $\mathcal{O}(n\log n)$ complexity per iteration. An important advantage of Algorithm 1 over the fair ranking LP employed in SPOFR Kotary et al. (2022), is that the solution iterates $\boldsymbol{P}^{(k)}$ automatically provide a decomposition of the policy matrix $\boldsymbol{\Pi}=\rho_k\boldsymbol{P}^{(k)}$ as a convex combination of rankings, by which it can be readily sampled as a discrete probability distribution. In contrast, the LP module used in SPOFR Kotary et al. (2022) provides as its solution only a matrix $\boldsymbol{\Pi}\in\mathcal{B}$, which must be decomposed using the Birkhoff Von Neumann decomposition, adding substantially to its total runtime.

4.6. Backpropagation

The formulation of the optimization module (4.11) allows for efficient solution via Algorithm 1, but gives rise to a novel challenge in backpropagating the regret loss function through $\Pi^\star(\hat{\boldsymbol{y}}_q)$. By including an OWA aggregation of group exposure, its objective function is nonlinear and nondifferentiable. This section shows how to train the integrated prediction and OWA optimization model $\boldsymbol{\Pi}^\star\circ\mathcal{M}_\theta$ to minimize the regret loss (4.12), despite this challenge. As a starting point, we recognize the existing literature on "Predict-Then-Optimize" frameworks Kotary et al. (2021); Mandi et al. (2024a) for minimizing the regret due to prediction error in the objective coefficients of a linear program, denoted $\boldsymbol{c}$ below:

$$\mathbf{x}^\star(\mathbf{c})=\underset{\mathbf{Ax}\leq\mathbf{b}}{\operatorname{argmin}}\,\mathbf{c}^T\mathbf{x}. \tag{4.18}$$

Several known methods have been proposed Elmachtoub & Grigas (2021); Pogančić et al. (2020); Wilder et al. (2019); Berthet et al. (2020) and well-established in the literature Mandi et al. (2024a) for end-to-end training of combined prediction and optimization models employing (4.18). Due to its OWA objective term, the fair ranking module (4.11) does not sat-

isfy the LP form (4.18) for which the aforementioned methods are taylored. The implementation of SOFaiR described here uses the "Smart Predict-Then-Optimize" (SPO) approach Elmachtoub & Grigas (2021), since its simple backpropagation rule requires only a solution to (4.18) using a blackbox solution oracle. This allows its adaptation to the OWA optimization setting by constructing (but not solving) an equivalent but intractable linear programming form to (4.11), as shown next.

End-to-End learning with SPO+ Loss. Viewed as a loss function, the regret (4.12) in solutions to problem (4.18) is nondifferentiable and discontinuous with respect to predicted coefficients $\hat{\boldsymbol{c}}$, since solutions $\boldsymbol{x}^\star(\boldsymbol{c})$ must occur at one of finitely many vertices in $\mathbf{Ax} \leq \mathbf{b}$. The SPO+ loss function proposed in Elmachtoub & Grigas (2021) is by construction a Fischer-consistent, *subdifferentiable* upper bound on regret. In particular, it is shown in Elmachtoub & Grigas (2021) that

$$L_{\texttt{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \max_{\mathbf{x}}(\mathbf{c}^T\mathbf{x} - 2\hat{\mathbf{c}}^T\mathbf{x}) + 2\hat{\mathbf{c}}^T\mathbf{x}^\star(\mathbf{c}) - \mathbf{c}^T\mathbf{x}^\star(\mathbf{c}), \tag{4.19}$$

possesses these properties, and a subgradient at $\hat{\mathbf{c}}$ is

$$\nabla L_{\texttt{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \boldsymbol{x}^\star(2\hat{\boldsymbol{c}} - \boldsymbol{c}) - \boldsymbol{x}^\star(\boldsymbol{c}). \tag{4.20}$$

Minimizing the surrogate loss (4.19) by gradient descent using (4.20) is key to minimizing the solution regret in problem (4.18) due to error in a predictive model which predicts the parameter $\mathbf{c}$.

SPO+ loss in SOFaiR. We now show how the SPO training framework described above for the problem type (4.18) can be used to efficiently learn optimal fair policies in conjunction with problem (4.11). The main idea is to derive an SPO+ subgradient for regret in (4.11), through an equivalent linear program (4.18), but without solving it as such. This is made possible by the fact that the subgradient (4.20) can be expressed as a difference of two optimal solutions, which can be furnished by any optimization oracle which solves the mapping (4.11), which includes Algorithm (1).

First note, as it is shown in Ogryczak & Śliwiński (2003), that the OWA function (4.10) can be expressed as

$$OWA_{\mathbf{w}}(\mathbf{r}) = \min_{\sigma \in \mathcal{P}} \ \mathbf{w}_\sigma \cdot \mathbf{r}, \tag{4.21}$$

and as an equivalent linear programming problem which views the min-

imum inner product above as the maximum lower bound among all possible inner products with the permuted OWA weights:

$$OWA_{\mathbf{w}}(\mathbf{r}) = \max_z \quad z \tag{4.22a}$$

$$s.t. \quad z \leq \mathbf{w}_\sigma \cdot \mathbf{r}, \qquad \forall \sigma \in \mathcal{P}, \tag{4.22b}$$

where $\mathcal{P}$ contains all possible permutations of $[n]$ when $\mathbf{w} \in \mathbb{R}^n$. This allows SOFaiR's OWA optimization model (4.11) to be recast in a linear programming form using auxiliary optimization variables $\mathbf{r}$ and z:

$$(\boldsymbol{\Pi}^\star, \boldsymbol{r}^\star, z^\star)(\hat{\boldsymbol{y}}_q) = \operatorname{argmax}_{\boldsymbol{\Pi} \in \mathcal{B},\, \mathbf{r},\, z} \quad (1-\lambda) \cdot \hat{\boldsymbol{y}}_q^\top \boldsymbol{\Pi}\, \boldsymbol{b} + \lambda \cdot z \tag{4.23a}$$

$$\text{subject to:} \quad z \leq \boldsymbol{w}_\sigma \cdot r, \qquad \forall \sigma \in \mathcal{P} \tag{4.23b}$$

$$r = \mathcal{E}_G(\boldsymbol{\Pi}). \tag{4.23c}$$

According to Ogryczak & Śliwiński (2003), this alternative LP form of OWA optimization is mostly of theoretical significance, since the set of constraints (4.23b) grows factorially in the size of $\mathbf{r}$, one for each possible permutation thereof. This makes (4.23) impractical for computing a solution to the original OWA problem (4.11), which we instead solve by Algorithm 1. On the other hand, we show that problem (4.23) *is* practical for deriving a *backpropagation* rule through the OWA problem (4.11).

Since the unknown parameters $\hat{\boldsymbol{y}}_q$ appear only in its linear objective function, this parametric LP problem (4.23) fits the form (4.18) required for training with SPO+ subgradients. To derive the subgradient explicitly, rewrite the linear objective term $\hat{\boldsymbol{y}}_q^\top \boldsymbol{\Pi}\, \boldsymbol{b} = \overrightarrow{\hat{\boldsymbol{y}}_q\, \boldsymbol{b}^T} \cdot \overrightarrow{\boldsymbol{\Pi}}$. Then in terms of the augmented variables $(\boldsymbol{\Pi}, \mathbf{r}, z)$, the objective function (4.23a) is

$$(1-\lambda)_x \cdot \hat{\boldsymbol{y}}_q^\top \boldsymbol{\Pi}\, \boldsymbol{b} + \lambda \cdot z = \underbrace{\begin{bmatrix} (1-\lambda)\overrightarrow{\hat{\boldsymbol{y}}_q\, \boldsymbol{b}^T} \\ \mathbf{0} \\ \lambda \end{bmatrix}^T}_{\hat{\boldsymbol{\gamma}}} \begin{bmatrix} \overrightarrow{\boldsymbol{\Pi}} \\ \mathbf{r} \\ z \end{bmatrix}. \tag{4.24}$$

Now the SPO+ loss subgradient can be readily expressed with respect to the augmented scores $\hat{\boldsymbol{\gamma}}$ defined as above:

$$\nabla L_{\texttt{SPO+}}(\hat{\boldsymbol{\gamma}}, \boldsymbol{\gamma}) = \begin{bmatrix} \overrightarrow{\boldsymbol{\Pi}}^{\star}(2\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}) \\ \mathbf{r}^{\star}(2\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}) \\ z^{\star}(2\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma}) \end{bmatrix} - \begin{bmatrix} \overrightarrow{\boldsymbol{\Pi}}^{\star}(\boldsymbol{\gamma}) \\ \mathbf{r}^{\star}(\boldsymbol{\gamma}) \\ z^{\star}(\boldsymbol{\gamma}) \end{bmatrix}, \tag{4.25}$$

using (4.20), and where $\boldsymbol{\gamma}$ is the augmented score based on ground-truth $\mathbf{y}_q$. Finally, backpropagation from $\hat{\boldsymbol{\gamma}}$ to the base prediction $\hat{\mathbf{y}}_q = \mathcal{M}_{\boldsymbol{\theta}}(\mathbf{x}_q)$ is performed by automatic differentiation, and likewise from $\hat{\mathbf{y}}_q$ to the model weights $\boldsymbol{\theta}$.

Both terms in (4.25) can be produced by using Algorithm 1 to solve (4.11) for $\boldsymbol{\Pi}^{\star}$. Then, the remaining variables $\mathbf{r}^{\star}$ and $z^{\star}$ are easily completed as groups exposures $\mathbf{r} = \mathcal{E}_G(\Pi^{\star})$ and their associated OWA value z, respectively. Importantly, the rightmost term of (4.25) is independent of any prediction; therefore it is *precomputed* in advance of training. Thus, backpropagation using (4.25) consists of computing the difference between two solutions, one of which comes from the forward pass and while the other is precomputed before training. The complexity of this backward pass consists of $\mathcal{O}(n^2)$ subtractions, which grows only linearly in the size of the matrix variable $\boldsymbol{\Pi} \in \mathbb{R}^{n \times n}$. The differentiable fair ranking optimization module of SOFaiR, with its forward and backward passes, is summarized in figure 4.3.

4.7. Experiments

Next we evaluate SOFaiR against two prior in-processing methods Singh & Joachims (2019); Zehlike et al. (2017), and the end-to-end framework Kotary et al. (2021), denoted as FULTR, DELTR, and SPOFR, respectively. We assess the performance on two datasets:

- Microsoft Learn to Rank (MSLR) is a standard benchmark for LTR with queries from Bing and manually-judged relevance labels. It includes 30,000 queries, each with an average of 125 assessed documents and 136 ranking feature. Binary protected groups is defined using the 50th percentile of QualityScore attribute. For multi-group cases, group labels are defined using evenly-spaced quantiles.
- Yahoo! Learning to Rank Challenge (Yahoo LETOR) contains 19,944

queries and 473,134 documents with 519 ranking features. Binary protected groups is defined using feature id 9 following Jia & Wang (2021) and the 50th percentile as the threshold.

For MSLR, we randomly sample 10,000 queries for training and 1,000 queries each for validation and testing. We create datasets with varying list sizes (20, 40, 60, 80, 100 documents) for MSLR and (20, 40 documents) for Yahoo LETOR.

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 λ in (4.9) and (4.11). Larger values of λ indicate more preference towards fairness. In SPOFR, the allowed violation (4.8) of group fairness is specified directly. Ranking utility and fairness violation are assessed using average DCG (Equation 4.1) and fairness violation (Equation 4.8), respectively. The metrics are computed as averages over the entire test dataset.

4.7.1 Running Time Analysis

Our analysis begins with a runtime comparison between SOFaiR and other LTR frameworks, to show how it overcomes **limitation C**, described in section 4.3. Figure 4.4 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 less than those required for solution of (4.11) at inference. Thus the reported results use 100 iterations in training and 500 at inference. Importantly, reported runtimes under-estimate the 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, as penalty-based methods, are more competitive in runtime. However, this comes at a cost of the achieved fairness level (**limitation A**), as shown in the next section.

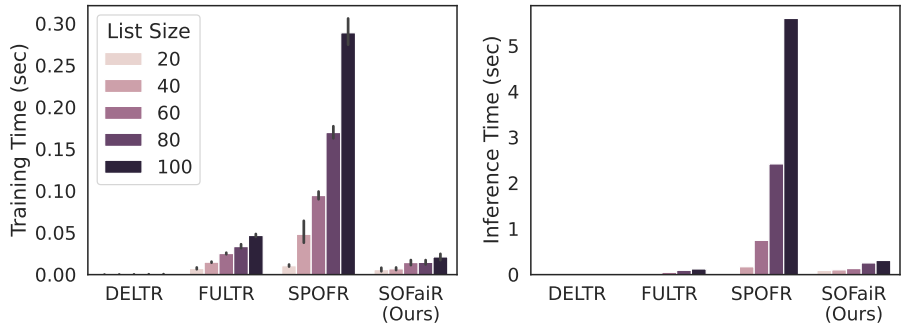


Figure 4.4: Running time benchmark on MSLR-Web10k dataset

4.7.2 Fairness and Utility Tradeoffs Analysis

Next, we focus on comparing the utility and fairness of the various LTR frameworks analyzed. This section focuses on the two-group case, as none of the methods compared against was able to cope with multi-group case in our experiments (see next section). Figure (4.5) presents the trade-off between utility and fairness across the test sets for both Yahoo LETOR and MSLR datasets, encompassing their lowest and highest list sizes. For each method, the intensity of colors represents the magnitude of its fairness parameter. A progression from lighter to darker colors indicates an increase in the importance placed on fairness. Consequently, darker colors are expected to correspond with more restrictive

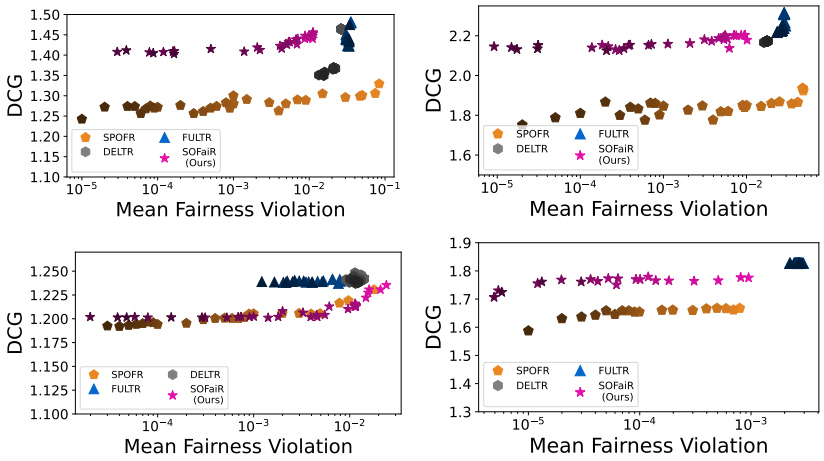


Figure 4.5: Benchmarking performance in term of fairnesss-utility trade-off on Yahoo-20 (top left), and Yahoo-40(top right). MSLR-20(bottom-left), MSLR-100 (bottom-right)

models, characterized by lower DCG scores (y-axis) but also fewer fairness violations (x-axis). Each point in the figure represents the largest DCG score obtained from a fairness hyperparameter search, as detailed in Appendix C. Note that points on the grid that are higher on the y-axis and lower on the x-axis represent superior results.

Firstly, notice that most points associated with methods DELTR and FULTR are clustered in a small region with both high DCG and (log-scaled) fairness violations. While these methods reach an order of magnitude reduction in fairness violation on some datasets, the effect is inconsistent, especially as the item list size increases (**limitation A**). In contrast, the end-to-end methods (SPOFR and the proposed SOFaiR) reach much lower fairness violations, underlining their effectiveness of their optimization modules in enforcing the fairness constraint.

Both DELTR and FULTR reach competitive utilities, but they consistently display relatively high fairness violations, underscoring their limitations in providing a fair ranking solution. SOFaiR shows competitive fairness and utility performance compared to SPOFR, with a marked

advantage in utility on some datasets. SPOFR ensures fairness but at the expense of efficiency, whereas SOFaiR reaches similar fairness levels at a fraction of the required runtime. Additional results on datasets of various list sizes are included in Appendix B.2.

4.7.3 Multi-Group Fairness Analysis

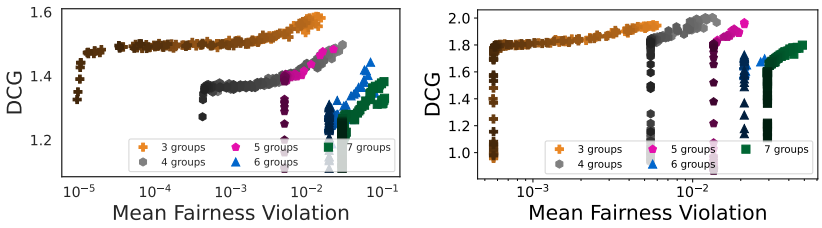


Figure 4.6: Fairness-utility tradeoff due to SOFaiR with multiple groups on MSLR-40 (left) and MSLR-100 (right) list size

Finally, this section analyzes the fairness-utility trade-off in multi-group scenarios using the SOFaiR framework. The SPOFR method returns infeasible solutions for most chosen fairness levels when multiple groups are introduced, preventing its evaluation on these datasets; this is naturally avoided in SOFaiR as the optimization of OWA aggregation without constraints simply increases fairness to the extent feasible. While FULTR provides no code to evaluate multigroup fairness, its penalty function is in principle ill-equipped to handle multiple groups as it must scalarize all group fairness violations into a single loss function as mentioned in Section 4.3 (**limitation C**). Figure 4.6 compares the average test DCG against the average fairness violation across various numbers of groups (ranging from 3 to 7) in the MSLR dataset, for list sizes of 40 and 100. Additional results for other list sizes in the MSLR dataset are available in Appendix B.1.

Each data point represents a single model's performance, with fairness parameters λ adjusted between 0 and 1. Models prioritizing fairness show reduced fairness violations and lower utilities, indicated by darker colored points, compared to those with a lower emphasis on fairness, represented by lighter colored points. A distinct trend is observed: as fair-

ness parameters are relaxed, utility increases for all metrics and datasets. It is also evident that multi-group fairness comes at a higher cost to utility. Predictably, saturation occurs in each curve, indicating that beyond a certain point, increasing the fairness weight does not further decrease fairness violations but merely reduces utility.

4.8. Conclusions

this thesis presented SOFaiR, a method that employs an Ordered Weighted Average optimization model to integrate fairness considerations into ranking processes. Its key contribution is to enable backpropagation through optimization of discontinuous OWA functions, which has makes it possible to precisely enforce flexible group fairness measures directly into the training process of learning to rank, without greatly compromising efficiency.

These advantages show that by leveraging modern developments in fair ranking optimization, the integration of constrained optimization and machine learning techniques can be a promising direction for future research in fair LTR.

CHAPTER 5

Learning to Optimize with Application in AC-OPF Problem

"Learn the rules like a pro, so you can break them like an artist."

Pablo Picasso

This chapter presents the original contributions of this thesis in the scope of Learning to Optimize for Optimal Power Flow (OPF) and is divided into two main sections. The first section investigates the predictability and robustness of deep learning models for approximating AC-OPF solutions, analyzing the underlying factors that influence prediction accuracy. This study builds upon the author's published work Dinh et al. (2023). The second section introduces a novel autoregressive deep learning framework that leverages iterative nonlinear solvers during training to enhance prediction robustness and scalability.

5.1. Deep Learning and Optimal Power Flow Problem

The Optimal Power Flow (OPF) problem finds the generator dispatch of minimal cost that meets the demands of a power system. The problem is required to satisfy the AC power flow equations, which are non-convex and nonlinear, and is a core building block in many power system applications. While its resolution has benefited from decades of research in power systems and operational research, the introduction of intermittent renewable energy sources is forcing system operators to adjust the generators set-points with increasing frequency. However, the resolution frequency to solve OPFs is limited by their computational complexity. To address this issue, system operators typically solve OPF approximations, such as the linear DC model, but, while more efficient computationally, their solutions may be sub-optimal and induce substantial economic losses.

Recently, an interesting line of research has focused on how to approximate AC-OPF using Deep Neural Networks (DNNs) Deka & Misra (2019); Zamzam & Baker (2020); Fioretto et al. (2020b). Once a DNN is trained, predictions can be computed on the order of milliseconds. While the recent results show that these learning models can approximate the generator set-points of AC-OPF with high accuracy, little is known on why these models can predict OPF solutions accurately, as well as about their robust predictions. This chapter provides a step forward to address

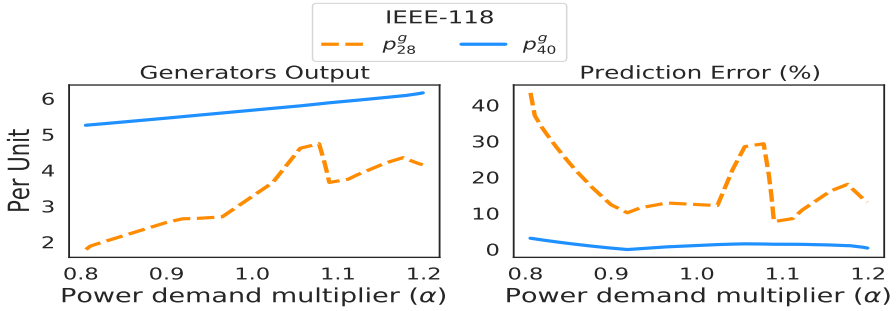


Figure 5.1: Generator output as a function of demand (right) and associated predictions (left). Orange (blue) colors show high (low) volatile curves while continuous (dashed) lines depict easy (hard) prediction tasks.

this knowledge gap and makes four main contributions.

We first ask: *Why are DNNs able to approximate OPF solutions with low errors?* To answer this question, the paper studies the relationship between the training data and their target outputs. Figure 5.1 (left) shows how generator outputs change as a function of the total demand for selected IEEE-118 generators. Notice that the blue curve suggests a linear dependence between the associated generator outputs and the loads, indicating that a simple learning model may effectively capture such behavior, as indeed confirmed in the corresponding low DNN prediction errors reported in Figure 5.1 (right). *The paper shows that when many generators exhibit this behavior, approximating OPF with DNNs produces accurate results, on average.*

There are, however, also generators whose outputs are inherently more difficult to predict. The orange curve in the figure depicts a much different scenario with a more volatile underlying function. The right plot shows the high prediction error attained, indicating robustness issues. *The paper sheds light on why these behaviors are not easily captured by standard learning models connecting the stability of the training data to the ability of a learning model to approximate it.*

Next, we ask: *What are the latent factors that affect the prediction accuracy of these generators?* To address this question, this chapter studies which characteristics of the OPF may be responsible for these

erroneous predictions, and indicates the need for modeling and predicting the behavior of the OPF engineering and physical constraints during training to capture the complexity of the predictions.

Finally, in light of the robustness issues observed in this study, we propose a new framework that relies on a deep autoregressive Recurrent Neural Network to exploit the data generated by iterative nonlinear solvers during training. The results show that this framework is not only able to improve the prediction robustness over existing DNN OPF predictors, but also it comes with a reduced memory footprint, thus, enabling it to predict very large instances, overcoming one of the limitations of existing DNN OPF predictors relying on fully-connected networks.

5.2. Related Work

The use of machine learning to accelerate the resolution of power system optimization procedures has recently seen a growing number of results. A recent survey by Hasan et al. Hasan et al. (2020) summarizes the development in the area.

In particular, Pan et al. Pan et al. (2019) explore DNN architectures for predicting DC-OPFs, a linear approximation of the full AC model. Deka et al. Deka & Misra (2019) and Ng et al. Ng et al. (2018) use a DNN architecture to learn the set of active constraints. By exploiting the linearity of the DC-OPF problem, once the set of relevant active constraints is identified, an exhaustive search can be used to find a solution that satisfies the active constraints. A deep learning approach for AC-OPFs is also proposed by Yang et al. Yang et al. (2020) to predict voltages and flows. This approach focuses on specific operational constraints while dismissing other physical and engineering constraints.

Other recent approaches have attempted to incorporate structure from OPF constraints into deep learning–based models. For instance, Fioretto et al. Fioretto et al. (2020b) propose a learning method which combines deep learning and Lagrangian duality, incorporating information about OPF dual variables into the learning loss function to promote the prediction of feasible solutions. Other approaches focus on enforc-

ing OPF constraints directly within the learning process. For instance, Zamzam and Baker Zamzam & Baker (2020) use a DNN to predict a partial OPF solution, and then solve for the remaining outputs using power the flow equations. Donti et al. Donti et al. (2020) extended this approach though the use of implicit layers which allows a DNN to reason about the hard constraints.

While these proposals have clearly shown that it is possible to approximate OPF solutions of high quality, and in vastly reduced computational times when compared to those required by traditional optimization solvers, a complete understanding of the reasons for the effectiveness of these learning models and their reliability is missing. The rest of the paper provides a first step toward addressing this knowledge gap.

5.3. Preliminaries

Optimal Power Flow. *Optimal Power Flow (OPF)* is the problem of determining the least-cost generator dispatch that meets the demands in a power network. A power network is viewed as a graph (N, E) where the set of nodes n describes n buses and the edges E describe e transmission lines. Here E is a set of directed arcs and E^R is used to denote the arcs in E but in reverse direction.

The AC power flow equations are based on complex quantities for current I, voltage V, admittance Y, and power S. The quantities are linked by constraints expressing Kirchhoff's Current Law (KCL), i.e., $I_i^g - \boldsymbol{I}_i^d = \sum_{(i,j)\in E\cup E^R} I_{ij}$, Ohm's Law, i.e., $I_{ij} = \boldsymbol{Y}_{ij}(V_i - V_j)$,, and the definition of AC power, i.e., $S_{ij} = V_i I_{ij}^*$. Combining these three properties yields the AC Power Flow equations, i.e.,

$$S_i^g - \boldsymbol{S}_i^d = \sum_{(i,j)\in E\cup E^R} S_{ij} \;\; \forall i \in N$$

$$S_{ij} = \boldsymbol{Y}_{ij}^* |V_i|^2 - \boldsymbol{Y}_{ij}^* V_i V_j^* \;\; (i,j) \in E \cup E^R$$

These non-convex nonlinear equations are the core building blocks in many power system applications. Practical applications typically include various operational constraints on the flow of power, which are captured in the AC OPF formulation in Model 1. The objective function (5.1)

Model 1 The AC Optimal Power Flow Problem (AC-OPF)

$$\textbf{variables:}\quad S_i^g, V_i \;\; \forall i \in N, \;\; S_{ij} \;\; \forall (i,j) \in E \cup E^R$$

$$\textbf{minimize:}\quad \sum_{i \in N} \boldsymbol{c}_{2i}(\Re(S_i^g))^2 + \boldsymbol{c}_{1i}\Re(S_i^g) + \boldsymbol{c}_{0i} \tag{5.1}$$

$$\textbf{subject to:}\quad \boldsymbol{v}_i^{\boldsymbol{l}} \leq |V_i| \leq \boldsymbol{v}_i^{\boldsymbol{u}} \;\; \forall i \in N \tag{5.2}$$

$$-\boldsymbol{\theta}_{ij}^{\boldsymbol{\Delta}} \leq \angle(V_i V_j^*) \leq \boldsymbol{\theta}_{ij}^{\boldsymbol{\Delta}} \;\; \forall (i,j) \in E \tag{5.3}$$

$$\boldsymbol{S}_i^{\boldsymbol{gl}} \leq S_i^g \leq \boldsymbol{S}_i^{\boldsymbol{gu}} \;\; \forall i \in N \tag{5.4}$$

$$|S_{ij}| \leq \boldsymbol{s}_{ij}^{\boldsymbol{u}} \;\; \forall (i,j) \in E \cup E^R \tag{5.5}$$

$$S_i^g - \boldsymbol{S}_i^d = \textstyle\sum_{(i,j) \in E \cup E^R} S_{ij} \;\; \forall i \in N \tag{5.6}$$

$$S_{ij} = \boldsymbol{Y}_{ij}^* |V_i|^2 - \boldsymbol{Y}_{ij}^* V_i V_j^* \;\; \forall (i,j) \in E \cup E^R \tag{5.7}$$

captures the cost of the generator dispatch. Constraints (5.2) and (5.3) capture the voltage and phase angle difference operational constraints. Constraints (5.4) and (5.5) enforce the generator output and line flow limits. Finally, constraints (5.6) capture KCL and constraints (5.7) capture Ohm's Law. Notice that this is a non-convex nonlinear optimization problem and is NP-Hard Verma (2009). Therefore, significant attention has been devoted to finding efficient approximation of Model 1.

Deep Learning Models. Supervised Deep Learning can be viewed as the task of approximating a complex non-linear mapping from labeled data. Deep Neural Networks (DNNs) are deep learning architectures composed of a sequence of layers, each typically taking as inputs the results of the previous layer LeCun et al. (2015). Feed-forward neural networks are basic DNNs where the layers are fully connected and the function connecting the layer is given by $\boldsymbol{o} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{x} \in \mathbb{R}^n$ and is the input vector, $\boldsymbol{o} \in \mathbb{R}^m$ the output vector, $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ a matrix of weights, and $\boldsymbol{b} \in \mathbb{R}^m$ a bias vector. The function $\sigma(\cdot)$ is often non-linear (e.g., a rectified linear unit (ReLU)).

5.4. OPF Learning Goals

The goal of this thesis is to analyze the effectiveness of learning an OPF mapping $\mathcal{O} : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$: Given the loads $\{\boldsymbol{S}_i^d\}_{i=1}^n$ (vectors of active and reactive power demand), predict the set-points $\{(\Re(S_i^g), |V_i|)\}_{i=1}^N$, of the generators, i.e., their active power and the voltage magnitude at their buses. In the following $\boldsymbol{p}^g$ and $\boldsymbol{v}$ are used as a shorthand for $\Re(S^g)$ and $|V|$.

The input of the learning task is a dataset $\mathcal{D} = \{(\boldsymbol{x}_\ell, \boldsymbol{y}_\ell)\}_{\ell=1}^N$, where $\boldsymbol{x}_\ell = \boldsymbol{S}^d$ and $\boldsymbol{y}_\ell = (\boldsymbol{p}^g, \boldsymbol{v})$ represent the ℓ^{th} observation of load demands and generator set-points which satisfy $\boldsymbol{y}_\ell = \mathcal{O}(\boldsymbol{x}_\ell)$. The output is a function $\hat{\mathcal{O}}$ that ideally would be the result of the following constrained empirical minimization problem

$$\textbf{minimize:} \quad \sum_{\ell=1}^{N} \mathcal{L}(\boldsymbol{y}_\ell, \hat{\mathcal{O}}(\boldsymbol{x}_\ell)) \tag{5.8a}$$

$$\textbf{subject to:} \quad \mathcal{C}(\boldsymbol{x}_\ell, \hat{\mathcal{O}}(\boldsymbol{x}_\ell)), \tag{5.8b}$$

where the loss function is specified by

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{p}^g - \hat{\boldsymbol{p}}^g\|^2 + \|\boldsymbol{v} - \hat{\boldsymbol{v}}\|^2,$$

and $\mathcal{C}(\boldsymbol{x}, \hat{\boldsymbol{y}})$ holds if there exists voltage angles and reactive power generated that produce a feasible solution to the OPF constraints with $\boldsymbol{x} = \boldsymbol{S}^d$ and $\hat{\boldsymbol{y}} = (\hat{\boldsymbol{p}}^g, \hat{\boldsymbol{v}})$, where the *hat* notation is adopted to denoted the predictions of the model.

One of the key difficulties of this learning task is the presence of the complex nonlinear feasibility constraints in the OPF. The approximation $\hat{\mathcal{O}}$ will typically focus on minimizing (5.8a) while ignoring the OPF constraints or using penalty-based methods Fioretto et al. (2020b). Its predictions will thus not guarantee the satisfaction of the problem constraints. As a result, the validation of the learning task uses a load flow computation Π_C that, given a prediction $\hat{\boldsymbol{y}} = \hat{\mathcal{O}}(\boldsymbol{x}_\ell)$, computes its projection onto the constraint set C, i.e., the closest feasible generator set-points $\Pi_C(\hat{\boldsymbol{y}}) = \operatorname{argmin}_{\boldsymbol{y} \in C} \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2$, with C being the OPF constraint set.

5.5. Deep Learning Proxies for AC-OPF: Roadmap

To perform the aforementioned analysis the paper assumes that the OPF approximation $\hat{\mathcal{O}}$ is given by a feed-forward fully connected (FCC) neural network, as is the case for most of the work reviewed in Section 5.2. The analysis will first focus on a model that minimizes (5.8a) while ignoring the AC-OPF constraints $\mathcal{C}(\boldsymbol{x}_\ell, \hat{\boldsymbol{y}}_\ell)$ as part of the learning process. As the paper will show later, surprisingly, this model often produce highly accurate predictions. Section 5 shed light on this surprising behavior by analyzing the characteristics of the OPF objective, and the ability for the model to also study the reliability of such prediction from a OPF view-point. In doing so, the paper, analyzes what characteristics of the learning model and of the OPF problem render OPF deep learning proxy effective and reliable.

The paper attempted to categorize the complexity of the generators based on its cost and the smoothness of its iterative solution trajectory generated by a nonlinear solver. To analyze the later point, the paper introduces the notion of complexity score to provide more insights on which generators are more difficult to predict. The analysis indicates that within each category of the generators, the prediction error differs significantly.

As expected, the presence of nonlinear constraints plays a key role on the learning process of OPF problem. This thesis provides a more comprehensive view of how this factor affecting the volatility of generators dispatch at varying of the input loads and of the objective function.

Our last investigation mostly focuses on the characteristics of the learning model. The paper provides analysis of model's parameter size, input scales, activation function, and its architecture. Motivated by the volatility analysis of generators dispatch, the paper observes that the current practice of deep learning model is not able to fully capture the complexity of the power network. To reason our conclusion, we provide a novel learning framework that is able to incorporate important information during the learning process. This deep learning framework would be able to imitate the iterative method to provide a more accurate prediction.

The next sections shed light on the reasons for these behaviors. Prior to do so, we describe the training data generation setting.

5.6. Generator's Characteristics.

The Characteristics of Neural Network's Input and Output

When investing the robustness issue of DNN on AC-OPF problem, we observe that there are patterns between inputs and prediction errors. To investigate these behaviors, this section first analyzes the change in magnitude of the optimal generators dispatch at varying of the input loads and then relates this analysis to the complexity of learning to approximate the generators dispatch.

Observe that, as illustrated in the motivating Figure 5.1, the solution trajectory associated with the generator set-points on various input load parameters can often be naturally approximated by piecewise linear functions. The goal of the mapping function $\hat{\mathcal{O}}$ is thus to approximate as best as possible these picewise linear functions associated with each generator's output. Intuitively, the more volatile the function is to approximate, the harder the associated learning task will be. The paper introduces the following notion:

Definition 1 (Complexity Score)**.** Given a piecewise linear function $f : \mathbb{R}^k \to \mathbb{R}$ with p pieces, each of width h_i for $i \in [p]$, the complexity score (CS) of f, CS_f is computed by:

$$CI_f = p\mu_f\sigma_f^{\lambda},$$

where p is the number of pieces of f, μ_f is the average slope of f over p pieces, σ_f is the standard deviation of $\{L_i\}_{i=1}^{p}$, and λ is the weight multiplier.

The complexity score allows us to reason about the *volatility* of a piecewise linear function. It will become apparent later how this concept relates to the learning ability of ReLU neural networks. Notice that the two piecewise linear functions can be compared, in terms of their volatility, by their associated complexity score using a lexicographic ordering.

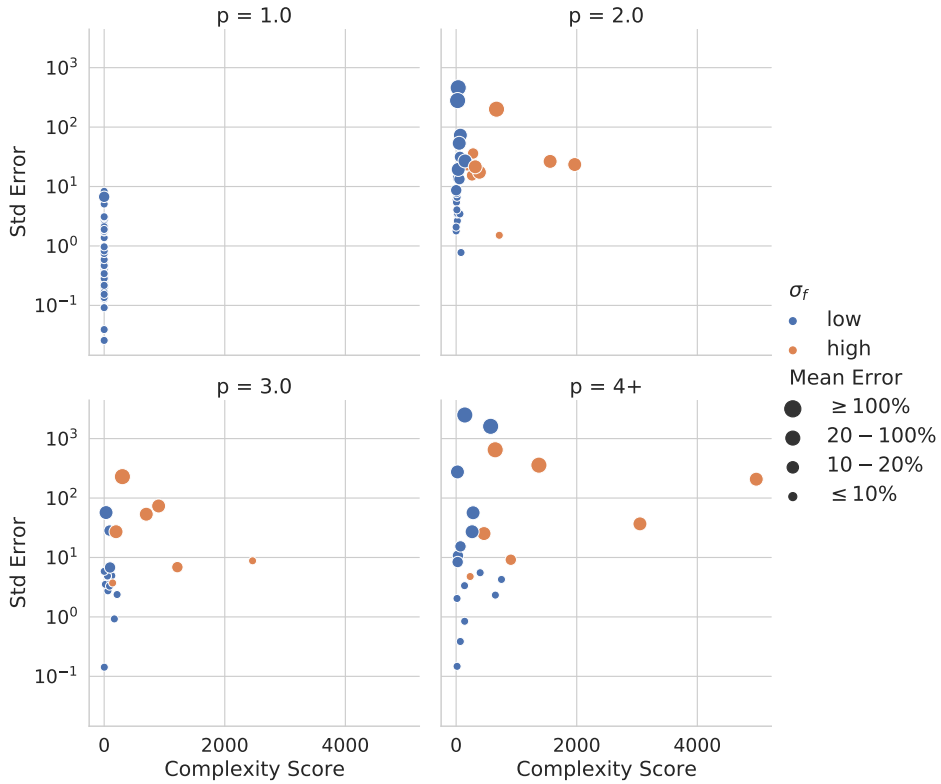


Figure 5.2: Standard deviation of prediction error (in percentage) vs complexity score, of an FCC neural network without constraint.

Since the generator dispatch trajectory can be approximated by a piecewise linear function, we refer to the complexity score of a generator g to denote the complexity score of the induced piecewise linear function of the optimal dispatch $\mathcal{O}(\boldsymbol{S}^d)$ of g at varying of the loads $\boldsymbol{S}^d$ in the domain of interest.

Figure 5.2 illustrates the relation between the complexity score of a generators with the standard deviation of prediction error (in percentage) obtained when comparing the optimal dispatches $\boldsymbol{p}^g$, associated with different input load, to their predictions $\hat{\boldsymbol{p}}^g$ obtained by an FCC

learning model as described in section 5.5. The figure reports the result of each generators for test cases IEEE-89, -118, -162, -189, and -300 ordered by their complexity score. Standard deviation of error is an important metric to investigate the robustness of the learning task under different input settings. Intuitively, DNNs have more difficulties with the generators having higher complexity scores. However, figure 5.2 points out that there are more patterns associated with each input dimension of the complexity score function. Notice four distinct patterns between prediction error and complexity score's input range: i) Generators with low complexity scores and $p = 1$ tend to have smaller error predictions in average and standard deviation. Their σ_f stays in low range. ii) The second generator group has a signature of medium range of complexity score, $p = \{2, 3\}$, low σ_f, and lower standard deviation and average error iii) Generator group with the same range $p = \{2, 3\}$ but high σ_f thus higher complexity score has higher error standard deviation, and iv) Generators with $p \geq 4$ tend to have higher complexity score, higher error standard deviation, and higher σ_f. The paper observes that DNNs performance is less robust in some groups of generators at particular load demand regions. These generators either have higher complexity scores, or higher magnitude in one or combined complexity score's input dimension. The issue at certain load demand regions will be discussed in Section VII.

Readers may notice outliers points that do not fit into discussed patterns, this happens because: i) The load test cases are randomly selected thus sometimes they do not cover the situation when the generator dispatch is more volatile than normal reducing in lower error standard deviation, ii) The value range of solution dispatch varies significantly between generators, thus affecting the computation of function's smoothness. iii) the codependent relationship between between generators input. The model performs better when predicting easy generators alone than when predicting easy and hard generators together.

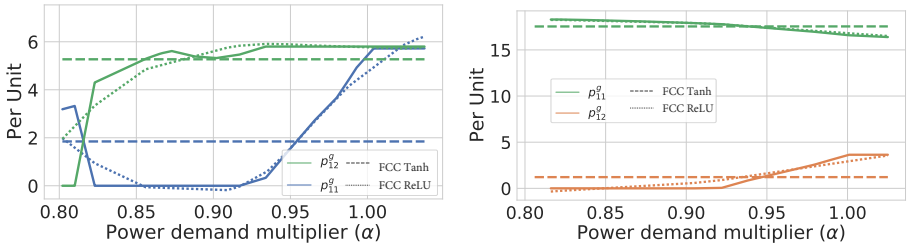
Generator's cost This section focuses on examining the cost aspect of each generator group to gain more insights on the difference in performance between groups.

The paper performs KMeans to cluster generators based on their dispatch values and cost coefficients from all power network. The cost coefficients of all generators are sorted to examine the distribution of generator clustering. The paper observes that each generator cluster has a distinctive cost distribution and dispatch trajectory pattern. Specifically, the cluster with small cost coefficients (less than 25th% quantile of cost coefficients distribution) does not change value across power demand. Generators with cost coefficients in the 50-75% quantile tend have more volatile trajectories, Finally, for the generator cluster with the highest cost (greater than 75% quantile), their trajectories stay inactive in the lower demand and increase dramatically at higher load demand. Unsurprisingly, these clustering results agree with our observed patterns about generator's complexity score and its error's standard deviation. The low cost-coefficient group contains generators with low complexity score, $p = \{1\}$, and low error's standard deviation. The most costly generators would have $p = \{2, 3, 4\}$, high σ_f, and high error's standard deviation. Finally, the generators with mid-range cost coefficients would also have $p = \{2, 3\}$ but relatively lower σ_f, and lower error's standard deviation. This observation matches the intuition that the power system would avoid using expensive generators and involve more costly generators only when constraints are harder to solve at higher load points, resulting in sudden changes in generator dispatch. Because the changes occur at later load points with such spare frequency, the regular DNN set-up does not have enough information to adjust its prediction and thus is less robust towards these difficult cases.

5.7. Network Characteristics

Activation Function This section focuses on the influence of activation function on the learning model of OPF problem. As observed above, the trajectory of the generators outputs can be described by piecewise linear functions. Next, note that ReLU networks capture piecewise linear functions Huang (2020).

This observation justifies the choice of ReLU activation function for



(a) IEEE-162 (b) IEEE-300

Figure 5.3: Accuracy of ReLU FCC vs Tanh FCC on selected generators.

DNNs used to approximate OPF solutions. Figure 5.3 illustrates a comparison between two FCCs differing only in the type of activation functions they adopt. The plots show the original generators trajectories (solid lines), the approximations learned with a ReLU network (dotted lines) and those learned with a Tanh network (dashed lines). The top and bottom plots show results for selected generators from, respectively, the IEEE-162 and IEEE-300 test cases. Notice how the ReLU network predictions can represent piecewise linear functions that better approximate the original generator trajectories, when compared to those obtained from a Tanh network.

Size of Parameter While these ReLU FCC models are compatible with the task of predicting the solutions of an OPF problem, the model capacity required to represent a target piecewise linear function exactly depends directly on the number of constituent pieces. Next, this section provides theoretical insights to link the ability of an FCC model to learn good approximations of generators trajectories of various Complexity Score.

Theorem 1 (Model Capacity Arora et al. (2016))**.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be a piecewise linear function with p pieces. If f is represented by a ReLU network with depth $k + 1$, then it must have size at least $\frac{1}{2}kp^{\frac{1}{k}} - 1$. Conversely, any piecewise linear function f that is represented by a ReLU network of depth $k+1$ and size at most s, can have at most $\left(\frac{2s}{k}\right)^k$ pieces.

The result above provides a lower bound on the model complexity to represent a given piecewise linear function. It implies that larger models may be able to better capture more complex relationships between inputs (loads) and output (generator set-points) values.

The second observation relates the load values with the total variation of the generators outputs. The following theorem bounds the approximation error when using continuous piecewise linear functions: it connects the approximation errors of a piecewise linear function with the *total variation in its slopes.*

Theorem 2. Suppose a piecewise linear function $f_{p'}$, with p' pieces each of width h_k for $k \in [p']$, is used to approximate a piecewise linear f_p with p pieces, where $p' \leq p$. Then the approximation error

$$\|f_p - f_{p'}\|_1 \leq \frac{1}{2} h_{\max}^2 \sum_{1 \leq k \leq p} |L_{k+1} - L_k|,$$

holds where L_k is the slope of f_p on piece k and $h_{\max}$ is the maximum width of all pieces.

Proof. Firstly, the proof proceeds with considering the special case in which f_p conincides in slope and value with $f_{p'}$ at some point, and that each piece of $f_{p'}$ overlaps with at most 2 distinct pieces of f_p. This is always possible when $p' \geq \frac{p}{2}$. Call I_k the interval on which $f_{p'}$ is defined by its k^{th} piece. If I_k overlaps with only one piece of f_p, then for $x \in I_k$,

$$|f_p(x) - f_{p'}(x)| = 0 \tag{5.9}$$

If I_k overlaps with pieces k and $k+1$ of f_p, then for $x \in I_k$,

$$|f_p(x) - f_{p'}(x)| \leq h_k |L_{k+1} - L_k| \tag{5.10}$$

Each of the above follows from the assumption that f_p and $f_{p'}$ are equal in their slope and value at some point within I_k. From this it follows

that on I_k,

$$\|f_p - f_{p'}\|_1 = \int_{I_k} |f_p - f_{p'}| \leq \frac{1}{2} \sum_{1 \leq i \leq p} h_k^2 |L_{k+1} - L_k|$$
$$\leq \frac{1}{2} h_{\max}^2 \sum_{1 \leq k \leq p} |L_{k+1} - L_k|, \tag{5.11}$$

so that on the entire domain of f_p and $f_{p'}$,

$$\|f_p - f_{p'}\|_1 \leq \frac{1}{2} h_{\max}^2 \sum_{1 \leq k \leq p} |L_{k+1} - L_k|. \tag{5.12}$$

Since removing the initial simplifying assumptions tightens this upper bound, the result holds. □

The result above indicates that the more volatile the generators trajectory, the harder it will be to learn. Moreover, for a neural network of fixed size, the more volatile the generator trajectory, the larger the approximation error will be in general.

The final observation is the fact that optimization problems typically satisfy a local Lipschitz condition, i.e., if the inputs of two instances are close, then they admit solutions that are close as well, i.e., for $\overset{\star}{\boldsymbol{y}}{}^{(i)} \in \mathcal{O}(\boldsymbol{x}^{(i)})$ and $\overset{\star}{\boldsymbol{y}}{}^{(j)} \in \mathcal{O}(\boldsymbol{x}^{(j)})$,

$$\| \overset{\star}{\boldsymbol{y}}{}^{(i)} - \overset{\star}{\boldsymbol{y}}{}^{(j)} \| \leq C \|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|, \tag{5.13}$$

for some $C \geq 0$ and $\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\| \leq \epsilon$, where ϵ is a small value. This observation suggests that, when this local Lipschitz condition holds, it may be possible to generate solution trajectories that are well-behaved and can be approximated effectively. Note that Lipschitz functions can be nicely approximated by neural networks as the following result indicates.

Theorem 3 (Approximation Chong (2020))**.** If $f : [0,1]^n \to \mathbb{R}$ is L-Lipschitz continuous, then for every $\epsilon > 0$, there exists some single-layer neural network ρ of size N such that $\|f - \rho\|_\infty < \epsilon$, where $N = \binom{n+\frac{3L}{\epsilon}}{n}$.

The result above illustrates that the model capacity required to approximate a given function depends to a non-negligible extent on the Lipschitz constant value of the underlying function.

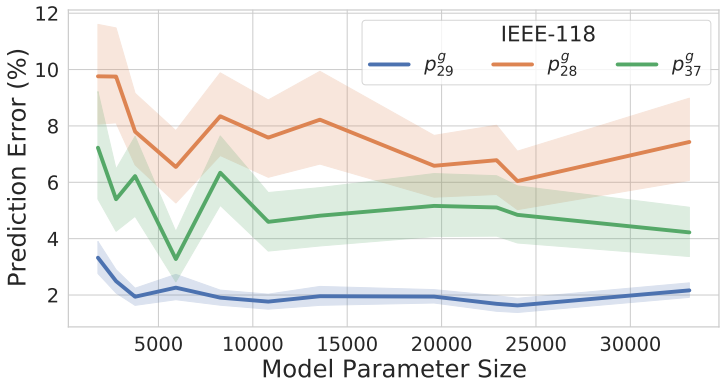


Figure 5.4: Prediction error for three key IEEE-118 generators at increasing of the FCC model complexity.

Combined with the observations reported in the previous section—showing that, for the test cases analyzed, a large number of generators have a low complexity score—the results above further illustrate the ability of DNNs to approximate OPF solutions with small average errors.

First notice that, in theory, it is to be expected that larger DNN models will be better suited to learning more complex solution trajectories (Theorem 1). However, this aspect was not observed in our experiments. Figure 5.4 illustrates this surprising behavior. It reports the prediction errors associated with the trajectories of three IEEE-118 generators at the varying of the model size. Notice how prediction errors improvements saturate quickly and that even increasing the model size substantially does not produce notable error reductions. In practice, however, the theoretical bounds rarely guarantee the training of good approximators, as the ability to minimize the empirical risk (see Equation (5.8a)) is often another significant source of error. This section implies that there are also additional factors that may affect the ability of the DNN models to learn good OPF approximators, including the presence of the OPF constraints. This aspect will be elaborated in the next section.

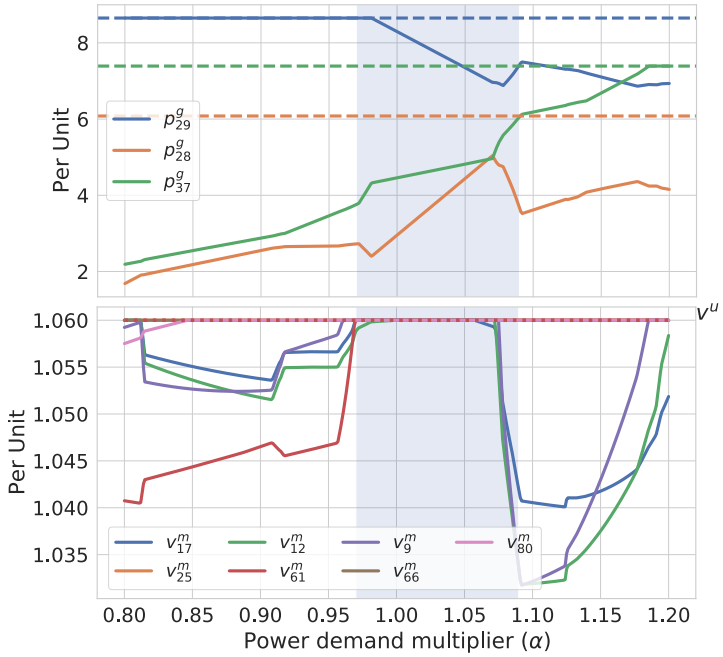


Figure 5.5: Non-linear patterns of generators around load multiplier $\alpha \in [0.97, 1.05]$ (top) and associated voltage bounds issues at various buses.

5.8. Constraints

The reminder of the section seeks to answer *why generators display irregularity patterns in some certain regions.*

To answer this question the paper analyzes generators analyzes generators with large complexity score before and during the irregularities occur. Indeed, high prediction errors pertain commonly to the optimal dispatch trajectories associated with these regions.

Figure 5.5 illustrates an example for the IEEE-118 test case, but these observations are consistent across the whole benchmark set analyzed. The figure highlights a region of *high volatility* involving several generators. The top plot shows the dispatch trajectories of three gener-

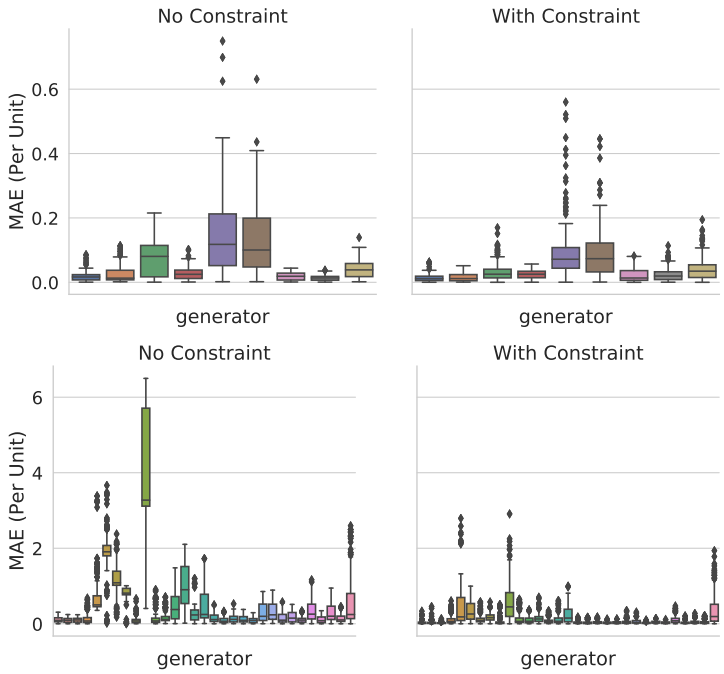


Figure 5.6: Error distributions of generators associated with problematic buses produced by FCC with and without constraint.

ators (continuous lines) at varying of the input load multipliers α and their associated upper bound limits (dashed lines) (see constraints (5.4) of Model 1). The shaded area highlights the region in which large volatilities are observed. This region also correspond to the portion associated with the higher dispatch error predictions. The bottom plot shows the trajectories of the voltage magnitude values for a selection of buses. The upper bounds (constraints (5.2)) are illustrated with a dashed line. Notice that, while the generators dispatch are within the feasible operating regions, the bottom plot highlights the presence of voltage issues on several buses. The reported buses all are associated with voltage magnitudes value which results in binding constraints (5.2) in the region of high volatility of the generators considered. *These prediction errors are*

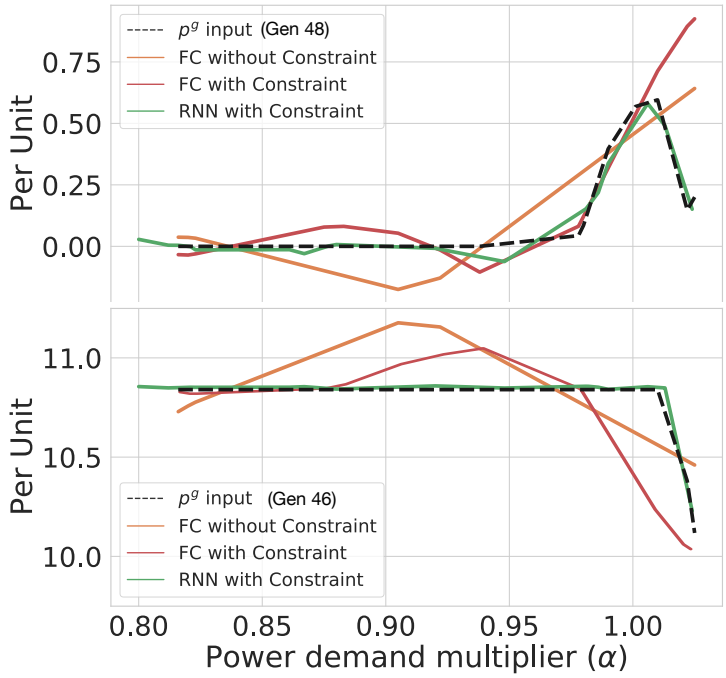


Figure 5.7: IEEE-300. Optimal generators trajectory (red) for generator 36 (top) and 48 (bottom). Predictions: FCC without constraints (orange), FCC with constraints green), and RNN with constraint (blue).

thus likely to arise as the hidden representation of the DNN does not accurately learn the operational and physical constraints which regulate the behavior of the OPF solutions. In other words, the model is unaware of these constraints. Figure 5.6 (first column) demonstrated how error of the generators that associates with bus with buses with binding constraints vary dramatically across different power demands. The error tend to have higher mean and higher standard deviation within the binding region.

Therefore, as investigated by several authors (including, Zamzam & Baker (2020); Donti et al. (2020); Fioretto et al. (2020b)) this work found that actively exploiting the problem constraints during training

Test case	FCC Without Constraint				FCC With Constraint				RNN With Constraint			
	Bound Vio	KLC Vio	LF Err. (%)	Opt. Gap (%)	Bound Vio	KLC Vio	LF Err. (%)	Opt. Gap (%)	Bound Vio	KLC Vio	LF Err. (%)	Opt. Gap (%)
IEEE-30	0.000	0.001	0.128		0.0 ± 0.0			0.001	0.0 ± 0.0	0.02 ± 0.09	0.86	0.27 ± 0.24
IEEE-118	0.01 ± 0.006	0.05 ± 0.06	102.8	2.2 ± 1.12	0.009 ± 0.007	0.04 ± 0.03	103.8	2.14 ± 1.17	0.002 ± 0.006	0.01 ± 0.03	23.6	0.310 ± 0.22
IEEE-162	0.01 ± 0.02	0.62± 0.66	25.46	1.84 ± 1.47	0.01 ± 0.01	0.17 ± 0.16	41.13	1.99 ± 1.04	0.006 ± 0.04	0.01 ± 0.04	4.478	0.28 ± 0.22
IEEE-300	0.05 ± 0.04	0.45 ± 0.63	20.63	7.68 ± 1.81	0.01 ± 0.02	0.1 ± 0.01	30.58	7.13 ± 5.07	0.008 ± 0.017	0.13 ± 0.98	10.99	0.35 ± 0.11

Table 5.1: Accuracy comparison: FCC with and without constraints and RNN models.

to be an effective mechanism to enhance the model accuracy. The constraints were added using a model similar to Fioretto et al. (2020b) which encourages the satisfaction of the OPF constraints by the means of a Lagrangian dual approach. Notice that the *constrained* and baseline models differ solely in the loss function, and not in the number of their parameters.

Table 5.1 reports the *average prediction errors* $\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_1$ over the test set, the *average load flow (LF) errors* $\|\Pi_C(\hat{\boldsymbol{y}}) - \boldsymbol{y}\|_1$ which compare the closest feasible solution $\Pi_C(\hat{\boldsymbol{y}})$ of the predictions $\hat{\boldsymbol{y}}$ with the optimal quantities $\boldsymbol{y}$, and the *average optimality gap*, as $\frac{|O(\Pi_C(\hat{\boldsymbol{p}^g}))-O^*(\boldsymbol{p}^g)|}{O^*(\boldsymbol{p}^g)}$, with O being the associated OPF cost. It also compares the average absolute constraint violations (in p.u.) for the set-points bounds (constraints (5.2) and (5.4)) and the KCL (constraint (5.6)). Notice how the constrained model reduces the constraint violations, when compared to the baseline. Figure 5.6 also illustrates how the mean and standard deviation error of the generators associated with the problematic buses are reduced when introducing Lagrange dual loss to the training process. Unsurprising, constrained DNN versions are more robust than the unconstrained models.

This aspect is also evident in Figure 5.7, which compares the prediction trajectories of the FCC model with (yellow curves) and without (red curves) constraints for two high-complexity IEEE-300 generators. Notice that the constrained model predictions follow more closely the original trajectories when compared to the simple model.

This aspect is surprising from an empirical risk minimization perspective: Including constraints using Lagrangian-based penalties adds additional terms to the loss function which can be interpreted as further regularizing terms, and thus, it may be expected they would reduce the

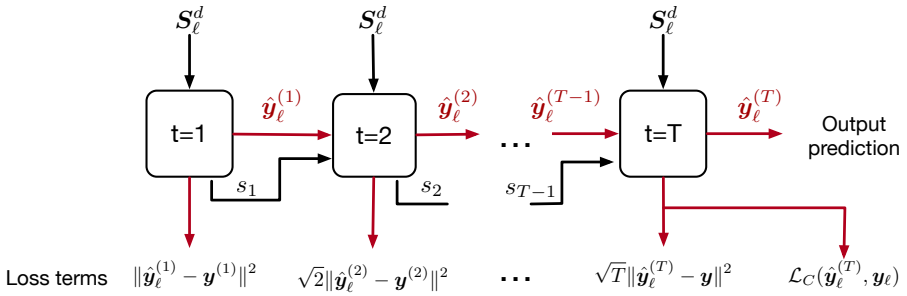


Figure 5.8: RNN model Overview.

model variance further.

However, Figure 5.7 also highlights some drawbacks of the constrained model. Despite its improved accuracy (and its ability to approximate precisely many *easy* generators) its predictions tend to discard the rapid changes in trajectories of the generators outputs (see bottom plot). From a data representation point of view, these cases (where the change in trajectory occurs) represent *outliers* and thus are hard to predict. This observation motivates the introduction of a novel model described next.

5.9. A Novel RNN-based Learning Framework

Motivation The issue observed above could be partially addressed by providing additional training data to the learning task with the goal of more suitably representing the inputs associated with the *outlier* set-points. Creating this data is, however, a very challenging task. It is unknown a-priori which set-point, within a trajectory, may be uncommon. Additionally, generating the input loads associated to a desired set-point would be an extremely challenging operation.

While generating additional targeted data is thus unfeasible, the paper reveals that DNNs has robustness issues when predicting generators with high cost coefficients and during binding constraint regions. To find the mitigation solution, the paper looks at how the traditional optimization system predict these generators before and during binding

constraints.The selected solver is IPOPT Wächter & Biegler (2006), a popular nonlinear solver implementing a primal-dual interior point line search filter method to find a local optimal solution to a given problem instance. IPOPT iteratively search for feasible solution and terminate when convergent condition is reached. The paper investigated IPOPT's iterative solution trajectory of each generator in different groups as mentioned in section VI.

Figure 5.9 (top) displays the solutions trajectory across load demand from selected generator at every group at test case IEEE-300. As mentioned, the dispatch solution trajectory of each generator group has unique volatility patterns. Figure 5.9 (bottom) shows the iterative solution trajectory outputed by IPOPT for the same generators before, during, and after binding constraints ($\alpha = \{0.87, 0.97, 1.03\}$). Firstly, notice the iterative trajectory of the hard generator groups are more volatile than the easier ones. Secondly, the iterative trajectories display more irregular patterns at $\alpha \in \{0.97, 1.03\}$. The volatility in the iterative trajectory curves is the result of IPOPT refreshing its states when getting into saddle regions. Notice the strong correlation between the volatility of iterative solution trajectory and the difficulty of the learning task. IPOPT *seems to behave differently to each generator group at different load points.* This observation connects the generators volatility with the hardness of the model to capture its output trajectory. Since only load demand information are supplied to predict solution dispatch, it is hard for DNNs to pick up these patterns to successfully foresee the sudden changes in generator dispatch. In other words, it is essential to incorporate information about the generator's volatility into the training process of DNNs. One intuitive solution is to design the system that can imitate the solver while taking advantage of big training data.

Training Data The paper analyzes the learning models behavior trained on test cases from the NESTA library Coffrin et al. (2014). For presentation simplicity, the analysis focuses primarily on the IEEE 118, 162 and 300-bus networks. However, the results are consistent across the entire benchmark set. The ground truth data are constructed as follows: For each network, different benchmarks are generated by altering the amount of nominal load $\boldsymbol{x} = \boldsymbol{S}^d$ within a range of $\pm 20\%$. For a given *load multi-*

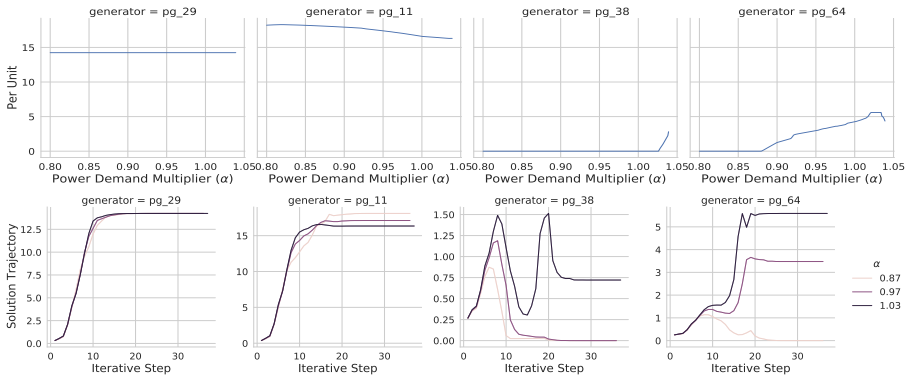


Figure 5.9: Selected generators from the forth group (first two columns), second group (third and forth column), and third group (last column). The first row shows their iterative solution trajectory at $\alpha \in \{0.87, 0.97, 1.03\}$. The second row display the optimal generator dispatches across loads.

plier α sampled uniformly in the interval $[0.8, 1.2]$, a load vector $\boldsymbol{x'} = \boldsymbol{S}^{d'}$ is generated by perturbing each load value $\boldsymbol{S}_i^d$ independently with additive Gaussian noise centered in α and such that $\sum_i \boldsymbol{S}_i^{d'} = \alpha \sum_i \boldsymbol{S}_i^d$. A network value that constitute a dataset entry $(\boldsymbol{x'}, \boldsymbol{y'} = \mathcal{O}(\boldsymbol{x'}))$ is a feasible OPF solution obtained by solving the AC-OPF problem detailed in Model 1. The data are normalized using the per unit (pu) system. The experiments use a 80/20 train-test split and report results on the test set.

Set-Up and Results This section introduces a DNN model for OPF predictions which relies on deep autoregressive Recurrent Neural Networks (RNN). RNNs are a powerful tool to learn from sequential data and have been vastly adopted in domains including natural language processing and computer vision Kreider et al. (1995); Connor et al. (1994); Liu et al. (2020). An autoregressive model is typically used in time-series modeling where the current time step value z_t depends linearly on some value $z_{t'}$ with $t' < t$. Similarly, autoregressive RNNs condition the pre-

Test Case	FCC	RNN	**Test Case**	FCC	RNN
IEEE-118	11.4	0.007	IEEE-300	47.8	0.04
IEEE-162	14.8	0.005	PEGASE-1354	154	0.32
EDIN-189	3.4	0.013	RTE-2868	2907	1.64

Table 5.2: RNN vs FCC: Model parameter size (Million).

diction of the current time step on the predictions of the previous steps. They thus are a natural fit for the intended purpose.

The proposed model is illustrated in Figure 5.8. The model is composed of T sequential Long Short Memory Term (LSMT) units. For unit $t \in [T]$, the model takes as input the demands $\boldsymbol{x} = \boldsymbol{S}^d$ as well as the embedding $\boldsymbol{y}^{(t-1)}$ outputted by unit $t-1$, and the *state* s_{t-1} of unit $t-1$. The first unit $t = 1$ is special and only considers the input demands $\boldsymbol{x}$. The model is trained with the following loss:

$$\sum_{\ell=1}^{N}\sum_{t=1}^{T}\sqrt{t}\mathcal{L}(\boldsymbol{y}_\ell^{(t)}, \hat{\boldsymbol{y}}_\ell^{(t)}) + \mathcal{L}_C(\boldsymbol{y}_\ell, \hat{\boldsymbol{y}}^{(T)}), \tag{5.14}$$

where $\mathcal{L}_C$ is the Lagrangian loss involving the prediction from the last unit to encourage constraint satisfaction, equivalently to that adopted by the constrained variant of the FCC model. The $\sqrt{t}$ multiplicative factor is adopted to give larger weights to the latter units. The model returns $\hat{\boldsymbol{y}}^{(T)}$ as its prediction, which is the output of the recurrent final unit.

The predictions of the proposed model are summarized in Table 5.1 (right). Notice how the model can reduce the load flow errors and optimality gaps by one order of magnitude when compared with the best FCC results. Notably, the RNN model predictions are much closer to satisfy the KLC than those produced by the constrained version of the FCC model. This is important as KLC are notoriously hard to satisfy for the predictions of DNN models Fioretto et al. (2020b). The ability of this model to capture robustly rare changes in generators trajectory can be appreciated in Figure 5.7.

Finally, Table 5.2 reports a comparison of the number of parameters (proxy to memory footprint) required by the FCC and the proposed

RNN models. Notice that the FCC grow very large with the size of the processed test case highlighting scalability issues, as also observed in Chatzos et al. (2020), which reported the inability of these models to fit in memory for test cases larger than 2000 buses. In contrast, the proposed RNN model does not incur this drawback rendering it applicable to very large power systems.

5.10. Conclusions

This thesis was motivated by the recent development around using deep neural networks (DNN) to approximate the solutions of Optimal Power Flow (OPF) problems. While these learning models show encouraging results, little is known on why they predict OPF solutions accurately, as well as about their predictions robustness. The paper provided a step forward to address this knowledge gap. It studied the connection between the volatility of the generators outputs with the ability of a learning model to approximate it, showing that many test cases are characterized by a large number of generators which are easy to predict. It also showed that operational and physical constraints are necessary to capture the complexity of the predictions. Finally, it proposed a new learning model based on recurrent neural networks, that was not only able to improve the prediction accuracy over existing supervised learning approaches, but also reduced the memory requirements.

CHAPTER 6

End-to-End Optimization and Learning of Fair Court Schedules

"Justice delayed is justice denied."

William Gladstone

This chapter takes a step toward integrating Predict-Then-Optimize (PtO) and Learning-to-Optimize (LtO). It proposes two frameworks in which learned optimization models—in the LtO sense—can be integrated end-to-end with predictive models. The key motivation is that such models are not only differentiable by construction but also efficiently callable, making them well-suited for inclusion in trainable pipelines. The approach is demonstrated in the context of court scheduling, where fairness is critical, as outdated practices like "cattle call" scheduling disproportionately burden defendants, leading to delays and systemic inequalities. Developing fairness-aware models in this domain remains challenging due to logistical constraints, data scarcity, and complex optimization requirements. This work is part of the author's unpublished research Dinh et al. (2024d).

6.1. Optimization and Learning of Fair Court Schedules

Criminal courts in the United States handle millions of cases annually, requiring schedules that balance the preferences and availability of courts, prosecutors, defendants, and defense counsel Graef et al. (2023); Gouldin (2024). However, jurisdictions often prioritize stakeholders unequally, frequently disregarding defendants' preferences. Many courts rely on rigid scheduling models, such as "cattle call" practices, where defendants are required to arrive early and wait extended periods for their cases to be called. Late arrivals are often marked as failures to appear, resulting in significant penalties such as arrest or detention National Council of Juvenile and Family Court Judges (2021).

Ensuring defendants return for trials and pretrial appearances has influenced pretrial decision-making for centuries Gould et al. (2016); Baughman (2018). Nonappearances not only delay justice but also disrupt court proceedings and impose significant rescheduling costs on courts, legal personnel, witnesses, and other defendants Gouldin (2024); Graef et al. (2023). Courts typically address this through sanctions like bench warrants, fines, or new charges rather than implementing time-certain

systems, which pose logistical and implementation challenges. Nevertheless, a significant number of defendants still fail to appear, making effective attendance interventions a crucial pretrial reform priority Gouldin (2024). Research highlights that indigence remains the most consistently associated factor to non appearance rates Zettler & Morris (2015), and defendants with unstable employment, caregiving responsibilities, or limited transportation face additional challenges, particularly in cases with extended durations and multiple required appearances McAuliffe et al. (2023).

Courts and commentators increasingly recognize that improving pretrial outcomes requires reforms in *court scheduling practices*. Supportive frameworks addressing barriers like transportation, work, and childcare have been shown to increase appearance rates. Alternatives to traditional "cattle call" systems, such as block or time-certain scheduling, can reduce burdens on defendants and improve compliance Gouldin (2024); Criminal Justice Innovation Lab (CJIL) (2022). However, designing schedules that accommodate all stakeholders, particularly defendants, remains challenging. While data from judges and attorneys is relatively easy to collect, logistical and financial barriers often prevent courts from gathering similar data about defendants. Even when some preference data is available, existing systems lack formal mechanisms to ensure procedural fairness in scheduling, which is essential for promoting justice. Developing fairness-aware scheduling models is further complicated by the need to integrate complex constraints into machine learning frameworks and the scarcity of comprehensive datasets for validation.

Thus, there is a clear need for a system that reduces nonappearance while ensuring fairness for all defendants. However, addressing these issues is far from trivial. Scheduling is fundamentally a decision-making process, often framed as a utility maximization problem. The goal is to assign people to time slots in a way that fairly maximizes overall utility. In court scheduling, this utility can be thought of as a function of an individual's preference, even though such data may be unavailable or uncertain. The task then becomes learning how to map defendants' socioeconomic and demographic characteristics to court schedules that meet fairness criteria. Traditional approaches typically rely on a two-

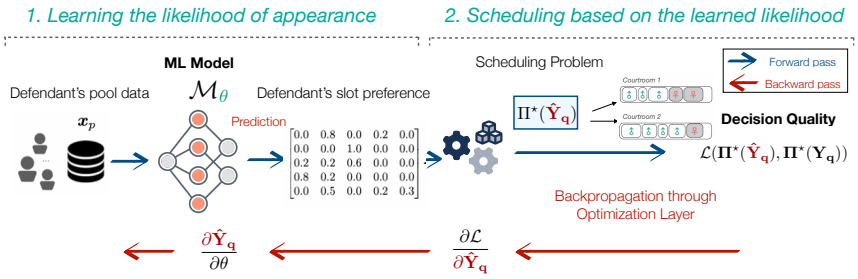


Figure 6.1: Proposed end-to-end framework for learning to schedule. Given candidates' socioeconomic and demographic identifiers, a neural network is trained to predict their preference score for each time slot. A differentiable surrogate model uses a predicted score to attain assignments and decision quality loss.

stage process, where machine learning models first predict preferences and optimization techniques then generate schedules based on these predictions. However, this two-stage approach has been shown to lead to suboptimal outcomes Mandi et al. (2024b). A key issue arises from the misalignment between prediction errors and the final task utilities, which, in the case of this thesis, account for the effectiveness and fairness of the court schedule. Since the prediction model is trained independently of the optimization task, inaccuracies in predicting defendants' preferences can directly impact the quality of the resulting schedule. This separation often leads to suboptimal outcomes, as prediction errors can propagate through the optimization step, negatively impacting the schedules' fairness and effectiveness.

To address these challenges, this thesis presents a joint optimization and learning framework that integrates machine learning models trained end-to-end with efficient scheduling algorithms. This approach aims to create an effective court scheduling system that maximizes a principled notion of fairness, even under uncertainty in defendants' scheduling preferences. A schematic overview of the framework is provided in Figure

6.1. The framework illustrates how a shift to time-certain scheduling can better accommodate defendants' needs and preferences while enhancing system-wide outcomes. As the first to explore the application of machine learning in court scheduling reform, this thesis contributes to the advancement of fairer and more efficient judicial systems.

Contributions. This thesis makes the following technical contributions:

1. It formalizes a fair scheduling problem tailored for pretrial court settings, where data about defendants' scheduling preferences is often incomplete or uncertain. The problem is formulated as a nonlinear integer program with a piecewise-defined fairness objective, over uncertain values in defendants' scheduling preferences. This formulation addresses the real-world challenges courts face in obtaining reliable preference data from defendants.
2. It presents a novel framework that integrates machine learning with optimization techniques, allowing to handle uncertainty in defendants' preferences by directly optimizing a decision loss function, thereby improving the robustness and fairness of the scheduling outcomes.
3. To overcome the computational complexity of the integer programming formulation, it proposes an end-to-end trainable model that predicts feasible schedules via the parameters of an efficient matching algorithm. This enables training on the fairness objective directly, enhancing scalability as well as satisfaction of the fairness objective.
4. Given the lack of publicly available datasets on court scheduling-dependent preferences, it presents a process for generating realistic synthetic datasets. This process incorporates demographic data on arrestee populations and estimates the prevalence of barriers such as childcare, transportation, and employment challenges. These synthetic datasets enable the training and validation of fairness-aware scheduling models that account for diverse and sensitive demographic factors.

6.2. Related Work

Court scheduling. Court scheduling refers to the process of organizing the court's calendar and scheduling cases at the appropriate time, location, and format for all necessary participants. This fundamental judicial function considers resources, schedules, and due process requirements Lane & Cox (1976); Graef et al. (2023). Despite its critical role, however, criminal court scheduling has been understudied partly because traditional approaches view defendants' nonappearance as a disruption rather than considering the barriers defendants face Gouldin (2024). Most existing research focuses on the efficacy of court reminder systems or primarily aims to improve pretrial outcomes for individual defendants McAuliffe et al. (2023); Criminal Justice Innovation Lab (CJIL) (2022).

Innovation in court scheduling faces numerous barriers. The traditional and hierarchical structure of courts resists efforts to streamline scheduling processes Ferguson (2022); Institute for Law and Social Research (INSLAW) (1988). Additionally, due process requirements and the adversarial nature of criminal proceedings further complicate reform. The complexity of the judicial process itself is a considerable hurdle to innovation Institute for Law and Social Research (INSLAW) (1988). Scheduling must account for multiple factors, such as the availability of judges, defendants, attorneys, and courtrooms. The complex landscape causes hesitation among administrators, who prefer to maintain the status quo. The system is marked by significant inefficiencies, causing delays and backlogs that can affect justice delivery. The lack of data regarding court scheduling practices is another significant hurdle to reforming court scheduling.

Advances in Machine Learning and Optimization for scheduling. Recent literature has focused on developing constrained optimization models that are trained end-to-end with machine learning models Kotary et al. (2021). In the Predict-Then-Optimize setting, a machine learning model predicts unknown coefficients for an optimization problem. Then, backpropagation through the optimal solution of the resulting problem allows for end-to-end training of its objective value, under ground-truth coefficients, as a loss function. The primary challenge lies

in backpropagating through the optimization model, for which various techniques have been proposed. Several techniques have been proposed to address this challenge. Early work focused on implicit differentiation through the first-order optimality conditions of unconstrained or smooth convex functions Gould et al. (2016). Subsequent research extended this approach to constrained problems by differentiating through Karush-Kuhn-Tucker (KKT) conditions, particularly for quadratic programs (QPs) Amos & Kolter (2017b); Amos et al. (2019). Other techniques include the differentiation of specific problem classes, such as sorting and ranking Blondel et al. (2020), linear programming Mandi & Guns (2020), and convex cone programming Agrawal et al. (2019b).

For discrete optimization problems, including mixed-integer and constraint programs, the challenge lies in the discontinuity of mappings with respect to input parameters. To address this, smoothing techniques have been developed to approximate the optimization mappings, producing gradients that are more useful for training Wilder et al. (2019); Ferber et al. (2020); Mandi & Guns (2020). While these approaches have demonstrated success in various domains, they often focus on general-purpose optimization objectives and do not explicitly address fairness considerations. This thesis builds on these advances by introducing a novel framework for fair optimization in scheduling, leveraging ordered weighted average (OWA) operators to encode fairness principles such as impartiality, equitability, and Pareto efficiency. These principles are particularly critical in the context of court scheduling, where fairness is not merely a desirable property but a fundamental requirement for improving access to justice.

6.3. Motivations and Problem Setting

Scheduling defendants in pretrial processes involves arranging their court appointments in a way that aligns with their preferences for various appointment times. In this context, a defendant's preference indicates their likelihood of attending a scheduled appointment. A significant challenge is that these preference data are often unavailable, necessitating estimation from available data to create meaningful schedules. This challenge

can be partially addressed by employing a pipeline where a machine learning (ML) model predicts defendants' preferences, which are then used as inputs for the scheduling task, as illustrated in Figure 6.1.

In many data-driven decision-making systems, learning and optimization are treated separately: an ML model is trained for predictive accuracy, and its predictions guide the decision-making task. This "two-stage" approach works well when predictive models are highly accurate, as precise predictions tend to inform the correct decisions. However, in practice, predictive models are rarely perfect, particularly when preference data are sparse or incomplete. These errors can lead to suboptimal schedules that misalign with defendants' actual preferences and may exacerbate *unfairness* by disproportionately affecting certain groups, resulting in biased scheduling outcomes Kotary et al. (2021). Left unaddressed, such disparities can lead to a "poor get poorer" effect with serious societal consequences. *To mitigate these disparate impacts, this thesis proposes an integrated optimization and learning framework for scheduling defendant court visit times to certify a desired fairness requirement.*

Problem Setting

The setting studied in this thesis considers a training dataset $\boldsymbol{D} = (\boldsymbol{x}_p, \boldsymbol{g}_p, \boldsymbol{Y}_p)_{p=1}^{N}$ of N elements, each describing a pool of n defendants to be scheduled on a given day. For each pool indexed by $p \in [N]$, $\boldsymbol{x}_p \in \mathcal{X}$ describes a list $(x_i^p)_{i=1}^{n}$ of n defendants to schedule, with each item x_i^p defined by a feature vector. These feature vectors encode representations of the individuals to schedule, e.g., their socioeconomic identifiers, addresses, and accusations. The elements $\boldsymbol{g}_p = (g_i^p)_{i=1}^{n}$ describe protected group attributes of the defendants in some domain $\mathcal{G}$. For example, they may denote employment type, whether the defendant has access to transportation, or whether

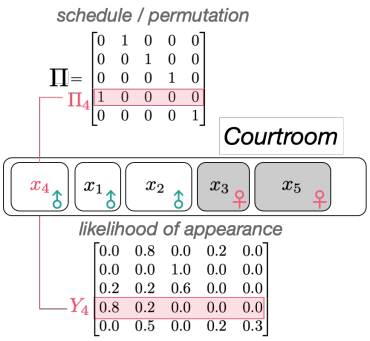


Figure 6.2: Court scheduling example.

they have childcare obligations. Together, the unprotected and protected attributes (x_i^p, g_i^p) provide a description of the defendant's i characteristics in pool p. Finally, the element $\mathbf{Y_p} \in \mathcal{Y} \subseteq \mathbb{R}^{\mathbf{n}\times\mathbf{n}}$ are supervision labels $(\boldsymbol{Y}_i^p)_{i=1}^n$ that associate a vector of non-negative values with each person $i \in [n]$, and describe the preference of individual i with respect to each slot in a possible schedule.

The goal is to learn a mapping between features $\boldsymbol{x}_p$ of a pool p of n defendants and a *schedule* which fairly optimizes the satisfaction of their preferences as indicated by $\boldsymbol{Y_p}$. A schedule is represented by a permutation of the list $[n]$, which determines the appointment times assigned to each defendant. When clear from the context, we may drop the various elements of the superscript or subscript p. For modeling purposes, and when clear from context, we represent a schedule $\mathbf{\Pi}$ by a permutation matrix, so that $\mathbf{\Pi}_{ij}$ indicates the assignment of individual i to the time slot j. Additionally, let $\mathbf{\Pi}_i$ represent the i^{th} row of the matrix $\mathbf{\Pi}$.

Figure 6.2 illustrates these elements in our problem setting. It shows a matrix $\mathbf{\Pi}$ representing a permutation that matches defendants to appointment slots. The highlighted defendant's feature vector x_4, is illustrated in light of their respective row $\mathbf{\Pi}_4 = [1, 0, 0, 0, 0]$ in $\mathbf{\Pi}$, which indicates that defendant 4 is to be scheduled first. The contextual information provided by x_4 will be used as input to an ML model to produce an estimated preference vector $\boldsymbol{y}_4 = [0.8, 0.2, 0, 0, 0]$, which associates the defendant's likelihood of appearance to the different court visit times. The protected group information $\boldsymbol{g}$, expressing gender in the example above, is represented by the different colored (white/gray) cells.

Learning Objective. The goal in our setting is to predict, for any pool of defendants $(\mathbf{x}, \mathbf{g}, \mathbf{Y})$, a permutation matrix $\mathbf{\Pi}$ representing a schedule which fairly maximizes the *utility* of each defendant, defined as:

$$u_i(\mathbf{\Pi}, \boldsymbol{Y}) = Y_i^T \Pi_i, \tag{6.1}$$

for defendant i. When Y_{ij} represents the likelihood of defendant i attending appointment j, this quantity represents the likelihood of defendant i attending their assigned appointment. Given any $\mathbf{\Pi}$ and $\boldsymbol{Y}$, computing

Equation (6.1) for each defendant results in a *utility vector* $\boldsymbol{u} \in \mathbb{R}^n$:

$$\boldsymbol{u}(\boldsymbol{\Pi}, \boldsymbol{Y}) = \mathit{diag}(\boldsymbol{Y}^T \boldsymbol{\Pi}). \tag{6.2}$$

In this setting, it is natural to ask:

> What does it mean to fairly optimize the utilities of each defendant, which are independent objectives?

Any optimization method requires a single-valued objective function to be defined. For example, one may choose to optimize the total utility of a scheduling policy, defined as the sum of individual utilities:

$$\mathbf{1}^T \boldsymbol{u}(\boldsymbol{\Pi}, \boldsymbol{Y}) = \mathrm{Tr}(\boldsymbol{Y}^T \boldsymbol{\Pi}), \tag{6.3}$$

which is a single-valued function of $\boldsymbol{u}$. However, a schedule $\boldsymbol{\Pi}$ which optimizes total utility does not account for the lowest utilities among all defendants, which may be arbitrarily low. This makes the total utility an undesirable objective in our framework, since *accepting low utilities for some defendants increases their odds of nonappearance.*

Instead we seek an aggregation of individual utilities which, when optimized, raises the utilities of all defendants uniformly, to the extent possible. Maximizing the minimum utility is one approach, but it leads to *pareto inefficient* solutions, meaning that some defendants' utilities are needlessly suboptimal, in the sense that $\boldsymbol{\Pi}$ can be chosen to raise those utilities without harm to the utilities of others Ogryczak & Śliwiński (2003). This can be viewed as a needless compromise to total utility. *The notion of court scheduling studied in this work calls for aggregation of individual utilities which, when optimized, raises the lowest individual utilities while maintaining pareto efficiency.*

A well-known aggregation function possessing these properties is the *Ordered Weighted Average* (OWA) Yager (1993). The OWA operator and its fairness properties are formally defined next, before delving into the description of the proposed framework for learning schedules which maximize the OWA-aggregated utility, in Section 6.6.

6.4. Preliminaries: Fair OWA Aggregation

The *Ordered Weighted Average* (OWA) operator Yager (1993) is a class of functions used for aggregating multiple independent values in settings requiring multicriteria evaluation and comparison Yager & Kacprzyk (2012). Let $\mathbf{y} \in \mathbb{R}^m$ be a vector of m distinct criteria, and $\tau : \mathbb{R}^m \to \mathbb{R}^m$ be the sorting map in increasing order so that $\tau_1(\mathbf{y}) \leq \tau_2(\mathbf{y}) \leq \cdots \leq \tau_m(\mathbf{y})$. Then for any $\mathbf{w}$ satisfying $\{\mathbf{w} \in \mathbb{R}^m \mid \sum_i w_i = 1, \mathbf{w} \geq 0\}$, the OWA aggregation with weights $\mathbf{w}$ is defined as a linear functional on $\tau(\mathbf{y})$:

$$\mathrm{OWA}_{\mathbf{w}}(\mathbf{y}) = \mathbf{w}^T \tau(\mathbf{y}), \tag{6.4}$$

which is piecewise-linear in $\mathbf{y}$ Ogryczak & Śliwiński (2003).

This thesis focuses on a specific instance of OWA, commonly known as *Fair OWA* Ogryczak et al. (2014), characterized by weights arranged in descending order: $w_1 > w_2 \ldots > w_n > 0$. With monotonic weights, Fair OWA is also concave. Fair OWA objectives are increasingly popular in optimization as fairness gains attention in decision-making processes.

The following three properties of Fair OWA functions are crucial for their use in fairly optimizing multiple objectives:

1. *Impartiality* ensures that Fair OWA treats all criteria equally. This means that for any permutation $\sigma \in \mathcal{P}_m$, where $\mathcal{P}_m$ is the set of all permutations of $[1, \ldots, m]$, the OWA aggregation with weights $\mathbf{w}$ yields the same result for any permutation of the input vector $\mathbf{y}$.
2. *Equitability* guarantees that marginal transfers from a criterion with a higher value to one with a lower value increase the OWA aggregated value. This condition holds that $\mathrm{OWA}_{\mathbf{w}}(\mathbf{y}_\epsilon) > \mathrm{OWA}_{\mathbf{w}}(\mathbf{y})$, where $\mathbf{y}_\epsilon = \mathbf{y}$ except at positions i and j where $(\mathbf{y}_\epsilon)_i = \mathbf{y}_i - \epsilon$ and $(\mathbf{y}_\epsilon)_j = \mathbf{y}_j + \epsilon$, assuming $y_i > y_j + \epsilon$.
3. *Monotonicity* ensures that $\mathrm{OWA}_{\mathbf{w}}(\mathbf{y})$ is an increasing function of each element of $\mathbf{y}$. This property implies that solutions optimizing the OWA objectives (6.4) are Pareto Efficient solutions of the underlying multiobjective problem, thus no single criteria can be raised without reducing another Ogryczak & Śliwiński (2003).

This last aspect is crucial in optimization, where Pareto-efficient solutions are preferred over those that do not possess this attribute. Taken

together, these properties define a notion of fairness in optimal solutions known as *equitable efficiency* Ogryczak & Śliwiński (2003), which is of particular interest for the pretrial court scheduling optimization.

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. This thesis employs fair OWA functions as a fair measure of overall utility with respect to defendants' preferences, ensuring that the resulting court schedules uphold principles of fairness and equity. As shown in the next section, while achieving these properties is desirable for court schedules, it also introduces a challenging optimization problem over the space of permutation matrices. Addressing such computational challenges is one of the key technical objectives of the paper.

6.5. Fair Optimization of Court Schedules

Using the concepts introduced above, we can form an optimization program to fairly maximize utilities in court scheduling. For a pool of defendants represented by $(\boldsymbol{x}, \boldsymbol{g}, \boldsymbol{Y})$, with known preferences $\boldsymbol{Y}$, the scheduling matrix $\boldsymbol{\Pi}$ that maximizes the OWA-aggregated utility over all defendants is modeled as a solution to an integer program:

$$\boldsymbol{\Pi}^{\star}(\boldsymbol{Y}) = \operatorname{argmax}_{\boldsymbol{\Pi}} \ \operatorname{OWA}_{\mathbf{w}}\left(\boldsymbol{u}(\boldsymbol{\Pi}, \boldsymbol{Y}) \right) \tag{6.5a}$$

$$\text{subject to:} \quad \boldsymbol{\Pi} \in \{0,1\}^{n \times n} \tag{6.5b}$$

$$\sum_i \Pi_{ij} = 1, \ \ \sum_j \Pi_{ij} = 1, \ \ \forall i, j \in [n]. \tag{6.5c}$$

where the utility vector $\boldsymbol{u}(\boldsymbol{\Pi}, \boldsymbol{Y})$ is defined as in equation (6.2). Here and throughout, we make a standard choice of fair OWA weights $w_j = \frac{n-i+1}{n}$, known as the *Gini indices* Do & Usunier (2022) Note that the constraints (6.5c) hold that each row and column of $\boldsymbol{\Pi}$ must sum to 1. Together with (6.5b), this ensures a single value of 1 in each row and column, so that problem (6.5) models the permutation matrix with maximal fair OWA-aggregated utility.

6.5.1 Group Fairness

The objective in problem (6.5) is to maximize the OWA-aggregated utility vector with respect to each individual defendant. We may also extend the use of the OWA operator to model different fairness objectives within our framework. In particular, we are interested in *group fairness*, a concept widely employed, for example, in web search rankings Singh & Joachims (2018, 2019); Do & Usunier (2022); Zehlike & Castillo (2020); Zehlike et al. (2017).

Individuals may grouped by any category between which fair outcomes are desired: for example by gender, race, or socioeconomic status. For any schedule $\mathbf{\Pi}$ the *group utility* u^g of group g is defined as the mean utility of all defendants in that group. For any group indicator g, let $\mathcal{S}_g$ be the set of defendants' indices belonging to that group. Then:

$$u^g(\mathbf{\Pi}, \boldsymbol{Y}) = \frac{1}{|\mathcal{S}_g|} \sum_{i \in \mathcal{S}_g} \boldsymbol{Y}_i^T \mathbf{\Pi}_i. \tag{6.6}$$

Let a *partition* $\mathcal{G} = \{g_i^p : p \in [N], i \in [n]\}$ represent the set of all unique protected group indicators (e.g. male and female, or the set of all income brackets). Our notion of group fairness is to maximize the OWA aggregation of all group utilities over a chosen partition. Letting $\boldsymbol{u}^{\mathcal{G}}$ be the vector of all group utilities $[u^g : g \in \mathcal{G}]$,

$$\mathbf{\Pi}^\star(\boldsymbol{Y}) = \text{argmax}_{\mathbf{\Pi}} \ \text{OWA}_{\mathbf{w}} \left(\boldsymbol{u}^{\mathcal{G}}(\mathbf{\Pi}, \boldsymbol{Y}) \right) \tag{6.7a}$$

$$\text{subject to:} \quad \mathbf{\Pi} \in \{0,1\}^{n \times n} \tag{6.7b}$$

$$\sum_i \Pi_{ij} = 1, \ \sum_j \Pi_{ij} = 1, \ \forall i, j \in [n] \tag{6.7c}$$

Of course, the models (6.7) and (6.5) coincide when each individual defendant constitutes their own group. In Section 6.7, we will evaluate the ability of our framework to fairly optimize group utilities with respect to various partitions.

6.5.2 Complexity of the Optimization Models

It is important to remark on the complexity of the integer program (6.7). Since the OWA function is piecewise linear Ogryczak & Śliwiński (2003),

they can be categorized as *nonlinear integer programs* with a piecewise-defined objective and is thus NP-hard. Given the form of this integer program, traditional integer programming approaches cannot be applied directly to (6.7). Specifically, methods such as branch-and-bound and cutting plane algorithms, which are commonly used for solving integer linear programs (ILPs), struggle with the piecewise linear nature of the OWA objective. These ILP approaches typically rely on linearity and convexity to efficiently explore the solution space, but the nonlinear, piecewise-defined objective complicates the feasible region and increases the computational burden.

Furthermore, optimizing a schedule for n defendants requires n^2 integer variables, causing the size of the program to grow quadratically with the size of the scheduling pool. This scalability issue makes exact solutions impractical for large instances due to excessive memory and time requirements. Thus, an important aspect of our integrated optimization and learning framework, described in the next section, is to *avoid solving* (6.7) directly.

6.6. Optimization and Learning for Fair Court Schedules

Problem (6.7) formalizes the fair scheduling problem as an optimization program dependent on unknown preference coefficients $\boldsymbol{Y}$. To address the lack of direct preference data, we use a neural network to learn these preferences from contextual information, such as defendants' demographics features $\boldsymbol{x}_p$. Recall from Section 6.3 that available training data consist of $\boldsymbol{D} = (\boldsymbol{x}_p, \boldsymbol{g}_p, \boldsymbol{Y}_p)_{p=1}^N$, where predictors $\boldsymbol{x}_p$ are known but $\boldsymbol{Y}_p$ are unknown at test time.

A straightforward combined prediction and optimization model trains a neural network $\mathcal{M}_\theta$ with weights θ to predict $\boldsymbol{Y}$ from $\boldsymbol{x}$, by minimizing the squared residuals of its predictions:

$$\min_\theta \frac{1}{N} \sum_p \left\| \mathcal{M}_\theta(\boldsymbol{x_p}) - \boldsymbol{Y_p} \right\|^2 . \tag{6.8}$$

With this approach, problem (6.7) can be approximately specified by

replacing $\boldsymbol{Y}$ with $\mathcal{M}_\theta(\boldsymbol{x})$ in (6.7a). It can then be solved assuming a suitable solution method. However, this "two-stage" approach faces two major challenges:

1. **Scalability issues:** As noted earlier, problem (6.7) is an NP-hard nonlinear integer program whose size grows quadratically with the number of defendants. Thus an approach based on solving (6.7) will lack scalability.
2. **Misaligned training objective:** The training objective in (6.8) minimizes prediction errors rather than optimizing the utility of the resulting schedules, leading to suboptimal outcomes. Literature emphasizes the importance of aligning model training with the end-task objective, as discussed in Section 6.2.

Motivated by these challenges, we propose a more efficient alternative. Instead of minimizing preference prediction errors as in (6.8), our model directly maximizes the OWA value of predicted schedules under ground-truth preference values. under the ground-truth preference values. This requires the computation of an optimal schedule as a function of preference values during each training iteration and performing backpropagation through the optimization process.

6.6.1 End-to-End Trainable Scheduling Model

The proposed end-to-end trainable scheduling model consists of three main components:

1. A neural network $\mathcal{M}_\theta$, which maps known features $\boldsymbol{x}_p$ to predicted preferences $\hat{\boldsymbol{Y}} = \mathcal{M}_\theta(\boldsymbol{x}_p)$.
2. A differentiable module $\boldsymbol{\Pi}$ that maps $\hat{\boldsymbol{Y}}$ to a *predicted permutation matrix* $\boldsymbol{\Pi}(\hat{\boldsymbol{Y}})$, satisfying constraints (6.7b)-(6.7c).
3. A loss function which allows training of $\mathcal{M}_\theta$ to optimize the objective $\mathrm{OWA}_{\mathbf{w}}\left(\boldsymbol{u}^{\mathcal{G}}(\ \boldsymbol{\Pi}(\hat{\boldsymbol{Y}}), \boldsymbol{Y}_p\ )\right)$ expressed in (6.7a) by gradient descent.

Composed of these elements, the resulting end-to-end ML and optimization training objective is:

$$\max_{\theta} \frac{1}{N} \sum_{p} \mathrm{OWA}_{\mathbf{w}}\left(\boldsymbol{u}^{\mathcal{G}}\left(\boldsymbol{\Pi}(\mathcal{M}_\theta(\boldsymbol{x}_p)), \boldsymbol{Y}_p\right)\right). \tag{6.9}$$

Comparing (6.9) with (6.7) highlights the motivation for this architecture: Gradient descent on the empirical objective (6.9) enables $\mathcal{M}_\theta$ to learn predictions $\hat{\boldsymbol{Y}}$ that yields the schedules $\boldsymbol{\Pi}(\hat{\boldsymbol{Y}})$ solving (6.7) for $\boldsymbol{\Pi}^\star(\boldsymbol{Y}_p)$, given features $\boldsymbol{x}_p$. This composite mapping, $\boldsymbol{\Pi} \circ \mathcal{M}_\theta$, achieves the goal of predicting schedules that solve the fair scheduling problem (6.7) *without requiring direct preference data* $\boldsymbol{Y}$! What remains is to determine efficient and *differentiable* implementations of the module $\boldsymbol{\Pi}$ and loss function $\text{OWA}_{\mathbf{w}}$, to enable backpropagation for gradient descent training of (6.9).

6.6.2 Differentiable Matching Layer

Our proposed fair scheduling model relies on a module $\boldsymbol{\Pi}(\boldsymbol{Y})$ which maps preference data to permutation matrices. Note that (6.7) is one such mapping. However, it is neither efficient to compute nor differentiable, thus unsuitable for gradient descent training (6.9). We address the efficiency aspect first, noting that the OWA objective is not required to yield a valid permutation. Instead, consider replacing the OWA objective in (6.7) with the total utility objective (6.3):

$$\boldsymbol{\Pi}(\boldsymbol{Y}) = \text{argmax}_{\boldsymbol{\Pi}} \ \text{Tr}\left(\boldsymbol{Y}^T \boldsymbol{\Pi}\right) \tag{6.10a}$$

$$\text{subject to:} \ \ \boldsymbol{\Pi} \in \{0,1\}^{n \times n} \tag{6.10b}$$

$$\sum_i \Pi_{ij} = 1, \ \ \sum_j \Pi_{ij} = 1, \ \ \forall i, j \in [n] \tag{6.10c}$$

As the sum of individual utilities, the objective (6.10a) is linear. Additionally, we identify the constraints (6.10c) as being *totally unimodular* with integer right-hand side. A well-known result in optimization states that the optimal solution to such a linear program (LP) must be integer-valued Bazaraa et al. (2008).

Thus, we may solve (6.10) by replacing (6.10b) with continuous bounds $\boldsymbol{\Pi} \in [0,1]^{n \times n}$ and solving the resulting LP. In fact, this LP is recognized as the classic *assignment problem*, which admits known solutions in $\mathcal{O}(n^3)$ time Bazaraa et al. (2008). We refer to the mapping (6.10) as the *matching layer*.

Proposition 1. The matching layer (6.10) has complexity $\mathcal{O}(n^3)$.

Proof. The matching layer problem (6.10) is equivalent to the assignment problem, which can be solved in $\mathcal{O}(n^3)$ time using algorithms such as the Hungarian method Kuhn (1955). Specifically, the cost matrix in the assignment problem is given by the negative of the preference matrix $\boldsymbol{Y}$, and the goal is to find the permutation matrix $\boldsymbol{\Pi}$ that maximizes the total utility. Therefore $\boldsymbol{\Pi}(\boldsymbol{Y})$ can be computed in cubic time with respect to the number of defendants n. □

To enable backpropagation for training (6.9), we implement differentiation using a method from Pogančić et al. (2020), which approximates gradients through a linear program by finite differences between two solutions of (6.10) under perturbed inputs $\boldsymbol{Y}$. Thus the combined forward and backward passes through (6.10) are assured $\mathcal{O}(n^3)$ complexity, yielding an efficient and differentiable matching layer.

6.6.3 OWA as a Loss Function

To enable gradient descent training in (6.9), we need a method for backpropagating the OWA loss function. Although the OWA is not differentiable, it is *subdifferentiable* with known subgradients Do & Usunier (2022):

$$\frac{\partial}{\partial \boldsymbol{x}} \mathrm{OWA}_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}_{(\sigma^{-1})}, \tag{6.11}$$

where σ is the sorting permutation for $\boldsymbol{x}$. Our main approach is to implement subgradient descent training for (6.9) using the formula (6.11). However, we also investigate the use of an alternative gradient rule in this thesis. For any convex function f, the *Moreau envelope* f^β is a $\frac{1}{\beta}$ smooth lower-bounding function with the same minimum:

$$f^\beta(\mathbf{x}) = \min_{\mathbf{v}} \; f(\mathbf{v}) + \frac{1}{2\beta}\|\mathbf{v} - \mathbf{x}\|^2. \tag{6.12}$$

It is proven in Do & Usunier (2022) that the gradient of OWA's Moreau envelope is equal to the projection of a vector $\mathbf{x}$ onto the permutahedron $\mathcal{C}(\tilde{\boldsymbol{w}})$ induced by the OWA weights $\boldsymbol{w}$:

$$\frac{\partial}{\partial \boldsymbol{x}} \mathrm{OWA}^\beta_{\boldsymbol{w}}(\mathbf{x}) = \mathrm{proj}_{\mathcal{C}(\tilde{\boldsymbol{w}})}\left(\frac{\boldsymbol{x}}{\beta}\right). \tag{6.13}$$

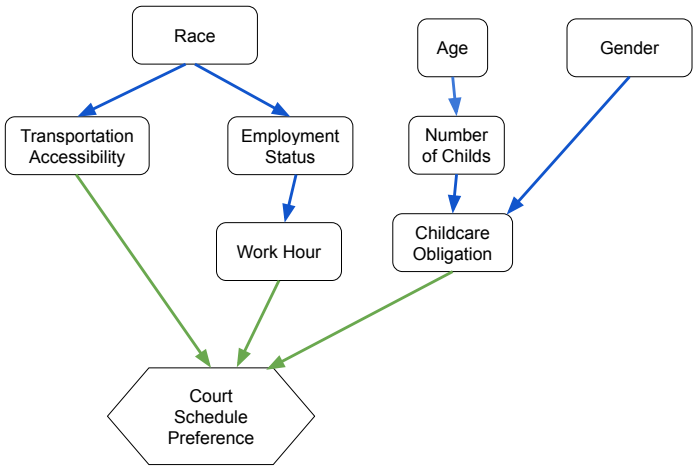


Figure 6.3: Causal factors influencing court schedule preferences: green arrows show direct, blue arrows show indirect relationships

As shown in 3.5.2, the projection solutions converge to the subgradient of the OWA, represented as $\boldsymbol{w}_{(\sigma^{-1})}$ Blondel et al. (2020). Thus, in this work, we use a subgradient approach to enable backpropagation of the OWA loss function.

6.7. Experimental Settings

To validate the effectiveness of our proposed integrated optimization and learning framework for fair court scheduling, we conduct a series of experiments on court scheduling using a novel synthetic dataset. We first focus on the approach adopted to generate the synthetic preference data for court scheduling.

6.7.1 Data Generation Process

The experiments simulate realistic court scheduling scenarios by creating a causal graph that models the data generation process, incorporating individuals' preferences, socioeconomic status, and demographic characteristics. This causal graph, depicted in Figure 6.3, illustrates the relationships among these factors. Each feature is treated as a categorical variable, generated based on the conditional probabilities detailed in Appendix .1, which are elicited by combining, and interviewing court scheduling experts with Census data.

Fairness is essential when resources are limited and demand is high. To simulate a scenario requiring fair scheduling, we generate individuals' preferences to predominantly favor specific time slots. For instance, people dependent on public transportation, working night shifts, or with childcare responsibilities tend to prefer morning slots (8-12 AM). Each individual's preference vector $\boldsymbol{y}_i$ is generated under the constraints that $\sum_j y_{ij} = 1$ and $0 \leq y_{ij} \leq 1$. Additionally, we assume that each person has three top choices, ranked by priority. The second and third choices are set one hour before and after the primary choice. This setup ensures that when specific time slots have limited availability, individuals' second or third preferences are considered, promoting fairness and flexibility in scheduling.

We generate different training sets with varying pool sizes of defendants, specifically $N \in \{25, 50, 100, 250, 500, 4000\}$, where each pool consists of $n = 12$ individuals assigned to corresponding time slots. Each individual is allocated to exactly one slot. The paper evaluates the proposed framework based on both group and individual fairness notions (where the group size is 1 for individual fairness). For group fairness evaluation, the paper creates different settings using protected group attributes such as transportation accessibility, employment status, and work hours. The performance is evaluated on the same test set containing $N = 500$ samples, generated using the same probability distribution as the training data. More details about the probability distributions of our dataset can be found in Appendix (.1).

6.7.2 Model Settings and Evaluation Metrics

Settings. A feedforward neural network $\mathcal{M}_\theta$ is trained to predict for a pool of n candidates, given features $\boldsymbol{x}$, their preference scores $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$ for n available slots. The network consists of two hidden layers, where the size of each successive layer is halved. The model is trained using the Adam optimizer, with a learning rate of 0.01 and a batch size $\in \{64, 128, 512\}$. Results for each hyperparameter setting are taken on average over five random seeds.

Evaluation metrics. All the experiments are evaluated using two key metrics: *regret* and *normalized pairwise distances*. **Regret.** Regret quantifies the loss of optimality in the obtained schedule due to prediction errors in estimated preferences $\hat{\boldsymbol{Y}}$. Regret measures the difference in the OWA objective value between using the true preferences $\boldsymbol{Y}$ and the predicted preferences $\hat{\boldsymbol{Y}}$. Formally, it is defined as:

$$\begin{aligned} \textit{regret}(\hat{\boldsymbol{Y}}, \boldsymbol{Y}) = \mathrm{OWA}_{\mathbf{w}} \left(\boldsymbol{u}^{\mathcal{G}}(\boldsymbol{\Pi}^{\star}(\boldsymbol{Y}), \boldsymbol{Y}) \right) \\ - \mathrm{OWA}_{\mathbf{w}} \left(\boldsymbol{u}^{\mathcal{G}}(\boldsymbol{\Pi}^{\star}(\hat{\boldsymbol{Y}}), \boldsymbol{Y}) \right), \end{aligned}$$

where: $\boldsymbol{Y}$ is the ground-truth preference matrix, $\hat{\boldsymbol{Y}}$ is the predicted matrix, $\boldsymbol{\Pi}^{\star}(\boldsymbol{Y})$ is the optimal schedule for $\boldsymbol{Y}$, $\boldsymbol{\Pi}^{\star}(\hat{\boldsymbol{Y}})$ is the schedule for $\hat{\boldsymbol{Y}}$, and $\boldsymbol{u}^{\mathcal{G}}$ computes the group utility. Lower regret indicates that the predicted schedule closely approximates the optimal one, with zero regret signifying that $\hat{\boldsymbol{Y}}$ produces the optimal schedule under $\boldsymbol{Y}$. Minimizing regret is crucial for ensuring accurate scheduling aligned with the OWA objective.

Normalized pairwise difference. The Normalized Mean Pairwise Difference (NMPD) is an intuitive fairness metric that quantifies how similarly individuals are treated by comparing differences in their outcomes. For a set of individuals with outcomes $(u_1, \ldots, u_n)$, the NMPD is defined as:

$$\textit{NMPD}(\boldsymbol{u}) = \frac{1}{n^2 \bar{\boldsymbol{u}}} \sum_{i=1}^{n} \sum_{j=1}^{n} |u_i - u_j| \quad \text{with } \bar{\boldsymbol{u}} = \frac{1}{n} \sum_{i=1}^{n} u_i \tag{6.14}$$

This metric measures fairness by reflecting how uniformly outcomes are distributed. A lower NMPD indicates more uniform treatment, suggesting a fairer distribution of utility, while a higher NMPD signals greater disparities, potentially highlighting biases or unfairness. Minimizing NMPD is essential for promoting equitable outcomes across individuals.

6.7.3 Baseline Models

To evaluate the effectiveness of our proposed model, we compare it against two baseline methods:

1. **Two-Stage Method**: This standard baseline in Predict-Then-Optimize frameworks, as discussed by Mandi et al. (2024a), is trained using Mean Squared Error (MSE) loss without considering downstream optimization.
2. **Total Utility (TU) Loss**: This method employs an end-to-end learning approach using a differentiable matching layer but optimizes for the total sum of utilities rather than incorporating the OWA objective.

6.8. Results

6.8.1 OWA Utility Regret

Figure 6.4 shows the OWA regret (as a percentage) across four fairness settings: individual fairness, employment status, transportation accessibility, and work hours. OWA regret quantifies the loss of optimality relative to ground-truth preferences, with lower values indicating closer alignment to the optimal solution. Across all settings, the **OWA Loss DQ** model demonstrates the lowest regret, showcasing superior performance. Notably, it outperforms the **TU Loss** model by 7.8% in employment status and 6.4% in transportation accessibility, while also surpassing the **Two-Stage MSE Loss** model by 6.7% and 6% in these respective settings. The performance gains in the individual fairness and work hours settings are more modest, with improvements of 1.7%

and 3.2%, respectively, suggesting that model effectiveness may depend on the dataset's characteristics and the specific fairness constraints applied. When group partitions have no competing demands—i.e., when preferences are non-conflicting—solutions tend to be nearly optimal regardless of the model. This emphasizes the importance of understanding data structure and group preference dynamics when evaluating fairness in scheduling.

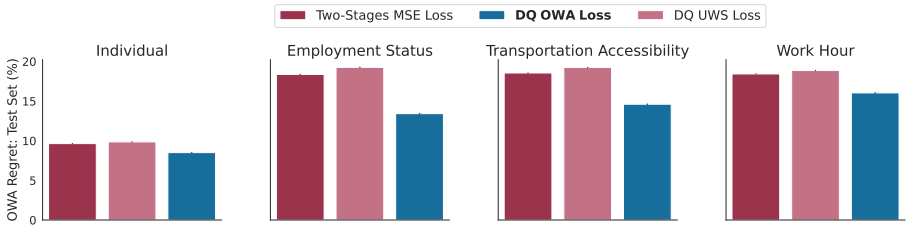


Figure 6.4: OWA utility regret (6.14) in court scheduling across varying fairness levels: individual fairness (first subplot) compared to group fairness (last three subplots). The evaluation is performed on a training dataset with N = 4000 samples.

The **Two-Stage MSE Loss** and **TU Loss** models show similar performance, with the Two-Stage model achieving slightly lower regret in some contexts, such as 1.1% in employment status and 0.4% in transportation accessibility. This advantage arises from the **Two-Stage MSE Loss** model's ability to minimize decision quality loss when large datasets enhance prediction accuracy. However, end-to-end approaches generally perform better, as they integrate prediction and optimization into a unified framework.

Figure 6.5 illustrates the OWA regret (as a percentage) across test sets with varying training data sizes and fairness settings: individual fairness, employment status, transportation accessibility, and work hours. The y-axis represents regret percentages, while the x-axis indicates the number of training samples (25, 50, 100, 250, 500, 4000). Across all settings, regret decreases as the number of samples increases, demonstrating improved model performance with more data. Different fairness

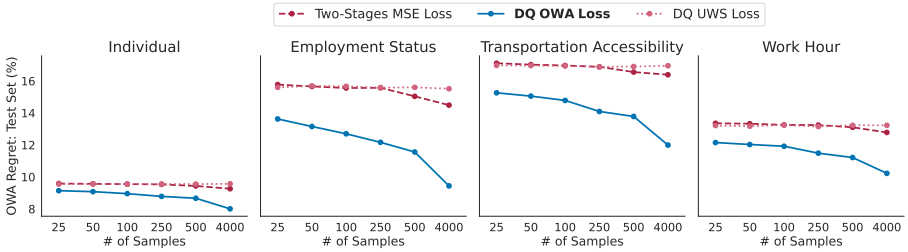


Figure 6.5: Benchmarking OWA Regret (in percentage) across different training data sizes.

settings exhibit varying regret levels, consistent with trends in Figure 6.4. For example, the transportation accessibility setting consistently exhibits higher regret, indicating greater optimization difficulty, while the employment status setting shows a sharper decrease in regret with increased training data. There is minimal reduction in regret between 25 and 100 samples, but a more notable improvement between 250 and 500 samples. The **Two-Stage MSE Loss** and **TU Loss** models perform poorly with smaller datasets and show limited gains as sample sizes grow. In contrast, the **OWA Loss DQ** model achieves a steeper learning curve and outperforms other models, even with small datasets (N=25), achieving regret percentages of 9.2%, 13.8%, 14.8%, and 12.5% for individual fairness, employment status, transportation accessibility, and work hours, respectively. This underscores the model's robustness in scenarios with limited data, a critical advantage for court systems where preference datasets are often sparse.

6.8.2 Normalized mean pairwise distances

Figure 6.6 presents the Normalized Mean Pairwise Difference (NMPD) values across various fairness settings, including employment status, transportation accessibility, and work hours. Across all settings, the NMPD values remain consistently low, typically below 0.08, indicating that the models achieve a high degree of fairness by minimizing disparities in individual outcomes. Among the end-to-end models, both the OWA Loss

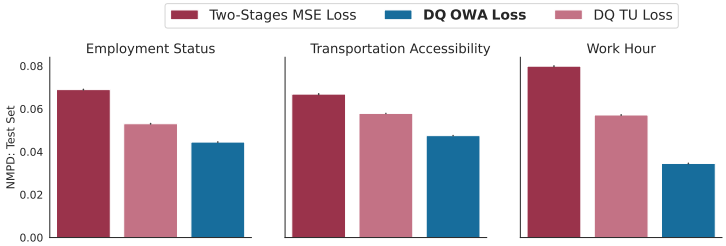


Figure 6.6: Normalized Mean Pairwise Difference (6.14) in court scheduling across varying fairness levels: individual fairness (first subplot) compared to group fairness (last three subplots). The evaluation is performed on a training dataset with N = 4000 samples.

DQ and TU Loss models demonstrate consistently low NMPD values, underscoring their ability to promote equitable treatment. The proposed OWA Loss DQ model performs particularly well, achieving even lower NMPD values than the TU Loss model, thereby ensuring greater fairness across all tested settings. In contrast, the Two-Stage MSE Loss approach exhibits slightly higher NMPD values, especially in the work hours and transportation accessibility settings. This indicates that the traditional two-stage method is less effective at maintaining fairness compared to the advanced end-to-end models.

6.8.3 Running Time

Figure 6.7 presents the run-time comparison of two optimization models for the court scheduling problem: *OWA-ILPs* corresponding to Problem (6.7), and the*Matching Layer* representing Problem (6.10).

The x-axis denotes the number of group partitions, while the y-axis indicates the average time (in seconds) required to solve the scheduling problem for each sample pool, averaged over $N = 1000$ runs. As detailed in Section 6.5, solving Problem (6.7) involves addressing an Integer Linear Program (ILP). ILPs are computationally intensive and face significant scalability challenges as the number of group partitions increases due to the exponential growth of constraints. This scalability issue is

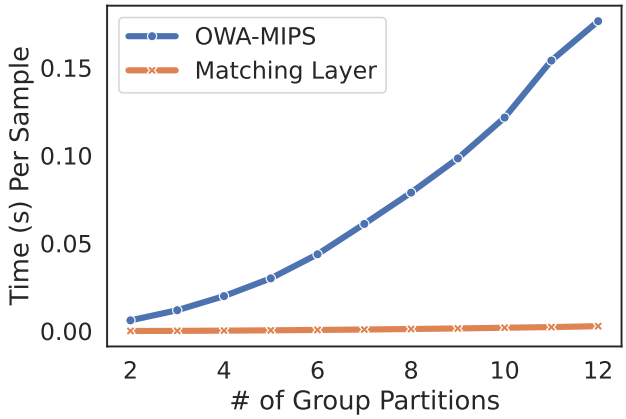


Figure 6.7: Benchmarking the runtime of two optimization models: OWA-ILPs used in the **Two-Stage MSE Loss** model during inference, and the Matching Layer employed in the **OWA Loss DQ** model.

clearly illustrated in Figure 6.7, where the runtime for OWA-ILPs grows polynomially with the number of partitions, rendering it impractical for large-scale scheduling tasks.

In contrast, the Matching Layer uses a linear optimization approach, greatly improving computation efficiency.Its runtime scales linearly with the number of partitions, as shown in Figure 6.7. For 12 partitions, the Matching Layer solves the problem in just 0.002 seconds, making it ideal for real-time, large-scale court scheduling applications.

The significant difference in runtime performance highlights the Matching Layer's technical advantage over traditional ILP-based approaches. By leveraging linear optimization and exploiting problem-specific structures, the Matching Layer reduces computational burden and integrates fairness objectives without sacrificing efficiency. This makes it a highly effective solution for practical court scheduling systems that require both fairness and scalability.

6.9. Conclusions

This thesis addressed the critical problem of fair court scheduling in pretrial processes. Scheduling defendants' court appearances in a manner that is both efficient and fair is essential for upholding justice and maintaining public trust in the legal system. The challenge lies in aligning court schedules with defendants' preferences, which are often unknown and must be predicted from available data while ensuring that the scheduling process adheres to fairness principles to prevent systemic biases and disparate impacts on different groups.

To address these challenges, this thesis proposed an integrated optimization and learning framework that combines machine learning with Fair Ordered Weighted Average (OWA) optimization. Unlike traditional two-stage approaches that handle prediction and optimization separately, our method integrates the prediction of defendants' preferences with the scheduling optimization process. This integration allows for direct optimization of scheduling utility under fairness constraints, leading to more equitable and efficient scheduling outcomes.

Through extensive experiments, we demonstrated that our integrated framework outperforms baseline models in terms of both scheduling optimality and fairness across various scenarios. Our results highlight the effectiveness of incorporating fairness objectives into the learning process, particularly in complex settings with competing group preferences. We believe that this work could pave the way for the utilization of Fair OWA in learning pipelines, enabling a wide range of critical multi-optimization problems across various domains that extend beyond scheduling applications.

CHAPTER 7

Future Directions: Diffusion for Learning-to-Optimize Constrained Optimization

"In all chaos, there is a cosmos; in all disorder, a secret order."

Carl Jung

This chapter focuses on leveraging diffusion models—a recent and powerful class of generative models—to learn to solve constrained optimization problems. We begin with a background on diffusion models and a review of related work applying them to optimization tasks in both continuous and discrete domains. To illustrate the potential of this approach, we first construct a simple diffusion model for a toy quadratic programming (QP) problem. We then examine the capabilities and limitations of current state-of-the-art diffusion-based solvers for combinatorial optimization, which motivates the need for improved methods. Building on these insights, the chapter introduces a new proposed solution designed to address key challenges in existing approaches.

7.1. Background and Related Work

Diffusion-based optimization has recently gained traction as a powerful and generalizable framework, achieving impressive results in a wide range of optimization tasks, from continuous domains to combinatorial spaces. Compared to traditional machine learning approaches, diffusion models offer greater flexibility and the ability to represent complex solution distributions. However, high computational costs and the enforcement of hard constraints during inference remain open challenges.

7.1.1 Diffusion Models

Diffusion models are generative models that learn to reverse a stochastic corruption process applied to clean data. Originally developed for high-dimensional continuous domains such as image generation Ho et al. (2020), these models progressively transform noise samples into structured outputs through a learned sequence of denoising steps.

Formally, the forward process perturbs clean data $\mathbf{x}$ using a parameterized noise schedule α_t:

$$\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{x} + \sqrt{1-\alpha_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \tag{7.1}$$

and the reverse model $p\theta(z_{t-\Delta t} \mid z_t)$ learns to iteratively denoise back to the data distribution. Training is typically done via a variational lower

bound:

$$\mathcal{L}_{\text{VLB}} = \mathcal{L}_{\text{recons}} + \mathcal{L}_{\text{diffusion}} + \mathcal{L}_{\text{prior}}$$

Discrete Diffusion Models. Discrete denoising models adapt this process to combinatorial spaces. D3PM Austin et al. (2021) introduces a multinomial corruption process suitable for token-based data, while SEDD Lou et al. (2023) extends diffusion to continuous time using continuous-time Markov chains (CTMCs).

Masked diffusion models, such as MDLM Sahoo et al. (2024), further simplify training by injecting noise via masking, allowing for tighter ELBO formulations and avoiding the need for complex CTMC-based derivations. The objective becomes:

$$\mathcal{L}_{\text{NELBO}} = \mathbb{E}_q \int_0^1 \frac{\alpha_t'}{1 - \alpha_t} \log\langle \mathbf{x}_\theta(\mathbf{z}_t), \mathbf{x} \rangle \, dt \tag{7.2}$$

These techniques achieve competitive results across tasks in language modeling, protein folding, and DNA sequence generation.

7.1.2 Diffusion Models for Continuous Optimization

A growing body of work explores the application of diffusion models to continuous optimization. These include trajectory optimization Li et al. (2024a), black-box optimization Krishnamoorthy et al. (2023); Chen et al. (2024), and general-purpose frameworks like IRED Du et al. (2024).

IRED combines energy-based models with denoising score matching Hyvärinen (2005); Ho et al. (2020) to learn a sequence of annealed energy landscapes. It predicts noise added to ground-truth optima across noise levels, allowing for gradient-based optimization. While effective, IRED is limited in two ways: (1) it ignores the algorithmic structure of optimization problems, leading to data inefficiency; and (2) it relies on continuous diffusion, which can only approximate discrete structures.

7.1.3 Diffusion Models for Discrete Optimization

In discrete domains, diffusion-based solvers have shown promise on combinatorial optimization tasks such as the Traveling Salesman Problem (TSP), Maximal Independent Set (MIS), and SAT. Unlike deterministic solvers that map directly from problem instances to solutions, diffusion models use iterative refinement and stochasticity to explore multimodal solution spaces.

DIFUSCO Sun & Yang (2023) and T2T Li et al. (2023) apply discrete-time denoising diffusion with multinomial corruption to model complex solution distributions. These models employ gradient-guided sampling and GNN-based denoisers. However, they require hundreds of denoising steps, which limits scalability.

Fast T2T Li et al. (2024b) introduces optimization consistency to accelerate inference by directly predicting solutions from partially corrupted inputs. DiffUCO Sanokowski et al. (2024) takes an unsupervised approach, approximating the Boltzmann distribution over solutions using the objective function as a surrogate energy model. While promising, DiffUCO is restricted to discrete-time and an older diffusion architecture.

Despite these advances, existing methods face challenges in computational efficiency, scalability to large instances, and convergence in multimodal or unstable energy landscapes.

7.2. Preliminaries: Diffusion Model on Learning to Solve Simple CO Problems

7.2.1 Quadratic Programming

Consider a QP problem is defined as follows:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + \mathbf{q}^\top \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \end{aligned} \tag{7.3}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive semi-definite matrix, $\mathbf{q} \in \mathbb{R}^n$, $A \in \mathbb{R}^{p\times n}$, and $\mathbf{b} \in \mathbb{R}^p$. The goal is to find a solution $\mathbf{x}^*$ that

minimizes the quadratic objective while satisfying equality constraints.In our setting, the training dataset consists of triplets $\mathcal{D} = \{(A_i, \mathbf{b}_i, \mathbf{x}_i^*)\}_{i=1}^N$ where each $\mathbf{x}_i^*$ is an optimal solution to a corresponding QP defined by parameters A_i and $\mathbf{b}_i$. Matrix Q and vector $\mathbf{c}$ is fixed across samples.

We train a conditional diffusion model that learns to generate optimal solutions $\mathbf{x}^*$ conditioned on QP instances (A, b). The training proceeds by corrupting the optimal solutions $\mathbf{x}^*$ with Gaussian noise and learning a denoising score function $s_\theta(\mathbf{x}, t \mid A, b)$ that approximates the gradient of the log-density of optimal solutions under a diffusion process.

To model the conditional distribution $p(\mathbf{x}^\star \mid A, b)$, we adopt a DDPM framework Ho et al. (2020). The approach consists of two processes: a forward diffusion process that corrupts optimal solutions with noise, and a learned reverse denoising process conditioned on (A, b).

Forward Process Given an optimal solution $\mathbf{x}_0 = \mathbf{x}^*$, we define the forward noising process as:

$$q(\mathbf{x}_t \mid x_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1 - \bar{\alpha}_t)I), \tag{7.4}$$

where $\{\bar{\alpha}_t\}_{t=1}^T$ is a fixed noise schedule and $t \in \{1, \ldots, T\}$ is the diffusion timestep.

Reverse Process and Training Objective We train a neural network $\epsilon_\theta(\mathbf{x}_t, t \mid A, b)$ to predict the noise added during the forward process. The network is trained to minimize the expected denoising loss:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{\mathbf{x}^*, t, \epsilon \sim \mathcal{N}(0,I)} \left[\|\epsilon - \epsilon_\theta \left(\mathbf{x}_t, t \mid A, b\right)\|_2^2 \right], \tag{7.5}$$

where the noisy input is constructed as:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}^* + \sqrt{1 - \bar{\alpha}_t}\epsilon. \tag{7.6}$$

Inference-Time Sampling : At test time, given a new constraint pair (A, b), a candidate solution is generated by starting from Gaussian noise and iteratively denoising it using the learned conditional model $\epsilon_\theta(\mathbf{x}_t, t \mid A, b)$ until reaching a final sample $\mathbf{x}_0$ that approximates the optimal QP solution.

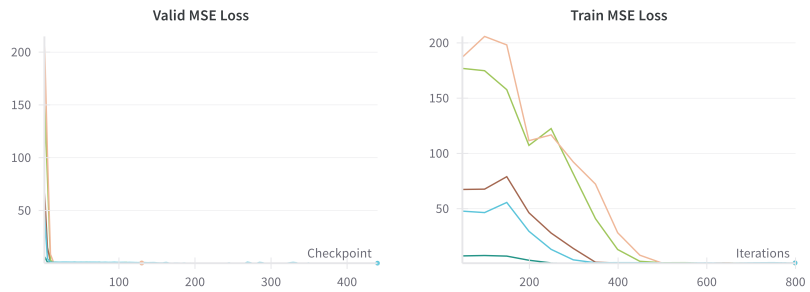


Figure 7.1: Training and validation loss (Eq. 7.5) across different seeds on a synthetic QP task with input $\boldsymbol{x} \in \mathbb{R}^2$,

Result: Figure 7.1 shows that the diffusion framework effectively learns to solve a QP problem on a 2D synthetic dataset, with consistent convergence of training and validation losses. The results indicate that, given constraint information, the model is able to recover optimal solutions with high precision.

7.2.2 Maximal Independent Set

The *Maximal Independent Set* (MIS) problem is a fundamental discrete optimization task over binary variables. Given a graph $G = (V, E)$, the objective is to find a subset of nodes that form an independent set—i.e., no two selected nodes share an edge—while maximizing the size of this set. We represent the solution as a binary vector $\mathbf{x} \in \{0, 1\}^n$, where $x_i = 1$ indicates inclusion of node i in the independent set. This can be formulated as the following integer program:

$$\begin{aligned} \max_{\mathbf{x} \in \{0,1\}^n} \quad & \mathbf{1}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{x}_i + \mathbf{x}_j \leq 1, \quad \forall (i, j) \in E, \end{aligned} \tag{7.7}$$

where the constraints enforce independence by preventing adjacent nodes from being selected simultaneously.

We replicate the setting introduced in Sun & Yang (2023), which pro-

poses the DIFUSCO framework—an approach that leverages discrete denoising diffusion models for solving combinatorial problems like MIS. Our experiments are conducted on the **SATLIB** benchmark, a dataset consisting of graph instances derived from SAT problems encoded in conjunctive normal form (CNF). In this transformation, nodes represent variables or clauses, and edges encode logical relationships, such as mutual exclusion between conflicting literals. The resulting graphs are large, irregular, and present significant combinatorial complexity, making them well-suited for benchmarking neural solvers.

Following Sun & Yang (2023), we generate ground-truth solutions using the KaMIS solver. Our model is trained on a subset of 10,000 instances and evaluated on 1,000 held-out test graphs.

Forward Process We use the discrete diffusion formulation from DIFUSCO to model a Markov chain that gradually corrupts the binary MIS solution $\mathbf{x}_0 = \mathbf{x}^*$ over time. At each timestep t, the binary vector is corrupted via a multinomial transition defined by:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathrm{Cat}(\mathbf{x}_t;\, p = \tilde{\mathbf{x}}_{t-1} Q_t), \tag{7.8}$$

where $\tilde{\mathbf{x}} \in \{0,1\}^{n\times 2}$ is the one-hot encoding of $\mathbf{x}$, and Q_t is a transition matrix that introduces Bernoulli noise governed by a schedule β_t. As $t \to T$, the process converges to a uniform distribution over $\{0,1\}^n$.

Reverse Process and Training Objective To recover clean solutions, we train a denoising model $p_\theta(\tilde{\mathbf{x}}_0 \mid \mathbf{x}_t)$ to estimate the original binary vector given a corrupted sample $\mathbf{x}_t$. The reverse process is modeled via a posterior:

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \sum_{\tilde{\mathbf{x}}_0} q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \tilde{\mathbf{x}}_0) \cdot p_\theta(\tilde{\mathbf{x}}_0 \mid \mathbf{x}_t), \tag{7.9}$$

where $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \tilde{\mathbf{x}}_0)$ is computed analytically. The denoising model is implemented as an Anisotropic Graph Neural Network (AGNN), which incorporates node features, edge structure, and temporal embeddings to predict denoised binary vectors at each step.

Inference and Decoding At inference time, sampling begins from a fully noisy state $\mathbf{x}_T \sim \text{Uniform}(\{0,1\}^n)$. The learned reverse transitions are applied iteratively from $t = T$ to $t = 0$, yielding a final binary vector $\mathbf{x}_0$. This output is post-processed via a greedy decoding strategy to ensure that the final solution is a valid independent set.

Observation We observe that the optimality gap across different training runs varies significantly, ranging from as low as 0.05% to nearly 20%. This variation indicates instability in the training process and suggests a strong dependence on random initialization. This experiment indicates the need for more robust discrete diffusion model for combinatorial optimization problems.

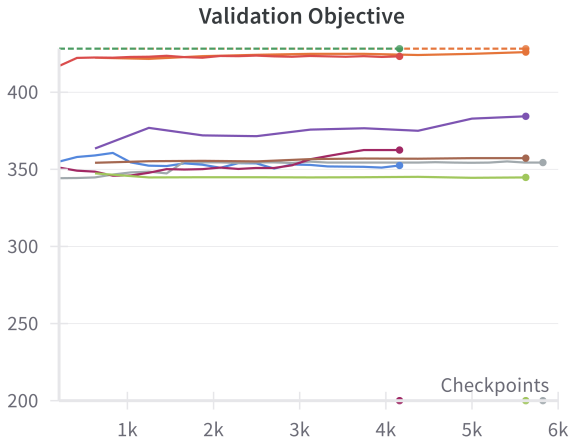


Figure 7.2: Validation objective values (as defined in Eq. (7.7)) across multiple training runs for the MIS experiment. The dashed line denotes the ground truth optimal objective, while the solid curves show predicted objectives from different model initializations.

7.3. Proposed Solutions: Neural Optimization via Energy-Based Diffusion Models

7.3.1 Optimization as Energy-Based Inference

Many real-world optimization problems—such as scheduling, routing, and resource allocation—can be formulated as:

$$\min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{c}, \mathbf{x}) \tag{7.10}$$

where $\mathbf{x} \in \mathcal{C} \subseteq \mathbb{R}^d$ is the decision variable, $\mathbf{c}$ parameterizes the problem, and f is a task-specific cost function. Classical optimization methods work well when f and $\mathcal{C}$ are tractable; however, real-world instances often involve non-convex, high-dimensional, or discrete domains, rendering exact or approximate solvers ineffective.

Instead of solving Eq. (7.10) directly, we reframe optimization as probabilistic inference, sampling solutions from a Boltzmann distribution that favors low-cost decisions:

$$p_B(\mathbf{x}; \beta \mid \mathbf{c}) = \frac{\exp(-\beta E(\mathbf{c}, \mathbf{x}))}{Z}, \quad Z = \int \exp(-E(\mathbf{c}, \mathbf{x}))\, d\mathbf{x}, \tag{7.11}$$

where $\beta = 1/T > 0$ is the inverse temperature. As $\beta \to \infty$, p_B concentrates on optimal solutions. However, sampling from p_B is intractable in high dimensions, especially for discrete problems.

To address this, we train a neural network $q_\theta(\mathbf{x})$ to approximate p_B without explicitly computing Z.

7.3.2 Continuous Domains: Sliced Score Matching and Langevin Sampling

For continuous optimization problems, we approximate the score function $\nabla_{\mathbf{x}} \log p_B(\mathbf{x} \mid \mathbf{c})$ using a neural network $s_\theta(\mathbf{x}_t, t, \mathbf{c})$, trained via Sliced Score Matching (SSM) Song et al. (2019). SSS sidesteps the intractable partition function $\log Z$ since $\nabla_{\mathbf{x}} \log Z = 0$; the gradient of the log-density depends only on the energy landscape. The forward process

adds Gaussian noise to the input, and the model learns to reverse this process:

$$\mathcal{L}_{\text{SSM}} = \mathbb{E}_{\substack{\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0) \\ \mathbf{v} \sim \mathcal{N}(0,\mathbf{I})}} \left[\mathbf{v}^\top \nabla_{\mathbf{x}_t} s_\theta(\mathbf{x}_t, t, \mathbf{c}) \mathbf{v} + \frac{1}{2} \| s_\theta(\mathbf{x}_t, t, \mathbf{c}) \|^2 \right]. \quad (7.12)$$

During inference, samples are generated using annealed Langevin dynamics:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \eta_t s_\theta(\mathbf{x}_t, t, \mathbf{c}) + \sqrt{2\eta_t}\epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \mathbf{I}). \quad (7.13)$$

7.3.3 Discrete Domains: Score Entropy Diffusion

For discrete optimization problems, we adopt the (SEDD) framework Lou et al. (2023). Unlike D3PM Austin et al. (2021), which relies on mean prediction and scales poorly in high dimensions, SEDD directly learns the concrete score:

$$s_\theta(\mathbf{x}, t)_y \approx \frac{p_t(y)}{p_t(\mathbf{x})}, \quad y \neq \mathbf{x}, \quad (7.14)$$

which represents the ratio of transition probabilities between neighboring discrete states. Similar to score matching, the score entropy formulation avoids explicit computation of $\log Z$ Lou et al. (2023). This score is trained using the score entropy loss:

$$\mathcal{L}_{\text{SE}} = \mathbb{E}_{\mathbf{x} \sim p_t} \sum_{y \neq \mathbf{x}} w_{xy} \left(s_\theta(\mathbf{x})_y - \frac{p_t(y)}{p_t(\mathbf{x})} \log s_\theta(\mathbf{x})_y \right), \quad (7.15)$$

which generalizes score matching to discrete domains while ensuring stable optimization.

The forward process is defined via continuous-time Markov transitions with matrices Q_t that perturb the input into a base distribution. During inference, the reverse process is simulated using a τ-leaping strategy, allowing efficient and parallel updates. This design supports arbitrary prompting and performs well in high-dimensional discrete tasks.

7.4. Conclusion

This proposal explores the potential of diffusion models for learning to solve continuous and discrete optimization problems. While prior work demonstrates their expressiveness, challenges remain in scalability, constraint handling, and stability. By adapting diffusion models to structured optimization tasks, this research aims to develop more robust, efficient, and generalizable solvers.

CHAPTER 8

Conclusion

"Every new beginning comes from some other beginning's end."

Seneca

This thesis has aimed to explore how machine learning (ML) and constrained optimization can be integrated to develop fair and scalable solutions to decision-making. We structured this investigation around two key frameworks—Predict-Then-Optimize (PtO) and Learning to Optimize (LtO)—each addressing different challenges in applying to decision-making process. Within each domain, we have described our methodological contributions as well as our original designs aimed at enhancing performance in various application areas, including learning-to-rank, court scheduling, portfolio management, and power systems optimization.

Beyond these individual frameworks, we explored a hybrid approach that combines elements of PtO and LtO to tackle intractable Mixed-Integer Programming (MIP) problems, exemplified through court scheduling. This work explores the synergies between these frameworks, demonstrating how techniques from one setting can benefit the other. By proposing algorithmic designs that blur the boundaries between Learning to Optimize and Predict-Then-Optimize, we aim for a unified field of study that fully integrates ML and optimization as complementary and interdependent disciplines.

As part of our discussion on future directions, we concluded with an exploration of modern generative models—particularly diffusion models—for learning to solve constrained optimization problems across both continuous and discrete domains. These models offer a promising approach for efficiently navigating complex solution landscapes by generating high-quality candidate solutions. Beyond decision-making, this line of research holds broader relevance for scientific fields, as many combinatorial objectives exhibit structural parallels with the Spin Glass model from statistical physics. This connection suggests that advances in generative modeling for optimization could contribute to solving foundational problems in physics, materials science, and network theory.

As machine learning continues to transform computational science and the availability of data grows, informed decision-making will increasingly rely on a fusion of predictive and prescriptive modeling. This thesis highlights the promise of deeply integrating ML and optimization, showing that their synergy enables solutions that are more scalable, fair,

and effective than either approach alone. We hope this work provides a comprehensive perspective on these technologies and inspires further research into how they can expand to different settings.

Bibliography

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019a). Differentiable convex optimization layers. *Advances in neural information processing systems*, *32*.

Agrawal, A., Barratt, S. T., Boyd, S. P., Busseti, E., & Moursi, W. M. (2019b). Differentiating through a cone program.
URL `https://api.semanticscholar.org/CorpusID:121394814`

Amos, B., & Kolter, J. Z. (2017a). Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, (pp. 136–145). JMLR. org.

Amos, B., & Kolter, J. Z. (2017b). Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, (pp. 136–145). PMLR.

Amos, B., Koltun, V., & Kolter, J. Z. (2019). The limited multi-label projection layer.

Arora, R., Basu, A., Mianjy, P., & Mukherjee, A. (2016). Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491*.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., & van den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, vol. 34, (pp. 17981–17993).

Baughman, S. B. (2018). *The Bail Book*. Cambridge, UK: Cambridge University Press.

Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D. (2008). *Linear programming and network flows*. John Wiley & Sons.

Beck, A. (2017). *First-order methods in optimization*. SIAM.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2017). Neural combinatorial optimization with reinforcement learning. *arXiv:1611.09940*.

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, *33*, 9508–9519.

Blondel, M., Teboul, O., Berthet, Q., & Djolonga, J. (2020). Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, (pp. 950–959). PMLR.

Boyd, S., Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Cajas, D. (2021). Owa portfolio optimization: A disciplined convex programming framework.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, (pp. 129–136).

Cappart, Q., Chételat, D., Khalil, E., Lodi, A., Morris, C., & Veličković, P. (2021). Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*.

Chatzos, M., Fioretto, F., Mak, T. W. K., & Hentenryck, P. V. (2020). High-fidelity machine learning approximations of large-scale optimal power flow. *arXiv preprint arXiv:2006.16356*.

Chen, C. S., Beckham, C., Liu, Z., Liu, X., & Pal, C. (2024). Robust

guided diffusion for offline black-box optimization.
URL https://arxiv.org/abs/2410.00983

Chen, Y., & Zhou, A. (2022). Multiobjective portfolio optimization via pareto front evolution. *Complex and Intelligent Systems*, *8*, 4301–4317.
URL https://doi.org/10.1007/s40747-022-00715-8

Chong, K. F. E. (2020). A closer look at the approximation capabilities of neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
URL https://openreview.net/forum?id=rkevSgrtPr

Coffrin, C., Gordon, D., & Scott, P. (2014). NESTA, the NICTA energy system test case archive. *CoRR*, *abs/1411.0359*.
URL http://arxiv.org/abs/1411.0359

Connor, J., Martin, R., & Atlas, L. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, *5*(2), 240–254.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

Criminal Justice Innovation Lab (CJIL) (2022). North carolina court appearance project: Findings and policy solutions from new hanover, orange, and robeson counties.

Dantzig, G. B. (1951). Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, *13*, 339–347.

Deka, D., & Misra, S. (2019). Learning for DC-OPF: Classifying active sets using neural nets. Https://arxiv.org/pdf/1902.05607.

Deka, D., & Misra, S. (2019). Learning for DC-OPF: Classifying active sets using neural nets. In *2019 IEEE Milan PowerTech*.

Detassis, F., Lombardi, M., & Milano, M. (2020). Teaching the old dog new tricks: supervised learning with constraints. In A. Saffiotti, L. Serafini, & P. Lukowicz (Eds.) *Proceedings of the First International Workshop on New Foundations for Human-Centered AI (NeHuAI)*, vol. 2659 of *CEUR Workshop Proceedings*, (pp. 44–51).

Dinh, M. H., Fioretto, F., Mohammadian, M., & Baker, K. (2023). An analysis of the reliability of ac optimal power flow deep learning proxies. In *IEEE PES Innovative Smart Grid Technologies*.
URL `https://ieee-isgt-latam.org`

Dinh, M. H., Kotary, J., & Fioretto, F. (2024a). Differentiable approximations of fair OWA optimization. In *ICML 2024 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*.
URL `https://openreview.net/forum?id=NBt4ZBOFth`

Dinh, M. H., Kotary, J., & Fioretto, F. (2024b). End-to-end learning for fair multiobjective optimization under uncertainty. In *Conference on Uncertainty in Artificial Intelligence*.

Dinh, M. H., Kotary, J., & Fioretto, F. (2024c). Learning fair ranking policies via differentiable optimization of ordered weighted averages. In *ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)*, (pp. 2508–2517).
URL `doi.acm.org?doi=3630106.3661932`

Dinh, M. H., Kotary, J., Gouldin, L. P., Yeoh, W., & Fioretto, F. (2024d). End-to-end optimization and learning of fair court schedules. *CoRR*, *abs/2410.17415*.

Do, V., Corbett-Davies, S., Atif, J., & Usunier, N. (2021). Two-sided fairness in rankings via lorenz dominance. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.) *Advances in Neural Information Processing Systems*, vol. 34, (pp. 8596–8608). Curran Associates, Inc.
URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/48259990138bc03361556fb3f94c5d45-Paper.pdf`

Do, V., & Usunier, N. (2022). Optimizing generalized gini indices for fairness in rankings. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 737–747).

Donti, P. L., Rolnick, D., & Kolter, J. Z. (2020). Dc3: A learning method for optimization with hard constraints. In *ICLR*.

Du, Y., Mao, J., & Tenenbaum, J. B. (2024). Learning iterative reasoning through energy diffusion. In *International Conference on Machine Learning (ICML)*.

Edelman, B., Luca, M., & Svirsky, D. (2017). Racial discrimination in the sharing economy: Evidence from a field experiment. *American economic journal: applied economics*, *9*(2), 1–22.

Elbassuoni, S., Amer-Yahia, S., Ghizzawi, A., & El Atie, C. (2019). Exploring fairness of ranking in online job marketplaces. In *22nd International Conference on Extending Database Technology (EDBT)*.

Elmachtoub, A. N., & Grigas, P. (2021). Smart "predict, then optimize".

Elmachtoub, A. N., Liang, J. C. N., & McNellis, R. (2020). Decision trees for decision-making under the predict-then-optimize framework. URL `https://arxiv.org/abs/2003.00360`

Ferber, A., Wilder, B., Dilkina, B., & Tambe, M. (2020). Mipaal: Mixed integer program as a layer.

Ferguson, A. G. (2022). Courts without court. *Vanderbilt Law Review*, *75*, 1461–1466.

Fioretto, F., Hentenryck, P. V., Mak, T. W., Tran, C., Baldo, F., & Lombardi, M. (2020a). Lagrangian duality for constrained deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, (pp. 118–135). Springer.

Fioretto, F., Mak, T. W., & Van Hentenryck, P. (2020b). Predicting AC opf: Combining deep learning and lagrangian dual methods. In *AAAI*.

Fioretto, F., Mak, T. W., & Van Hentenryck, P. (2020c). Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, (pp. 630–637).

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., & Guo, E. (2016). On differentiating parameterized argmin and argmax problems with application to bi-level optimization.

Gouldin, L. P. (2024). Keeping up appearances. *University of California at Davis Law Review*, *58*. Forthcoming.

Graef, L., Mayson, S. G., Ouss, A., & Stevenson, M. T. (2023). Systemic failure to appear in court. *University of Pennsylvania Law Review*, *172*, 1, 11.

Hardy, G. H., Littlewood, J. E., & Pólya, G. (1952). *Inequalities*. Cambridge university press.

Hasan, F., Kargarian, A., & Mohammadi, A. (2020). A survey on applications of machine learning for optimal power flow. In *2020 IEEE Texas Power and Energy Conference (TPEC)*, (pp. 1–6).

Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of optimization theory and applications*, *4*(5), 303–320.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, vol. 33, (pp. 6840–6851).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hopfield, J., & Tank, D. (1985). Neural computation of decisions in optimization problems. *Biological cybernetics*, *52*, 141–52.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

Huang, C. (2020). Relu networks are universal approximators via piecewise linear or constant functions. *Neural Computation*, *32*(11), 2249–2278.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, *6*(24), 695–709.
URL `http://jmlr.org/papers/v6/hyvarinen05a.html`

Iancu, D. A., & Trichakis, N. (2014). Fairness and efficiency in multiportfolio optimization. *Operations Research*, *62*(6), 1285–1301.
URL `https://doi.org/10.1287/opre.2014.1310`

Institute for Law and Social Research (INSLAW) (1988). Decision-related research on technology utilized by local government: Court scheduling phase ii.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, (pp. 448–456).

Jeong, J., Jaggi, P., Butler, A., & Sanner, S. (2022). An exact symbolic reduction of linear smart Predict+Optimize to mixed integer linear programming. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.) *Proceedings of the 39th International Conference on Machine Learning*, vol. 162 of *Proceedings of Machine Learning Research*, (pp. 10053–10067). PMLR.
URL `https://proceedings.mlr.press/v162/jeong22a.html`

Jia, Y., & Wang, H. (2021). Calibrating explore-exploit trade-off for fair online learning to rank.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.
URL `https://arxiv.org/abs/1412.6980`

Kool, W., Van Hoof, H., & Welling, M. (2018). Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Kostreva, M. M., & Ogryczak, W. (1999). Linear optimization with multiple equitable criteria. *RAIRO-Operations Research-Recherche Opérationnelle*, *33*(3), 275–297.

Kotary, J., Dinh, M. H., & Fioretto, F. (2023). Backpropagation of unrolled solvers with folded optimization. In *International Joint Conference on Artificial Intelligence*, (pp. 1963–1970). ijcai.org.
URL `https://doi.org/10.24963/ijcai.2023/218`

Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). End-to-end constrained optimization learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, (pp. 4475–4482).
URL `https://doi.org/10.24963/ijcai.2021/610`

Kotary, J., Fioretto, F., Van Hentenryck, P., & Zhu, Z. (2022). End-to-end learning for fair ranking systems. In *Proceedings of the ACM Web Conference 2022*, (pp. 3520–3530).

Kreider, J. F., Claridge, D. E., Curtiss, P., Dodier, R., Haberl, J. S., & Krarti, M. (1995). Building Energy Use Prediction and System Identification Using Recurrent Neural Networks. *Journal of Solar Energy Engineering*, *117*(3), 161–166.

Krishnamoorthy, S., Mashkaria, S. M., & Grover, A. (2023). Diffusion models for black-box optimization.
URL `https://arxiv.org/abs/2306.07180`

Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, *2*(1–2), 83–97.

Lan, G. (2013). The complexity of large-scale convex programming under a linear optimization oracle.

Lane, P., & Cox, W. (1976). *Guide to Court Scheduling, 1 - A Framework for Criminal and Civil Courts*. Institute for Law and Social Research.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, A., Ding, Z., Dieng, A. B., & Beeson, R. (2024a). Constraint-aware diffusion models for trajectory optimization.
URL `https://arxiv.org/abs/2406.00990`

Li, Y., Guo, J., Wang, R., & Yan, J. (2023). T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. *NeurIPS*.

Li, Y., Guo, J., Wang, R., Zha, H., & Yan, J. (2024b). Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. *NeurIPS*.

Liu, Y., Gong, C., Yang, L., & Chen, Y. (2020). Dstp-rnn: A dual-stage attention-based rnn for long-term and multivariate time series prediction. *Expert Systems with Applications*, *143*, 113082.

Lou, A., Meng, C., & Ermon, S. (2023). Discrete diffusion language modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*.
URL `https://arxiv.org/abs/2310.16834`

Mandi, J., & Guns, T. (2020). Interior point solving for lp-based prediction+optimisation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Mandi, J., Kotary, J., Berden, S., Mulamba, M., Bucarey, V., Guns, T., & Fioretto, F. (2024a). Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, *TBA*, TBA.

Mandi, J., Kotary, J., Berden, S., Mulamba, M., Bucarey, V., Guns, T., & Fioretto, F. (2024b). Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, *81*, 1623–1701.

Márquez-Neila, P., Salzmann, M., & Fua, P. (2017). Imposing hard constraints on deep networks: Promises and limitations. In *International Conference on Computer Vision (ICCV)*.

McAuliffe, S., Hammer, S., Fishbane, A., & Wilk, A. (2023). National guide to improving court appearances. IDEAS42, 1.
URL `https://www.ideas42.org/wp-content/uploads/2023/05/national-guide-improving-court-appearance.pdf`

Nasdaq (2022). Nasdaq end of day us stock prices. `https://data.nasdaq.com/databases/EOD/documentation`. Accessed: 2023-08-15.

National Council of Juvenile and Family Court Judges (2021). Research report: Assessing time-certain calendaring dockets. Tech. rep., National Council of Juvenile and Family Court Judges. Accessed: 2024-10-03.
URL `IncludetheURLhereifavailable`

Ng, Y., Misra, S., Roald, L., & Backhaus, S. (2018). Statistical learning for DC optimal power flow. In *Power Systems Computation Conference*.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Nowak, A., Villar, S., Bandeira, A. S., & Bruna, J. (2018). Revised note on learning algorithms for quadratic assignment with graph neural networks.

Ogryczak, W., Luss, H., Pióro, M., Nace, D., & Tomaszewski, A. (2014). Fair Optimization and Networks: A Survey. *Journal of Applied Mathematics*, *2014*(SI08), 1 – 25.
URL `https://doi.org/10.1155/2014/612018`

Ogryczak, W., & Śliwiński, T. (2003). On solving linear programs with the ordered weighted averaging objective. *European Journal of Operational Research*, *148*(1), 80–91.

Pan, X., Zhao, T., & Chen, M. (2019). DeepOPF: Deep neural network for dc optimal power flow. In *SmartGridComm*, (pp. 1–6).

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Pathak, D., Krahenbuhl, P., & Darrell, T. (2015). Constrained convolutional neural networks for weakly supervised segmentation. In *International Conference on Computer Vision (ICCV)*.

Perron, L. (2011). 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*, (pp. 2–2). Springer.

Pogančić, M. V., Paulus, A., Musil, V., Martius, G., & Rolinek, M. (2019). Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*.

Pogančić, M. V., Paulus, A., Musil, V., Martius, G., & Rolinek, M. (2020). Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations (ICLR)*.

Powell, M. J. (1969). A method for nonlinear constraints in minimization problems. *Optimization*, (pp. 283–298).

Sahoo, S. S., Arriola, M., Schiff, Y., Gokaslan, A., Marroquin, E., Chiu, J. T., Rush, A., & Kuleshov, V. (2024). Simple and effective masked diffusion language models.
URL `https://arxiv.org/abs/2406.07524`

Salas, J., & Yepes, V. (2020). Enhancing sustainability and resilience through multi-level infrastructure planning. *International Journal of Environmental Research and Public Health*, *17*(3), 962.

Sanokowski, S., Hochreiter, S., & Lehner, S. (2024). A diffusion model framework for unsupervised neural combinatorial optimization. *ICML*.

Siddique, U., Weng, P., & Zimmer, M. (2020). Learning fair policies in multiobjective (deep) reinforcement learning with average and discounted rewards. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Singh, A., & Joachims, T. (2018). Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (pp. 2219–2228).

Singh, A., & Joachims, T. (2019). Policy learning for fairness in ranking.

Song, Y., Garg, S., Shi, J., & Ermon, S. (2019). Sliced score matching: A scalable approach to density and score estimation.
URL `https://arxiv.org/abs/1905.07088`

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, vol. 15, (pp. 1929–1958).

Sun, W., et al. (2020). Evolution and impact of bias in human and machine learning algorithm interaction. *PLOS ONE*.
URL `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0235502`

Sun, Z., & Yang, Y. (2023). Difusco: Graph-based diffusion solvers for combinatorial optimization. *NeurIPS*.

Terlouw, T., AlSkaif, T., Bauer, C., & van Sark, W. (2019). Multi-objective optimization of energy arbitrage in community energy storage systems using different battery technologies. *Applied Energy*, *239*, 356–372.
URL `https://www.sciencedirect.com/science/article/pii/S0306261919302478`

Tran, C., Fioretto, F., & Hentenryck, P. V. (2021). Differentially private and fair deep learning: A lagrangian dual approach. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations, (ICLR)*.

Velloso, A., & Van Hentenryck, P. (2020). Combining deep learning and optimization for security-constrained optimal power flow. *arXiv:2007:2007.07002*.

Verma, A. (2009). *Power grid security analysis: An optimization approach*. Ph.D. thesis, Columbia University.

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS*, (pp. 2692–2700).

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57.

Wilder, B., Dilkina, B., & Tambe, M. (2019). Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *AAAI*, vol. 33, (pp. 1658–1665).

Wilson, G., & Pawley, G. (1988). On the stability of the travelling salesman problem algorithm of hopfield and tank. *Biological Cybernetics*, *58*(1), 63–70.

Xu, J., Chen, C., Xu, G., Li, H., & Abib, E. R. T. (2010). Improving quality of training data for learning to rank using click-through data. In *Proceedings of the third ACM international conference on Web search and data mining*, (pp. 171–180).

Yadav, H., Du, Z., & Joachims, T. (2019). Fair learning-to-rank from implicit feedback. *DeepAI*.

URL https://deepai.org/publication/fair-learning-to-rank-from-implicit-feedback

Yager, R. R. (1993). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. In D. Dubois, H. Prade, & R. R. Yager (Eds.) *Readings in Fuzzy Sets for Intelligent Systems*, (pp. 80–87). Morgan Kaufmann.
URL https://www.sciencedirect.com/science/article/pii/B9781483214504500110

Yager, R. R., & Kacprzyk, J. (2012). *The Ordered Weighted Averaging Operators: Theory and Applications*. Springer Publishing Company, Incorporated.

Yang, Y., Yang, Z., Yu, J., Zhang, B., Zhang, Y., & Yu, H. (2020). Fast calculation of probabilistic power flow: A model-based deep learning approach. *IEEE Transactions on Smart Grid*, *11*(3), 2235–2244.

Zamzam, A., & Baker, K. (2020). Learning optimal solutions for extremely fast AC optimal power flow. In *IEEE SmartGridComm*.

Zehlike, M., Bonchi, F., Castillo, C., Hajian, S., Megahed, M., & Baeza-Yates, R. (2017). Fa*ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, (p. 1569–1578). New York, NY, USA: Association for Computing Machinery.
URL https://doi.org/10.1145/3132847.3132938

Zehlike, M., & Castillo, C. (2020). Reducing disparate exposure in ranking: A learning to rank approach. In *Proceedings of the web conference 2020*, (pp. 2849–2855).

Zettler, H., & Morris, R. (2015). An exploratory assessment of race and gender-specific predictors of failure to appear in court among defendants released via a pretrial services agency. *Criminal Justice Review*, *40*.

Appendices

Appendix for Chapter 6

"Knowledge is like a sphere; the more its volume expands, the greater its contact with the unknown."

Blaise Pascal

.1. Causal Graph Conditional Probability Tables

Below are the conditional distributions of the causal graph depicted in Figure 6.3.

Table 1: Distribution of Race among Defendants

Race	**P(x)**
White	1/2
Non White	1/2

Table 2: Distribution of Age Groups among Defendants

Age Group	**P(x)**
Below 18	1/20
18-54	4/5
Above 55	3/20

Table 3: Distribution of Gender among Defendants

Gender	**P(x)**
Male	9/20
Female	11/20

Table 4: Transportation Accessibility Conditional on Race

Transportation Accessibility (x)	**Race (y)**	**P(x|y)**
Public transportation	White	$4/5$
Private transportation	White	$1/5$
Public transportation	Non White	$3/5$
Private transportation	Non White	$2/5$

Table 5: Employment Status Conditional on Race

Employment Status (x)	**Race (y)**	**P(x|y)**
Employed	White	$4/5$
Unemployed	White	$1/5$
Employed	Non White	$7/10$
Unemployed	Non White	$3/10$

Table 6: Work Hour Conditional on Employment Status

Work Hour	**Employment Status**	**P(x|y)**
Day shift	Employed	$1/2$
Night shift	Employed	$3/10$
Irregular shift	Employed	$9/50$
No shift	Employed	$1/50$
Day shift	Unemployed	0.0
Night shift	Unemployed	0.0
Irregular shift	Unemployed	0.0
No shift	Unemployed	1.0

Table 7: Number of Children Conditional on Age Group

Number of Children	**Age Group**	**P(x|y)**
No child	Under 18	$19/20$
+1 child	Under 18	$1/20$
No child	18-54	$11/20$
+1 child	18-54	$9/20$
No child	Above 55	$1/5$
+1 child	Above 55	$4/5$

Table 8: Childcare Obligation Conditional on Gender and Number of Children

Childcare Obligation	**Gender, Number of Children**	**P(x|y,z)**
No obligation	Female, no child	1.0
No obligation	Female, +1 child	$3/10$
No obligation	Male, no child	1.0
No obligation	Male, +1 child	$17/20$
Have obligation	Female, no child	0.0
Have obligation	Female, +1 child	$7/10$
Have obligation	Male, no child	0.0
Have obligation	Male, +1 child	$3/20$

Table 9: Schedule preferences (Part 1)

Schedule Preference (o)	**Transportation Accessibility (l), Work Hour (m), Childcare Obligation (n)**	**P(o | l,m,n)**
8:00AM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	1/6
8:30AM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	1/6
9:00AM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	1/6
9:30AM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	1/6
10:00AM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	1/6
10:30AM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	1/6
1:00PM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	0
1:30PM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	0
2:00PM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	0
2:30PM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	0
3:00PM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	0
3:30PM	Public Transportation, Day or Regular Shift, Have or Don't Have Childcare Obligation	0

Table 10: Schedule preferences (Part 2)

Schedule Preference (o)	**Transportation Accessibility (l), Work Hour (m), Childcare Obligation (n)**	**P(o | l,m,n)**
8:00AM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
8:30AM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
9:00AM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
9:30AM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	1/3
10:00AM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	1/3
10:30AM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	1/3
1:00PM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
1:30PM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
2:00PM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
2:30PM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
3:00PM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0
3:30PM	Public Transportation, Night Shift, Have or Don't Have Childcare Obligation	0

Table 11: Schedule preferences (Part 3)

Schedule Preference (o)	**Transportation Accessibility (l), Work Hour (m), Childcare Obligation (n)**	**P(o | l,m,n)**
8:00AM	Private Transportation, Day Shift, Have Childcare Obligation	1/6
8:30AM	Private Transportation, Day Shift, Have Childcare Obligation	1/6
9:00AM	Private Transportation, Day Shift, Have Childcare Obligation	1/6
9:30AM	Private Transportation, Day Shift, Have Childcare Obligation	1/6
10:00AM	Private Transportation, Day Shift, Have Childcare Obligation	1/6
10:30AM	Private Transportation, Day Shift, Have Childcare Obligation	1/6
1:00PM	Private Transportation, Day Shift, Have Childcare Obligation	0
1:30PM	Private Transportation, Day Shift, Have Childcare Obligation	0
2:00PM	Private Transportation, Day Shift, Have Childcare Obligation	0
2:30PM	Private Transportation, Day Shift, Have Childcare Obligation	0
3:00PM	Private Transportation, Day Shift, Have Childcare Obligation	0
3:30PM	Private Transportation, Day Shift, Have Childcare Obligation	0

Table 12: Schedule preferences (Part 4)

Schedule Preference (o)	**Transportation Accessibility (l), Work Hour (m), Childcare Obligation (n)**	**P(o | l,m,n)**
8:00AM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	1/4
8:30AM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	1/4
9:00AM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	1/4
9:30AM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	1/4
10:00AM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
10:30AM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
1:00PM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
1:30PM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
2:00PM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
2:30PM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
3:00PM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0
3:30PM	Private Transportation, Night or Irregular Shift, Have or Don't Have Childcare Obligation	0